

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTERNATIONAL BUSINESS MACHINES CORPORATION

Petitioner

v.

SECURITY FIRST INNOVATIONS, LLC

Patent Owner

Case IPR2025-01200

U.S. Patent No. 8,271,802

Claims 1-27

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.....	15
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.....	42
D. Recovering from Data Loss.....	45
VI. The '802 Patent.....	55
A. Prosecution History	57
VII. Level of Ordinary Skill in the Art	59
VIII. Claim Construction.....	59

IX.	Specific Grounds For Challenge.....	60
A.	Ground I: Torre And Tajima Render Obvious Claims 1-3, 5, 9, 10-12, 14, 18, 19-21, 23, 27.	60
1.	Torre/Tajima Color-Coding Scheme	60
2.	Overview of Torre (EX1003).....	61
3.	Overview of Tajima (EX1004)	70
4.	Combining Torre With Tajima (“Torre/Tajima”).....	74
i.	Torre System With Tag That Includes Encryption Key As Taught By Tajima (“Torre/Tajima System- 1”)	74
a.	Reasons to Combine Torre And Tajima As Provided In Torre/Tajima System-1	81
ii.	Torre/Tajima System-1 In View Of Tajima’s Data Remapping And Splitting Method (“Torre/Tajima System-2”)	90
a.	Reasons to Combine Torre/Tajima System-1 Further with Tajima As Provided In Torre/Tajima System-2	96
5.	Claim 1: A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of	

shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.....	103
i. [1Pre] A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising:.....	103
ii. [1A] creating integrity information using the data set;.....	106
iii. [1B] encrypting the data set based on an encryption key to produce an encrypted data set;	108
iv. [1C] generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed;.....	114
a. Torre/Tajima System-1	114
b. Torre/Tajima System-2	124
v. [1D] separating the encrypted data set into the plurality of shares based on the data splitting information;	128
a. Torre/Tajima System-1	128
b. Torre/Tajima System-2	130
vi. [1E] including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and.....	132
vii. [1F] causing the plurality of shares to be stored in respective separate storage locations;.....	135

viii.	[1G] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.	137
6.	Claim 2: The method of claim 1, wherein the step of separating the encrypted data set comprises using a deterministic technique.	140
i.	Torre/Tajima System-1	140
ii.	Torre/Tajima System-2	144
7.	Claims 3 and 5	145
i.	[3] The method of claim 1, wherein the step of separating the encrypted data set comprises using a substantially random technique.	145
	[5] The method of claim 1, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially randomly [sic] distribution of the encrypted data set.	145
8.	Claim 9: The method of claim 1, wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices.	148
9.	Claim 10: A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on	

the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.....150

i. [10Pre] A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of.....151

10. Claim 19: A computer system for securing a data set, the system comprising: at least one processor; a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.....153

i. [19Pre] A computer system for securing a data set, the system comprising: 154

ii.	[19A] at least one processor;	154
iii.	[19B] a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of:	155
11.	Claims 11-12, 14, 18, 20-21, 23, and 27.....	155
B.	Ground II: Torre, Tajima, And Aldridge Render Obvious Claims 4, 13, and 22.	156
1.	Torre/Tajima/Aldridge Color-Coding Scheme.....	156
2.	Overview of Aldridge (EX1005)	157
3.	Torre/Tajima With Aldridge (“Torre/Tajima/Aldridge”).....	159
a.	Reasons To Combine Torre/Tajima With Aldridge	161
4.	Claim 4: The method of claim 1, further comprising causing a plurality of data units in each of the shares to be rearranged relative to one another after the separating step.	164
5.	Claim 13: The non-transitory computer readable medium of claim 10, wherein the method further comprises causing a plurality of data units of the encrypted data set to be rearranged relative to one another after the separating step.....	167
	Claim 22: The system of claim 19, wherein the method further comprises causing a plurality of data units of the encrypted data set to be rearranged relative to one another after the separating step.	167
C.	Ground III: Torre, Tajima, And Krawczyk Render Obvious Claims 6, 15, and 24.....	168

1.	Torre/Tajima/Krawczyk Color-Coding Scheme.....	168
2.	Overview of Krawczyk (EX1006).....	169
3.	Torre/Tajima In Combination With Krawczyk ("Torre/Tajima/Krawczyk")	170
4.	Claim 6: The method of claim 1, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.	177
5.	Claim 15: The non-transitory computer readable medium of claim 10, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.....	178
	Claim 24: The system of claim 19, wherein data indicative of the encryption key comprises data that was created using a Shamir algorithm.....	178
D.	Ground IV: Torre, Tajima, And Foster Render Obvious Claims 7-8, 16-17, and 25-26.	178
1.	Torre/Tajima/Foster Color-Coding Scheme	179
2.	Overview of Foster (EX1007)	179
3.	Torre/Tajima In Combination With Foster ("Torre/Tajima/Foster").....	181
i.	Reasons To Combine Torre/Tajima System with Foster's Key Encryption Method	184
4.	Claim 7: The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.....	189
i.	[7A] The method of claim 1, wherein the step of including data indicative of the encryption key	

	comprises: encrypting the encryption key with a second key; and.....	189
ii.	[7B] including in the plurality of shares data indicative of the encrypted encryption key.	189
5.	Claim 8: The method of claim 7 further comprising the method step of storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.	191
6.	Claim 16: The non-transitory computer readable medium of claim 10, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.....	194
	Claim 17: The non-transitory computer readable medium of claim 16 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.....	194
	Claim 25: The system of claim 19, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.....	194
	Claim 26: The system of claim 25 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.	194
E.	Ground V: Orsini And Torre Render Obvious Claims 1-5, 9-14, 18-23, and 27.....	195

1.	Torre/Tajima/Orsini Color-Coding Scheme	195
2.	Overview of Orsini (EX1008)	196
3.	Orsini In Combination With Torre (“Orsini/Torre”)	202
i.	Reasons To Combine Orsini and Torre	207
4.	Claim 1: A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.....	213
i.	[1Pre] A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising:.....	213
ii.	[1A] creating integrity information using the data set	214
iii.	[1B] encrypting the data set based on an encryption key to produce an encrypted data set;	214
iv.	[1C] generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed;	217

- v. [1D] separating the encrypted data set into the plurality of shares based on the data splitting information; 219
- vi. [1E] including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and.....222
- vii. [1F] causing the plurality of shares to be stored in respective separate storage locations;.....224
- viii. [1G] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares. 225
- 5. Claim 2: The method of claim 1, wherein the step of separating the encrypted data set comprises using a deterministic technique. 226
- Claim 3: The method of claim 1, wherein the step of separating the encrypted data set comprises using a substantially random technique. 226
- Claim 5: The method of claim 1, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially randomly distribution of the encrypted data set. 226
- 6. Claim 4: The method of claim 1, further comprising causing a plurality of data units in each of the shares to be rearranged relative to one another after the separating step. 228
- 7. Claim 9: The method of claim 1, wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices. 229
- 8. Claim 10: A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer

system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.....230

i. [10Pre] A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of.....231

9. Claim 19: A computer system for securing a data set, the system comprising: at least one processor; a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality

	of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.....	233
10.	Claims 11-14, 18, 20-23, and 27.....	234
F.	Ground VI: Orsini, Torre, And Krawczyk Render Obvious Claims 6, 15, and 24.....	235
1.	Orsini/Torre/Krawczyk Color-Coding Scheme.....	235
2.	Claim 6: The method of claim 1, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.	235
3.	Reasons to Combine Orsini/Torre with Krawczyk.....	237
4.	Claim 15: The non-transitory computer readable medium of claim 10, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.....	242
	Claim 24: The system of claim 19, wherein data indicative of the encryption key comprises data that was created using a Shamir algorithm.....	242
G.	Ground VII: Orsini, Torre, And Foster Render Obvious Claims 7-8, 16-17, and 25-26.	243
1.	Orsini/Torre/Foster Color-Coding Scheme	243
2.	Orsini/Torre In Combination With Foster (“Orsini/Torre/Foster”)	243
i.	Reasons To Combine Orsini/Torre With Foster.....	246
3.	Claim 7: The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second	

key; and including in the plurality of shares data indicative of the encrypted encryption key.....	250
i. [7A] The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and.....	250
ii. [7B] including in the plurality of shares data indicative of the encrypted encryption key.	250
4. Claim 8: The method of claim 7 further comprising the method step of storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.	252
5. Claim 16: The non-transitory computer readable medium of claim 10, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.....	254
Claim 17: The non-transitory computer readable medium of claim 16 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.....	254
Claim 25: The system of claim 19, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.....	254
Claim 26: The system of claim 25 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing	

	less than all, but at least a threshold number of, the plurality of shares, and the second key.	254
X.	Conclusion	254
XI.	Availability for Cross-Examination	255
XII.	Right to Supplement	255
XIII.	Jurat.....	255

LISTING OF CLAIMS

Claim 1

[1Pre] A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising:

[1A] creating integrity information using the data set;

[1B] encrypting the data set based on an encryption key to produce an encrypted data set;

[1C] generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed;

[1D] separating the encrypted data set into the plurality of shares based on the data splitting information;

[1E] including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and

[1F] causing the plurality of shares to be stored in respective separate storage locations;

[1G] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.

Claim 2

[2] The method of claim 1, wherein the step of separating the encrypted data set comprises using a deterministic technique.

Claim 3

[3] The method of claim 1, wherein the step of separating the encrypted data set comprises using a substantially random technique.

Claim 4

[4] The method of claim 1, further comprising causing a plurality of data units in each of the shares to be rearranged relative to one another after the separating step.

Claim 5

[5] The method of claim 1, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially random distribution of the encrypted data set.

Claim 6

[6] The method of claim 1, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.

Claim 7

[7] The method of claim 1, wherein the step of including data indicative of the encryption key comprises:

[7A] encrypting the encryption key with a second key; and

[7B] including in the plurality of shares data indicative of the encrypted encryption key.

Claim 8

[8] The method of claim 7 further comprising the method step of storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

Claim 9

[9] The method of claim 1, wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices.

Claim 10

[10Pre] A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of:

[10A] creating integrity information using the data set;

[10B] encrypting the data set based on an encryption key to produce an encrypted data set;

[10C] generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed;

[10D] separating the encrypted data set into the plurality of shares based on the data splitting information;

[10E] including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and

[10F] causing the plurality of shares to be stored in respective separate storage locations;

[10G] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.

Claim 11

[11] The non-transitory computer readable medium of claim 10, wherein the step of separating the encrypted data set comprises using a deterministic technique.

Claim 12

[12] The non-transitory computer readable medium of claim 10, wherein the step of separating the encrypted data set comprises using a substantially random technique.

Claim 13

[13] The non-transitory computer readable medium of claim 10, wherein the method further comprises causing a plurality of data units of the encrypted data set to be rearranged relative to one another after the separating step.

Claim 14

[14] The non-transitory computer readable medium of claim 10, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially randomly distribution of the encrypted data set.

Claim 15

[15] The non-transitory computer readable medium of claim 10, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.

Claim 16

[16] The non-transitory computer readable medium of claim 10, wherein the step of including data indicative of the encryption key comprises:

[16A] encrypting the encryption key with a second key; and

[16B] including in the plurality of shares data indicative of the encrypted encryption key.

Claim 17

[17] The non-transitory computer readable medium of claim 16 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

Claim 18

[18] The non-transitory computer readable medium of claim 10, wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices.

Claim 19

[19Pre] A computer system for securing a data set, the system comprising:

[19A] at least one processor;

[19B] a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of:

[19C] creating integrity information using the data set;

[19D] encrypting the data set based on an encryption key to produce an encrypted data set;

[19E] generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed;

[19F] separating the encrypted data set into the plurality of shares based on the data splitting information;

[19G] including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and

[19H] causing the plurality of shares to be stored in respective separate storage locations;

[19I] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.

Claim 20

[20] The system of claim 19, wherein the step of separating the encrypted data set comprises using a deterministic technique.

Claim 21

[21] The system of claim 19, wherein the step of separating the encrypted data set comprises using a substantially random technique.

Claim 22

[22] The system of claim 19, wherein the method further comprises causing a plurality of data units of the encrypted data set to be rearranged relative to one another after the separating step.

Claim 23

[23] The system of claim 19, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially randomly distribution of the encrypted data set.

Claim 24

[24] The system of claim 19, wherein data indicative of the encryption key comprises data that was created using a Shamir algorithm.

Claim 25

[25] The system of claim 19, wherein the step of including data indicative of the encryption key comprises:

[25A] encrypting the encryption key with a second key; and

[25B] including in the plurality of shares data indicative of the encrypted encryption key.

Claim 26

[26] The system of claim 25 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by

accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

Claim 27

[27] The system of claim 19, wherein the separate storage locations are located on at least two separate storage devices.

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. 8,271,802 (the “’802 Patent”).

I. Background and Qualifications

3. My *curriculum vitae*, which includes a more detailed summary of my background, experience and publications, is attached as EX1018.

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 '802 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 ’802 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 '802 patent.

II. Materials Considered

30. I understand that the '802 Patent has a filing date of February 10, 2012, claiming priority to U.S. Provisional Application Nos. 60/622,146, filed October 25, 2004, and 60/718,185, filed September 16, 2005. EX1001 ('802 Patent); EX1014 (Provisional Application No. 60/622,146); EX1015 (Provisional Application No. 60/718,185).

31. Therefore, I assume that the earliest possible priority date of the '802 Patent is October 25, 2004.

32. I have considered issues from the perspective of a person of ordinary skill in the art as described in §VII below as of October 25, 2004.

33. I have reviewed the following documents in preparing this declaration. I understand that these documents are prior art to the '802 Patent.

- U.S. Patent Application Publication No. 2003/0065656 (“Torre”) (EX1003), published on April 3, 2003
- U.S. Patent Application Publication No. 2003/0028493 (“Tajima”) (EX1004), published on February 6, 2003
- U.S. Patent Publication No. 2004/0030921 (“Aldridge”) (EX1005), published on February 12, 2004
- “Secret Sharing Made Short,” by Hugo Krawczyk, Advances in Cryptology - CRYPTO ’93, 13th Annual International Cryptology Conference, LNCS 773, 136-146, 1993 (“Krawczyk”) (EX1006), published on August 22, 1993
- U.S. Patent Application Publication No. 2003/0200176 (“Foster”) (EX1007), published on October 23, 2003
- U.S. Patent Application Publication No. 2004/0049687 (“Orsini”) (EX1008), published on March 11, 2004

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

Exhibit No.	DESCRIPTION
1001	U.S. Patent No. 8,271,802 (“’802 Patent”)
1003	U.S. Patent Publication No. 2003/0065656 (“Torre”)
1004	U.S. Patent Publication No. 2003/0028493 (“Tajima”)
1005	U.S. Patent Publication No. 2004/0030921 (“Aldridge”)
1006	“Secret Sharing Made Short,” Hugo Krawczyk, Advances in Cryptology - CRYPTO ’93, 13th Annual International Cryptology Conference, LNCS 773, 136-146, 1993 (“Krawczyk”)

Exhibit No.	DESCRIPTION
1007	U.S. Patent Publication No. 2003/0200176 (“Foster”)
1008	U.S. Patent Publication No. 2004/0049687 (“Orsini”)
1009	File History of U.S. Patent No. 8,271,802 (“’802 File History”)
1010	File History of U.S. Patent Application No. 11/258,839 (“’839 App. File History”)
1011	File History of U.S. Patent Application No. 13/371,363 (“’363 App. File History”)
1012	Comparison Of The ’802 Patent’s Specification And Orsini
1013	<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”)
1014	U.S. Provisional Application No. 60/622,146 (“Provisional Application No. 60/622,146”)
1015	U.S. Provisional Application No. 60/718,185 (“Provisional Application No. 60/718,185”)
1016	Microsoft Computer Dictionary, 5 th ed., 2002, excerpts (“Microsoft Computer Dictionary”)
1017	Bruce Schneier, <i>Applied Cryptography</i> , 2 nd ed., 1996, excerpts (“Schneier”)
1018	Curriculum Vitae of Dr. Erez Zadok
1019	Yitzhak Birk, “Random RAIDs with Selective Exploitation of Redundancy for High Performance Video Servers,” IEEE 1997 (“Birk”)
1020	U.S. Patent Publication No. 2003/0016596 (“Chiquoine”)
1022	John Kubiawicz, et al., “OceanStore: An Architecture for Global-Scale Persistent Storage,” ACM 2000 (“Kubiawicz”)
1023	U.S. Patent No. 6,226,618 (“Downs”)
1024	P. Venkat Rangan et al., “Designing File Systems for Digital Video and Audio,” ACM SIGOPS Operating Systems Review, Volume 25, Issue 5, 81–94 (1991) (“Rangan”)

III. Summary of Opinions

35. It is my opinion that claims 1-27 of the '802 Patent are obvious in view of Grounds identified below:

Ground	References	Challenged Claims
I	Torre, Tajima	1-3, 5, 9-12, 14, 18-21, 23, 27.
II	Torre, Tajima, Aldridge	4, 13, 22.
III	Torre, Tajima, Krawczyk	6, 15, 24.
IV	Torre, Tajima, Foster	7-8, 16-17, 25-26.
V	Orsini, Torre	1-5, 9-14, 18-23, 27.
VI	Orsini, Torre, Krawczyk	6, 15, 24.
VII	Orsini, Torre, Foster	7-8, 16-17, 25-26.

IV. Relevant Law

36. 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

37. 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.”

38. 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.

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

40. 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

41. 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.

42. 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.

43. 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.

44. 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.

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

46. 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

47. 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.

48. 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

49. 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 '802 Patent, *i.e.*, October 25, 2004.

A. General Computer Operations

50. 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.

51. 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.

52. 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.

53. 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.” EX1016 (Microsoft Computer Dictionary), 489.

54. 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.

55. 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.” EX1016 (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.” EX1016 (Microsoft Computer Dictionary), 469.

56. 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.” EX1016 (Microsoft Computer Dictionary), 212-213.

57. 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”. EX1016 (Microsoft Computer Dictionary), 552.

58. 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.” EX1016 (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.” EX1016 (Microsoft Computer Dictionary), 350.

59. Similarly, an “.avi” is “[t]he file extension that identifies an audiovisual interleaved data file in the Microsoft RIFF format.” EX1016 (Microsoft Computer Dictionary), 46. An “AVI” denotes the “[a]cronym for **A**udio **V**ideo **I**nterleaved. A Windows multimedia file format for sound and moving pictures that uses the

Microsoft RIFF (Resource Interchange File Format) specification.” EX1016 (Microsoft Computer Dictionary), 46.

B. Networking

60. 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.” EX1016 (Microsoft Computer Dictionary), 304, 315.

61. 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.” EX1016 (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.

62. 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).” EX1016 (Microsoft Computer Dictionary), 561, 566.

63. 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). EX1016 (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.”).

64. 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. EX1016 (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).”

65. 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.)

66. EX1016 (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.”).

67. A client/server architecture describes how client computers interact with server computers. EX1016 (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

68. 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. EX1016 (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.”).

69. 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.*” EX1016 (Microsoft Computer Dictionary), 357, 362.

70. 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.*” EX1016 (Microsoft Computer Dictionary), 463, 498-499.

71. 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.

72. 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.

73. 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 manifold.

74. 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." EX1016 (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.”).

75. 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.

76. 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.” EX1016 (Microsoft Computer Dictionary), 168.

77. 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.” EX1016 (Microsoft Computer Dictionary), 167.

78. “[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.” EX1016 (Microsoft Computer Dictionary), 168.

79. 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.” EX1016 (Microsoft Computer Dictionary), 168.

C. Data Security and Privacy

80. All modern systems include measures of security to protect their data from unauthorized access. EX1016 (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.”).

81. The POSIX standard, first started in the late 1980s, codifies the access application program interfaces (APIs) that operating systems export to user applications. EX1016 (Microsoft Computer Dictionary), 414 (“**POSIX** *n.* Acronym for **P**ortable **O**perating **S**ystem **I**nterface for **U**NIX. 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.”).

82. A simple data protection scheme defines whether users can only read files, only write files, or have both read and write access. EX1016 (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.”).

83. 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).

84. 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. EX1016 (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).

85. 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.” EX1016 (Microsoft Computer Dictionary), 142, 192.

86. 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.” EX1016 (Microsoft Computer Dictionary), 192.

87. 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.” EX1016 (Microsoft Computer Dictionary), 65.

88. “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).” EX1016 (Microsoft Computer Dictionary), 142, 153.

89. 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.” EX1016 (Microsoft Computer Dictionary), 192.

90. 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.” EX1016 (Microsoft Computer Dictionary), 458.

91. 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.” EX1016 (Microsoft Computer Dictionary), 422.

92. 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.” EX1016 (Microsoft Computer Dictionary), 422.

93. 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.” EX1016 (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.” EX1016 (Microsoft Computer Dictionary), 143.

94. 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.” EX1016 (Microsoft Computer Dictionary), 159.

95. 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.” EX1016 (Microsoft Computer Dictionary), 429.

96. 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.*” EX1016 (Microsoft Computer Dictionary), 159-160.

97. 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.*” EX1016 (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

98. 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.” EX1016 (Microsoft Computer Dictionary), 579.

1. Hash Functions and Their Uses

99. 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.

100. 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* EX1016 (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.”).

101. 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).

102. 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.

103. 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

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

105. “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....” EX1017 (Schneier), 70.

106. “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.” EX1017 (Schneier), 70.

107. Schneier describes a simple secret-splitting scheme using XOR among several people: Alice, Bob, Carol, Dave, and Trent. EX1017 (Schneier), 70-71. Schneier then outlines problems with this simple scheme.

108. “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." EX1017 (Schneier), 71.

109. 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." EX1017 (Schneier), 71.

110. "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." EX1017 (Schneier), 71.

111. "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." EX1017 (Schneier), 71.

112. “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**.” EX1017 (Schneier), 71.

113. “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.” EX1017 (Schneier), 71.

114. “General threshold schemes are even more versatile. Any sharing scenario you can imagine can be modeled.” EX1017 (Schneier), 72.

115. 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.” EX1017 (Schneier), 72.

D. Recovering from Data Loss

116. 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.

117. 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.

118. 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.

119. 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.

120. 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.

121. 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).” EX1016 (Microsoft Computer Dictionary), 437.

122. 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.

123. 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.

124. 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.

125. 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.

126. 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.

127. 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.

128. 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.

129. 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.

130. 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.

131. 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.

132. 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.

133. 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.” EX1016 (Microsoft Computer Dictionary), 196-197.

134. 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.

135. 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.

136. 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.

137. 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.” EX1019 (Birk) 13. Birk considers this problem in the context of “High Performance Video Servers”. EX1019 (Birk), 13.

138. Birk discusses techniques for “Exploitation of redundancy for load balancing”. EX1019 (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.” EX1019 (Birk), 16-17. Moreover, “[s]imilar observations were later made by Rabin [12] and dubbed ‘information dispersal algorithm’.” EX1019 (Birk), 17.

139. 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.” EX1019 (Birk), 17.

140. 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.” EX1019 (Birk), 17-18.

141. 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.” EX1020 (Chiquoine), [0004].

142. 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.” EX1020 (Chiquoine), [0005].

143. “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....” EX1020 (Chiquoine), [0010].

144. “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.” EX1020 (Chiquoine), [0038].

145. 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.

146. “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.” EX1022 (Kubiatowicz), 1 (Abstract).

147. “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.” EX1022 (Kubiatowicz), 8.

VI. The '802 Patent

148. The '802 Patent pertains to a “secure data parser” that “parses data and then splits the data into multiple portions that are stored or communicated distinctly.” EX1001 ('802 Patent), Abstract. In connection with its claims, Figure 21—which I copied and annotated below—“show[s] the steps of the process performed by the secure data parser on data to store the session master key with the parsed data.” EX1001 ('802 Patent), 55:22–25.

149. For instance, Figure 21 illustrates the following steps: (1) generating a “session master key” (highlighted in magenta); (2) encrypting the data set using the session master key (highlighted in lime green); (3) “[s]eparating the resulting encrypted data [set] into four shares or portions of parsed data” (highlighted in light blue); (4) likewise dividing the session master key into “key data” shares (highlighted in red); and (5) “appending the key data [shares] to the encrypted parsed data” (highlighted in orange and magenta) to produce final data shares. EX1001 ('802 Patent), 55:22–39. These data shares, containing both encrypted content and key information, are then further processed and stored in their designated storage units. *See* EX1001 ('802 Patent), 58:13–43, 58:55–62.

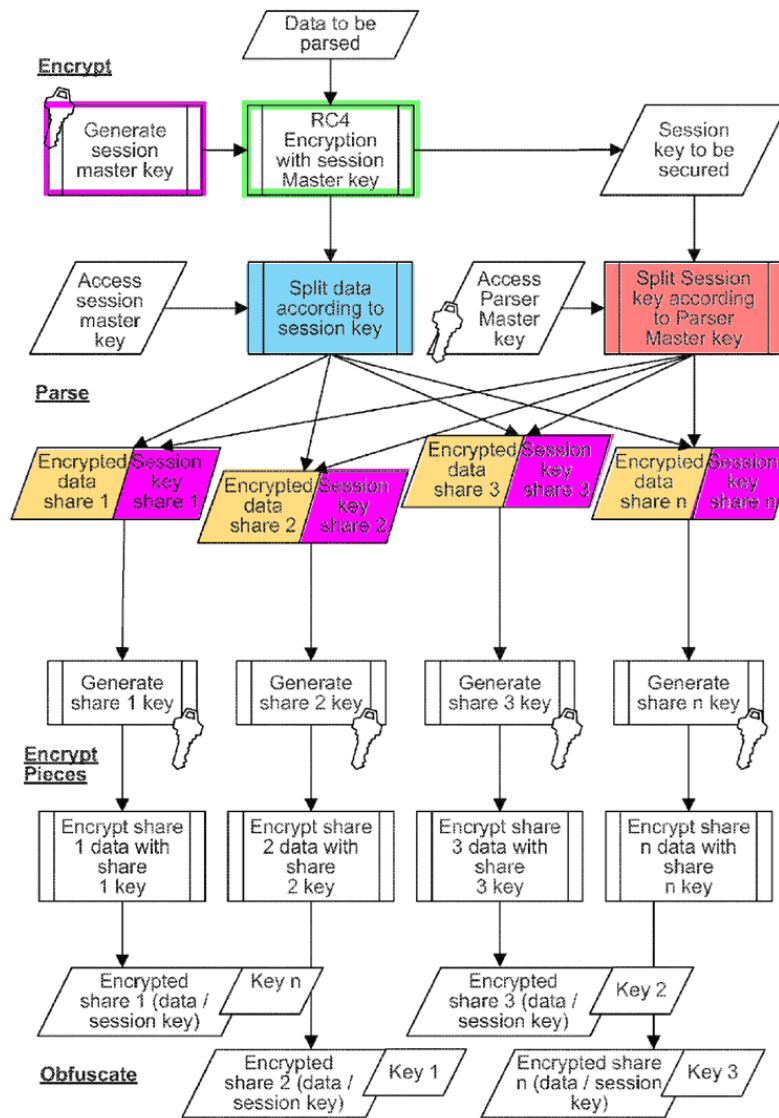


FIG. 21

EX1001 ('802 Patent), Figure 21 (annotated).

150. The '802 Patent describes the act of dividing data into shares as a “[a] cryptographic split (cryptosplit).” EX1001 ('802 Patent), 52:40–41. According to the patent, this cryptosplit technique is designed to “provide sufficient redundancy in the shares such that only a subset of the shares are needed to reassemble or restore the data to its original or useable form.” EX1001 ('802 Patent), 53:15–19. As one

illustrative embodiment, the patent notes that “each unit of data is stored in two shares, the primary share and the backup share,” ensuring that even if a share is missing later, the remaining shares can still “reassemble or restore the original data with no missing data units.” EX1001 (’802 Patent), 53:28–34. *See also* EX1001 (’802 Patent), 53:34–54:40.

151. The ’802 Patent further discloses that “[t]he secure data parser may include ... an error-checking component.” EX1001 (’802 Patent), 64:3–5. It explains that “[a]s portions of data are created using the secure data parser ..., to assure the integrity of the data within a portion, a hash value is taken at preset intervals within the portion and is appended to the end of the interval.” EX1001 (’802 Patent), 64:5–9. Additionally, the patent clarifies that “digital signature information” may serve as the “integrity information” used for this purpose. EX1001 (’802 Patent), 64:26–35.

A. Prosecution History

152. The ’802 Patent was filed as U.S. Application No. 13/371,361 (“’361 Application”), filed on February 10, 2012. EX1009 (’802 File History), 1; EX1001 (’802 Patent), Cover. The ’361 Application is a continuation of U.S. Application No. 11/258,839 (EX1010, “’839 Application”), which was filed on October 25, 2005. EX1001 (’802 Patent), Cover. The ’839 Application in turn claims the benefit

of two provisional applications: No. 60/622,146, filed October 25, 2004, and No. 60/718,185, filed September 16, 2005. EX1009 ('802 File History), 249; EX1010 ('839 App. File History), 240.

153. The claims of the '802 Patent were allowed without amendment following a single round of examination. On June 11, 2012, Examiner Smithers issued a Non-Final Office Action rejecting the claims as originally filed, citing U.S. Patent No. 7,391,865 as anticipating the claims. EX1009 ('802 File History), 324–328. In response, on June 29, 2012, the Applicant did not amend the claims but instead argued that the invention “improve[s] upon [U.S. Patent No. 7,391,865] by including, in a plurality of shares, data indicative of integrity information created from a data set.” EX1009 ('802 File History), 561. Thereafter, Examiner Smithers allowed the claims in their original form. EX1009 ('802 File History), 569–576.

154. While the '802 Patent's claims were being examined, Applicant also prosecuted a related application in the same family: U.S. Patent Application No. 13/371,363 (EX1011, “'363 Application”). The '363 Application originally included a claim 1 that was identical to claim 1 of the '802 Patent, except that the claim term “integrity information” was replaced with an example of such information—“hash information.” EX1011 ('363 App. File History), 34. But that claim of the '363 Application was rejected over prior art that Examiner Smithers did not analyze with respect to the '802 Patent's claims. *See* EX1011 ('363 App. File

History), 1167, 1205. Moreover, unlike the '802 Patent's claims that moved quickly and easily through the prosecution, the '363 Application was allowed only after seven rounds of Office Actions and with significant amendments. *See* EX1011 ('363 App. File History), 1167, 1205.

VII. Level of Ordinary Skill in the Art

155. In my opinion, a person of ordinary skill in the relevant field or art ("POSITA") as of the earliest claimed priority date of the '802 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.

156. I met and/or exceeded these requirements for a POSITA at the time of the earliest claimed priority date of the '802 Patent.

VIII. Claim Construction

157. I have reviewed the claims, specification, and the file history of the '802 Patent. EX1001 ('802 Patent), EX1009 ('802 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. EX1010 ('839 App. File History), EX1014 (Provisional Application No. 60/622,146), EX1015 (Provisional Application No. 60/718,185).

158. 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.

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

IX. Specific Grounds For Challenge

A. Ground I: Torre And Tajima Render Obvious Claims 1-3, 5, 9, 10-12, 14, 18, 19-21, 23, 27.

160. Claims 1-3, 5, 9, 10-12, 14, 18, 19-21, 23, 27 would have been obvious in view of Torre and Tajima.

161. In this Ground, I introduce illustrations to show how the Torre/Tajima combination renders Claims 1-3, 5, 9, 10-12, 14, 18, 19-21, 23, 27 obvious.

1. Torre/Tajima Color-Coding Scheme

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

Color	Disclosures in Torre and/or Tajima
violet	Torre's SHA-1 digital signature information
pink	Torre's encryption operation
lime green	Torre's encrypted data set
	Tajima's encrypted data
royal blue	Torre: selection information for selecting between allocation modes
	Tajima: arbitrary length information
red	Torre's shreds
light blue	Torre's intermediate data blocks

Color	Disclosures in Torre and/or Tajima
	Tajima's data fragments
magenta	Tajima's encryption key
olive	Torre's storage servers
brown	Torre's data unit from data blocks
yellow	Torre's allocator
	Tajima's allocation method
forest green	Collection of the encrypted data, SHA-1 digital signature information, and/or encryption key
purple	A tag including SHA-1 digital signature information, and encryption key

2. Overview of Torre (EX1003)

163. Torre is highly similar to the '802 Patent. Torre describes “a system and method for shredding and deshredding [data] ... to store and retrieve [the] data while emphasizing considerations such as redundancy, integrity, and security.” EX1003 (Torre), [0049]. Torre explains that the input data is transformed “into smaller pieces of data called shreds” (EX1003 (Torre), [0051]) and “writing the shredded input data ... [in] multiple storage units” (EX1003 (Torre), [0052]). In addition, “[s]hredding is done in a manner that adds redundancy to the shreds so that the input data can be recovered from a fewer number of shreds than the total number stored.” EX1003 (Torre), [0051].

164. The system's “shredder” processes input data through a series of transformation and processing steps, as illustrated in Figure 4 below. EX1003 (Torre), [0063] (“As mentioned, numerous passes of shredding can be implemented

such as the YES branch of decision step 126 in FIGS. 3 and 4. An implementation using numerous passes of shredding is found in FIG. 5 showing a multilevel shredding approach. Input data 139 is sent to a shredder pool 140, which contains a first shredder 142 that receives the input data. The shredder 142 is coupled by an interconnect 144 to a plurality of additional shredders 146 in the shredder pool 140 that successively shred the shredded data outputted by the first shredder 142 based upon their hierarchical position in the levels of multilevel shredders. The levels of additional multilevel shredders 146 are also coupled by interconnects 144.”). *See also* EX1003 (Torre), [0055]-[0057].

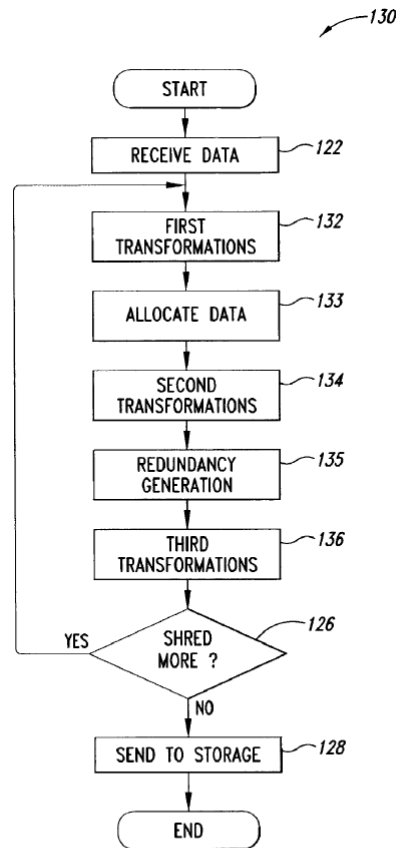


Fig. 4

EX1003 (Torre), Figure 4.

165. When input data is received (step 122), the shredder “performs first transformations (step 132),” which “could include one or more of compression, encryption, or signature generation functions.” EX1003 (Torre), [0061]. These initial transformations prepare the data for further processing.

166. After the first transformations, the system “allocates data transformed by the first transformations into various different blocks of data (step 133)” referred

to as “pre-shred[s]” (EX1003 (Torre), [0061], [0080]) by “split[ting] up the [transformed] data.” EX1003 (Torre), [0062] (“Allocation (step 133) is not necessarily found in all implementations. Those implementations that use allocation (step 133) split up the data transformed by the first transformations (step 132) to be further processed in parallel by multiple processors or sequentially by one processor with the second transformations (step 134), the redundancy generation (step 135), and the third transformations (step 136).”).

167. Torre then applies a second transformation (step 134), which “further splits the units of data into smaller sized units of data and generates additional redundant data (step 135) so that fewer pieces of stored data are necessary to restore the stored input data.” EX1003 (Torre), [0061]. Finally, the shredder performs a third transformation (step 136) to complete the shredding process. EX1003 (Torre), [0061]-[0062] (“The third transformations (step 136) can include functions to partition the shredded data based upon the number of storage units available. For instance, if the shredded data has eight shreds, but there are only seven units of storage available, either more than one of the shreds may be placed on one of the units of storage or since the shreds are redundant, one of the shreds may simply be dropped. The third transformations (step 136) can also include signature generation to further improve upon data integrity.”).

168. The shredder may consist of multiple processing components arranged in tiers. For example, the shredder architecture shown in Figure 7 below includes several distinct elements. First, a first-tier component carries out the first transformation step 132 and the allocation step 133 using a “first transformer 174” and an “allocator 176,” respectively. Second, multiple second-tier components each take in a pre-shred from the first tier and perform the second transformation step 134 and the redundancy generation step 135, using a “second transformer 178” and a “redundancy generator 180,” respectively. Finally, third-tier components perform the third transformation step 136. EX1003 (Torre), [0073]-[0074].

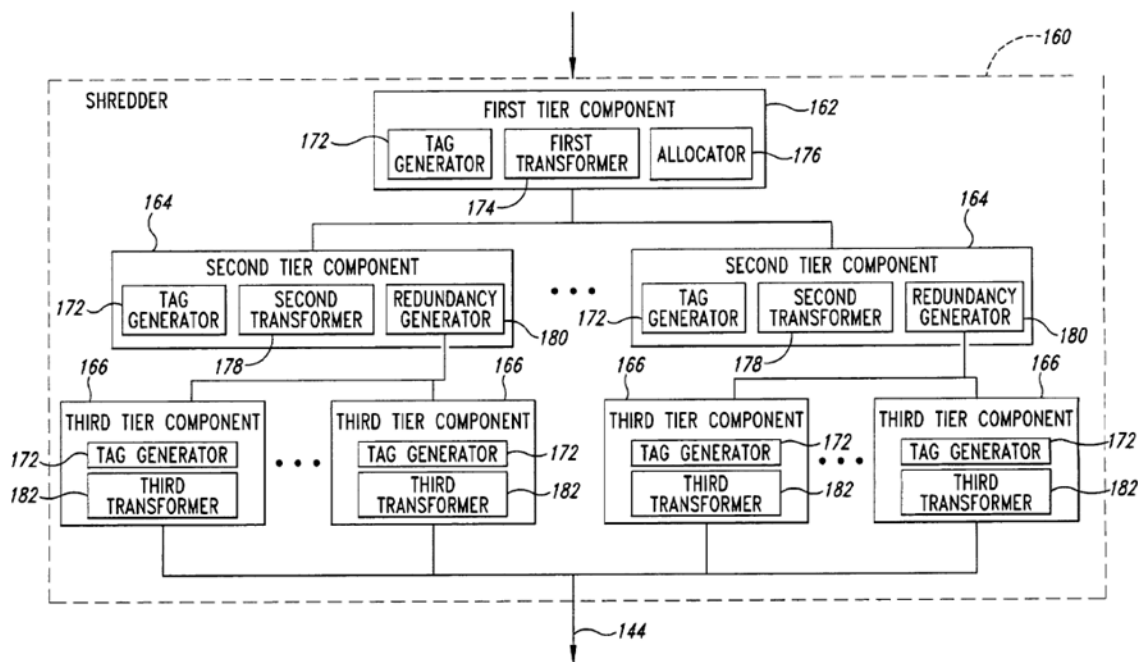


Fig. 7

EX1003 (Torre), Figure 7.

169. As input data moves through Torre's various processing steps of the shredder, the shredder attaches "tags" to the processed data "to keep track of what is been done to the pieces of the input data 139 as the pieces of the input data move through the shredder." EX1003 (Torre), [0078].

170. These tags contain metadata that describes the transformations applied to each piece of data. For example, "the tags 110 can contain information about its associated piece of the input data 139 describing such details as data geometries, data relationships, data nesting, structure, coding structure, data alignment, other data descriptions including the size and length, and data qualities such as hash." EX1003 (Torre), [0078]. Torre explains that "[i]nformation [in the tags] describing how the shreds were created is used to reassemble the shreds through processes inverse to those used to create the shreds." EX1003 (Torre), [0053].

171. Torre provides one implementation of a shredder in Figure 34 below that illustrates specific processing steps. In this configuration: (1) the shredder's first transformer 174 "uses a SHA-1 signature generation function to generate a signature associated with the received data to generate a pre-shred 466 containing the received data and a tag of the SHA-1 signature" (EX1003 (Torre), [0131]); (2) the shredder's allocator 176 "splits the pre-shred 466 into three data blocks 494, which are sent on to three redundancy generators [180/]468 each located in three of

the second-tier components 164" (EX1003 (Torre), [0135]); and (3) each of the redundancy generators 180 generate six redundant pre-shreds 470.

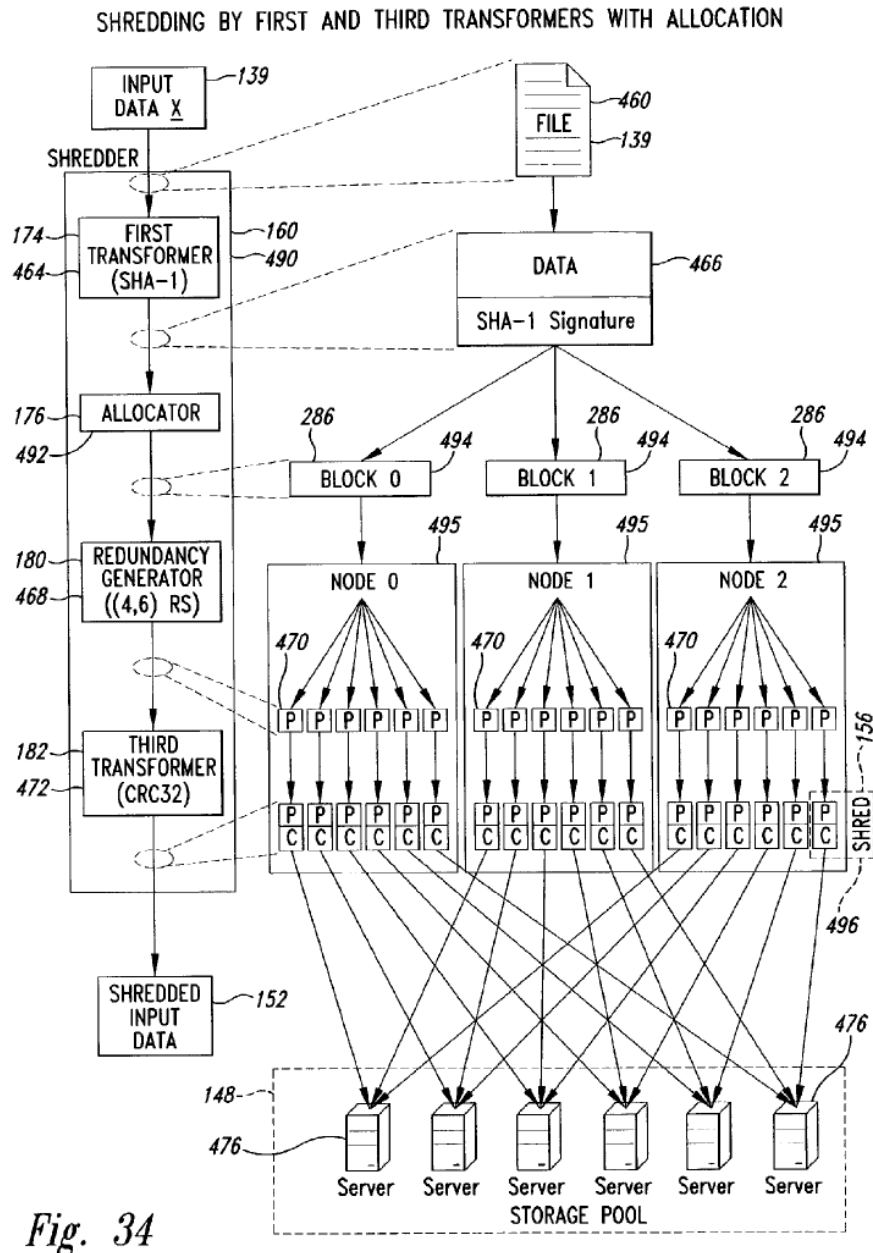


Fig. 34

EX1003 (Torre), Figure 34.

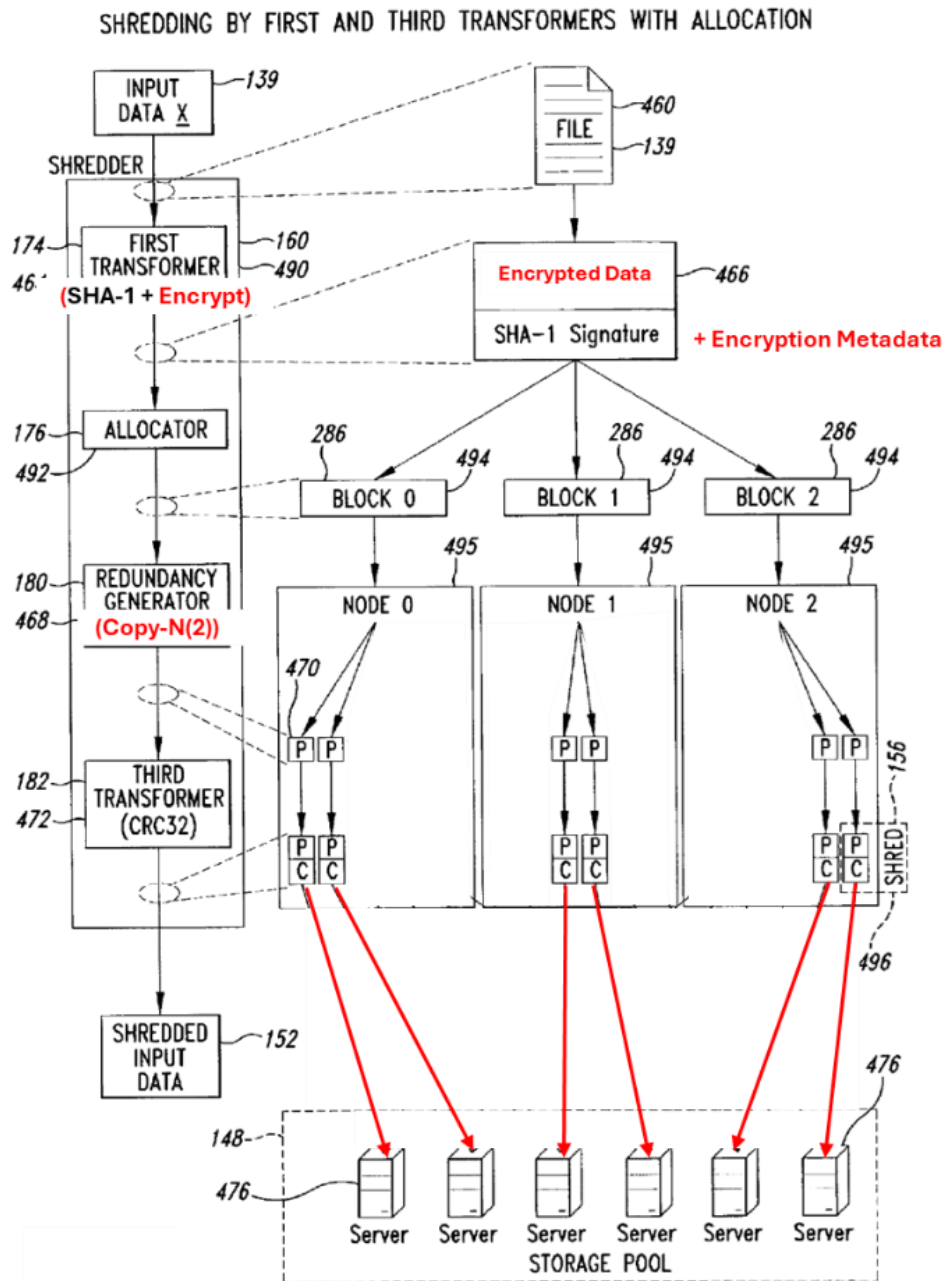
172. Torre indicates that both the first transformer and the redundancy generator can be implemented using different techniques. For instance, the first transformer may apply encryption after generating a digital signature and append a tag that reflects both the encryption and signature generation steps. EX1003 (Torre), Figure 11 (Rule 9), [0091] (“For instance for the rule 40 line, the first transformer 174 first performs a compression function, and the first transformer performs the allocation function, then the second transformer 178 performs an encryption function, and then the second transformer performs a signature generation function.”). *See also* EX1003 (Torre), [0092] (“Consequently any of the entries in the table of FIG. 11 that had *signature generation* alone or *before* compression, allocation, or *encryption* would be a candidate for high data integrity.”).¹ In addition, the redundancy generator may implement “[a] redundancy transformation called *Copy-N* ... to simply make multiple exact copies of existing data.” EX1003 (Torre), [0111], [0113].

173. Torre expressly describes and teaches that variations in the implementations of the first transformer and the redundancy generator are applicable to the embodiment shown in Figure 34. *See* EX1003 (Torre), [0147] (“From the foregoing it will be appreciated that, although specific embodiments of the invention

¹ In this Declaration, all emphases added are italicized bold or bold text unless otherwise noted.

have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention.”).

174. The following figure (annotated below) illustrates these variations as applied to Figure 34. This variation is referred to herein as the “Torre System.” The Torre System incorporates the features of Figure 34, but is updated (as shown in red) so that the first transformer performs encryption after digital signature generation and appends a tag for both the encryption and digital signature generation steps and the redundancy generator uses the Copy-N redundancy scheme as described in the preceding paragraph. EX1003 (Torre), [0111], [0113].



EX1003, Figure 34 (annotations in red added).

3. Overview of Tajima (EX1004)

175. Tajima describes “a personal information management method ... that can improve the security of personal information that is registered with areas that

are connectible to the Internet.” EX1004 (Tajima), [0044]. *See also* EX1004 (Tajima), [0048] (“Thus, because personal information that is registered with areas that are connectible to the Internet is divided into a plurality of data portions and then registered with areas that are each under separate control, the personal information cannot be viewed unless all of the areas in which data are registered are exposed, and an improvement can therefore be obtained in the security of personal information that is registered with areas that are connectible to the Internet.”).

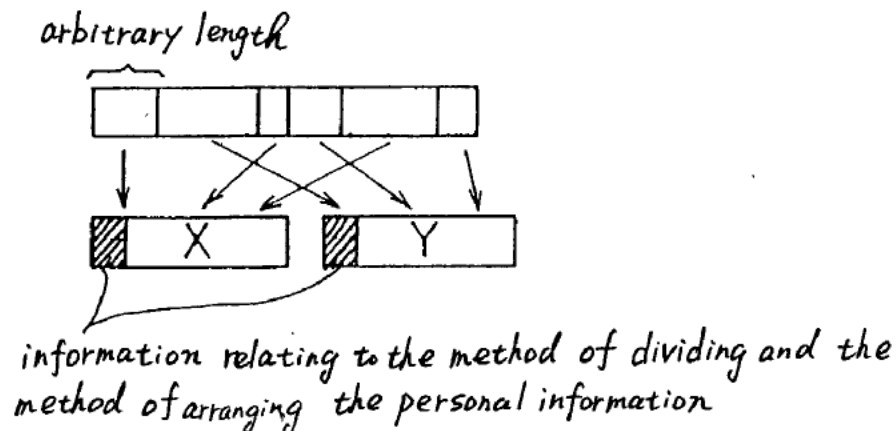
176. In particular, Tajima discloses “divid[ing] [] personal information into a plurality of data portions, register[ing] at least one of the plurality of data portions with [one server]..., and register[ing] the other data portions with other areas [*e.g.*, another server] that are connectible to the Internet and that are under [separate] control.” EX1004 (Tajima), [0046]. Because the information is distributed and stored across multiple servers under distinct administrative control, Tajima explains that “an improvement can therefore be obtained in the security of personal information.” EX1004 (Tajima), [0048].

177. Tajima describes multiple “method[s] of dividing personal information,” one of which is shown in Figure 10 below. EX1004 (Tajima), [0100], [0106]. Tajima teaches that “personal information ... is first divided into a plurality of data portions each of different arbitrary length according to a function of, for example, random numbers, time, or file capacity, and this plurality of data portions

is then arranged into two data fragments, data fragment X and data fragment Y, according to a set method of arrangement, as shown in FIG. 10.” EX1004 (Tajima), [0106].

178. Tajima further explains that “[b]ecause the personal information is divided into data portions of different arbitrary lengths in this case, the arbitrary lengths, which is information relating to the method of dividing and method of arranging, must be attached to each of data fragments X and Y.” EX1004 (Tajima), [0107]. The data fragments X and Y, along with the information on “the arbitrary lengths,” are stored separately on two different servers to strengthen data security. EX1004 (Tajima), [0107].

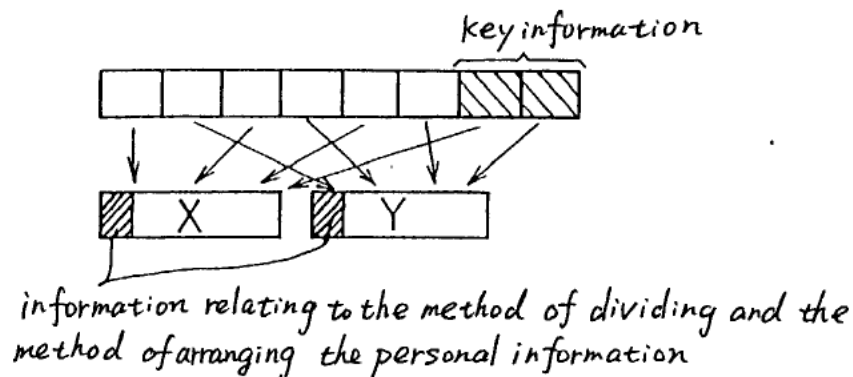
Fig. 10



EX1004 (Tajima), Figure 10.

179. Tajima additionally clarifies that the technique described in Figure 10 may be applied to protect encrypted data as well. For instance, Figure 11 below depicts a variation in which “personal information ... is first encrypted, and the encrypted personal information and information regarding the key that is used in the encryption are then divided into a plurality of data portions each of ... arbitrary length as shown in ... FIG. 10.” EX1004 (Tajima), [0109].

Fig. 11



EX1004 (Tajima), Figure 11.

180. Claim 8 of Tajima further explains that “information regarding the key” may be the “encrypted key” used to encrypt the personal information. EX1004 (Tajima), Claim 8. Tajima also teaches that “the plurality of data portions are then arranged into two data fragments, data fragment X and data fragment Y, according to a set method of arrangement.” EX1004 (Tajima), [0109].

4. Combining Torre With Tajima (“Torre/Tajima”)

181. A POSITA would have been motivated to combine the Torre System with Tajima to implement Tajima’s teachings in the Torre System.

182. For example, a POSITA would have found it obvious to combine the Torre System with Tajima’s teaching to include the encryption key as one of the tags appended to the encrypted data, referred below as the “Torre/Tajima System-1.” *See* §IX.A.4.i below.

183. A POSITA would have also found it obvious to further combine the Torre/Tajima System-1 with Tajima’s teaching to generate data splitting information and to use it to split encrypted data into data portions, referred below as the “Torre/Tajima System-2.” *See* §IX.A.4.ii below.

184. Next I describe the combinations referred to as the Torre/Tajima System-1 and the Torre/Tajima System-2, and I explain why a POSITA would have found both the Torre/Tajima System-1 and the Torre/Tajima System-2 obvious.

i. Torre System With Tag That Includes Encryption Key As Taught By Tajima (“Torre/Tajima System-1”)

185. Torre describes a process in which the first transformer of a shredder applies a transformation to input data and generates a tag identifying the operation performed. This tag is then attached to the transformed data to produce a pre-shred. EX1003 (Torre), [0102] (“The first transformer 174 processes the input data 139 and

outputs a pre-shred 270 containing a tag 272 describing the first transformer processing and transformed data 274 resulting from the first transformer processing.”); EX1003 (Torre), [0078] (“The tags 110 help keep track of what is been done to the pieces of the input data 139 as the pieces of the input data move through the shredder from the first-tier component 162 to the second-tier components 164 and to the third tier components 166. The tags 110 are used to indicate which functions should be performed in various stages of reconstructing the input data 139 from the shredded input data 132.”).

186. Torre further explains that this tag is later referenced by the de-shredder to reverse the transformation and reassemble the original input data. *See* EX1003 (Torre), [0078] (“The tags [] are used to indicate which functions should be performed in various stages of reconstructing the input data [] from the shredded input data [].”). “For instance, if [] encryption [] was done by [the first] transformer of the shredder 160, then [] decryption [] would be performed by the inverse transformer of the deshredder 390 corresponding to the [first] transformer of the shredder.” EX1003 (Torre), [0123].

187. Figure 14 of Torre (annotated below) illustrates the process of adding a tag to the transformed data. The light green box shows a data transformed by the first transformer 174; the purple box shows the tag appended to the transformed data; the green box shows the preshred 270 that includes the transformed data and the tag;

the yellow box shows the allocator that receives the preshred 270; and the blue boxes show the data portions (or preshreds) created by the allocator 176.

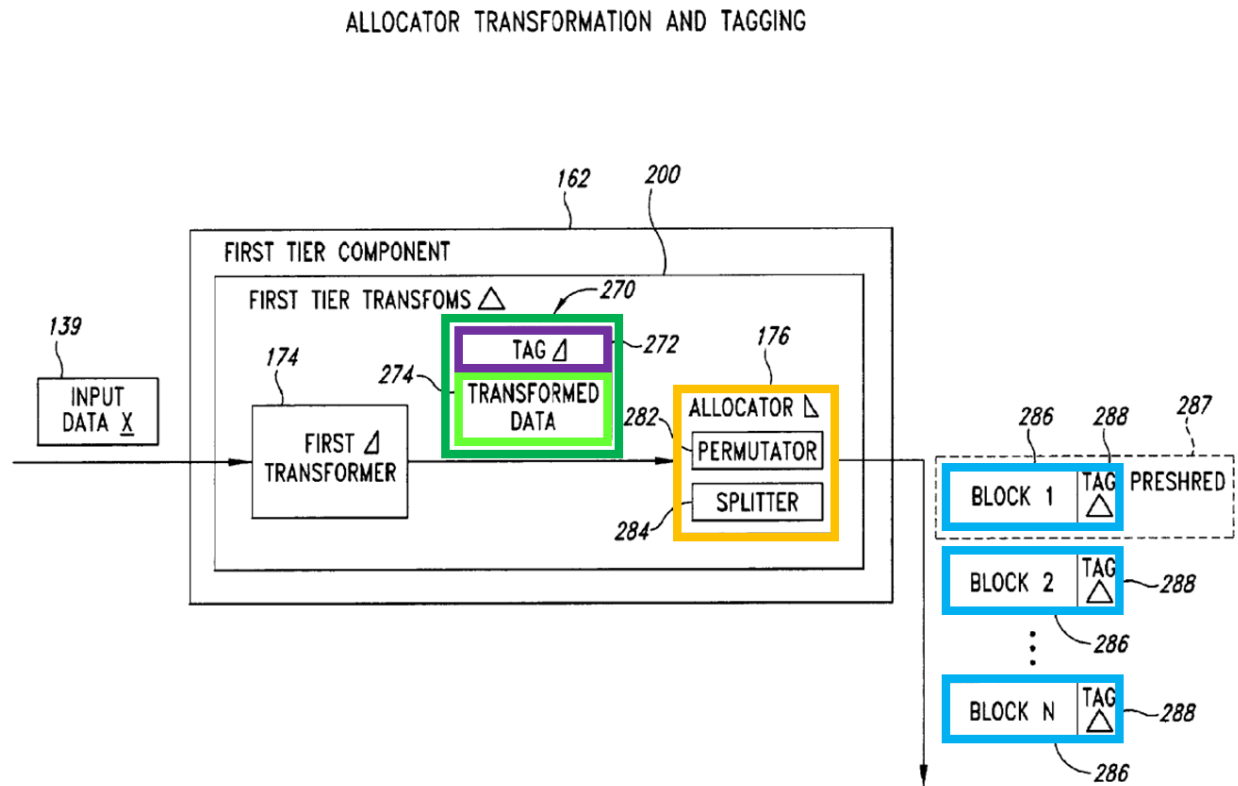


Fig. 14

EX1003 (Torre), Figure 14 (annotated).

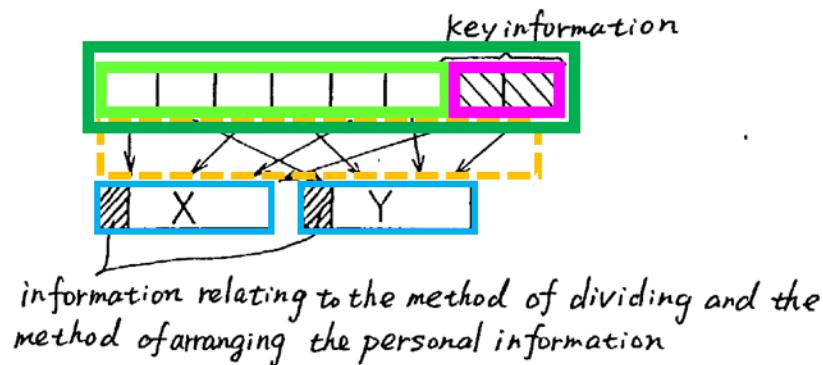
188. When the first transformer performs encryption on input data, the associated tag would include an “identifier[]” of the specific encryption function used. *See* EX1003 (Torre), [0079] (“Each of the tags 110 can also contain identifiers of the particular functions performed by the shredder 160 on the associated data piece. ... For example, an encryption function may be used as a certain type of cipher and the cipher block chaining would be a mode of the encryption function.”). A

POSITA would have recognized that such a tag—which identifies the encryption function—includes an encryption key. Without the encryption key, it would not have been feasible to reverse the encryption process during de-shredding, as Torre expressly teaches. EX1003 (Torre), [0123]. *See also* §V.C above.

189. It would have been, at a minimum, obvious to modify Torre to include the encryption key within the tag, given that doing so was well known in the art. For instance, Tajima describes an approach where input data is encrypted using an encryption key; the encrypted data is then appended with the encryption key; and the combined set is divided into data fragments. EX1004 (Tajima), [0109]; [0110] (“the key information that was attached to the encrypted personal information”); Claim 8 (“encrypts said personal information and divides the encrypted personal information and the encryption key used in the encryption into a plurality of data portions” that form data fragments).

190. Figure 11 of Tajima (annotated below) illustrates such an approach. In the annotated Figure 11 below, the lime green box shows the encrypted data; the pink box shows the key information (*i.e.*, encryption key); the dark green box shows the collection of the encrypted data and the key information; the dotted yellow box shows the process of assigning the collection into data portions shown in blue. *See* §IX.A.1 above.

Fig. 11



EX1004 (Tajima), Figure 11 (annotated).

191. Combining the teachings of Torre and Tajima would have been obvious. For instance, the first transformer of Torre, when executing an encryption function, can append a tag that includes the encryption key, as taught by Tajima.

192. A visual depiction of the combination of Torre and Tajima ("Torre/Tajima") is shown and annotated below using the same color-coding scheme as that described in §IX.A.1. In particular, Tajima's encryption key is shown in the Torre's Figures 14 and 34 in pink. This combination is referred to as the "Torre/Tajima System-1."

Torre/Tajima System-1

ALLOCATOR TRANSFORMATION AND TAGGING

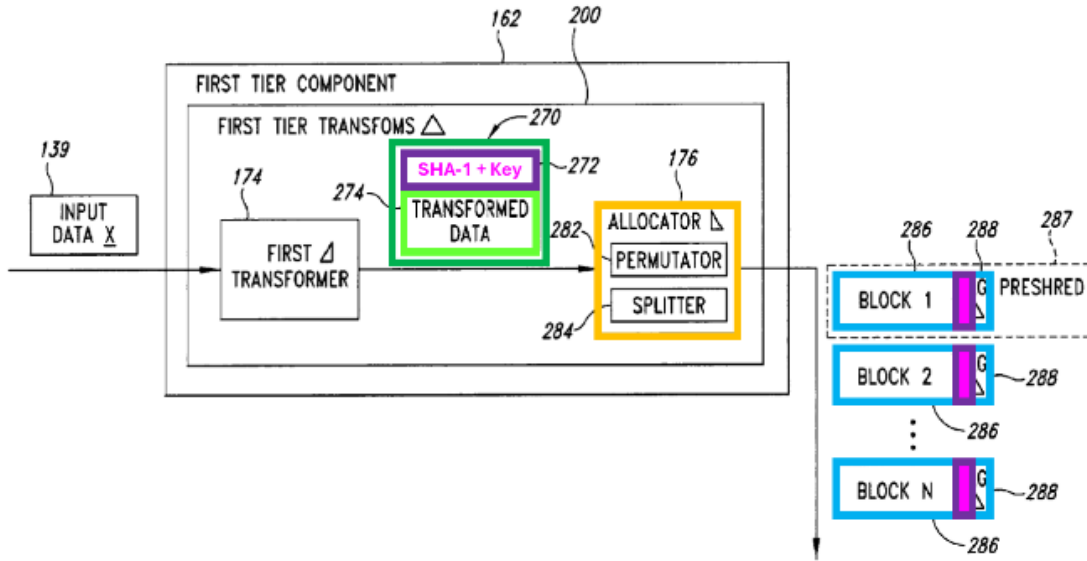
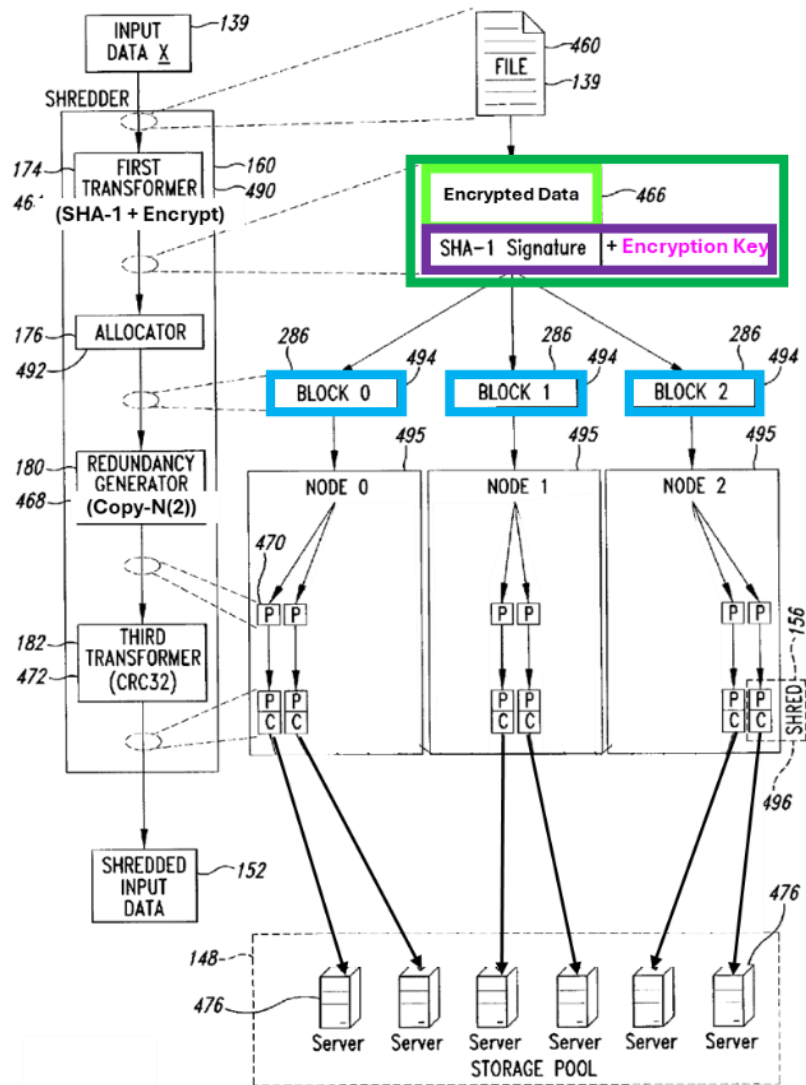


Fig. 14

EX1003 (Torre), Figure 14 (annotated).

Torre/Tajima System-1

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), Figure 34 (annotated).

**a. Reasons to Combine Torre And Tajima As
Provided In Torre/Tajima System-1**

193. A POSITA would have been motivated to combine Torre and Tajima as reflected in the Torre/Tajima System-1 figures above for at least the following three reasons.

194. **First**, both Torre and Tajima are aimed at addressing a closely related problem—securely storing data across multiple storage resources—using similar solutions. *Compare* EX1003 (Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and scalability and adaptability to resources.*”) with EX1004 (Tajima), [0044] (“It is an object of the present invention to provide a personal information management system, a personal information management method, and a *server that can improve the security of personal information that is registered with areas that are connectible to the Internet.*”), [0048] (“Thus, because *personal information that is registered with areas that are connectible to the Internet is divided into a plurality of data portions and then registered with areas that are each under separate control, ...*”).

195. Specifically, each reference teaches a method for enhancing data security by applying transformations and distributing the resulting data across distinct locations. *Compare* EX1003 (Torre), [0051]–[0052] (“In the process of

shredding, input data is generally *transformed into many smaller pieces of data called shreds*. ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of data (step 134).*”) with EX1004 (Tajima), [0109] (“As still another example of a method of dividing personal information, as shown in FIG. 11, personal information that has been submitted to certificate authority 40a by the user of user terminal 10 is first encrypted, and *the encrypted personal information and information regarding the key that is used in the encryption are then divided into a plurality of data portions each of fixed length as shown in FIG. 8, or of arbitrary length as shown in FIG. 9 or FIG. 10.* The plurality of data portions are then arranged into two data fragments, data fragment X and data fragment Y, according to a set method of arrangement, and *data*

fragment X is then registered with database 41a of certificate authority 40a and data fragment Y is registered with database 41b of certificate authority 40b.”).

196. For example, both Torre and Tajima teach transforming the data through, for example, encryption. *Compare* EX1003 (Torre), [0061] (“After receiving the input data (step 122), the method goes on to perform the shredding step 124. The shredding step 124 performs first transformations (step 132).... The first transformations could include one or more of compression, **encryption**, or signature generation functions.”) *with* EX1004 (Tajima), [0109] (“personal information that has been submitted to certificate authority 40a by the user of user terminal 10 is first **encrypted**.”).

197. Both Torre and Tajima also teach breaking up the transformed data into multiple portions while reordering or remapping the transformed data. *Compare* EX1003 (Torre), [0061]-[0062] (“The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into **various different blocks of data** (step 133), and performs second transformations on these blocks of data (step 134).”) *with* EX1004 (Tajima), [0109] (“the encrypted personal information and information regarding the key that is used in the encryption are then **divided into a plurality of data portions**.”).

198. Both Torre and Tajima further teach storing the multiple portions in different storage devices to improve the security. *Compare* EX1003 (Torre),

[0051]–[0052] (“In the process of shredding, input data is generally *transformed into many smaller pieces of data called shreds*. ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of data (step 134).*”) with EX1004 (Tajima), [0109] (“As still another example of a method of dividing personal information, as shown in FIG. 11, personal information that has been submitted to certificate authority 40a by the user of user terminal 10 is first encrypted, and *the encrypted personal information and information regarding the key that is used in the encryption are then divided into a plurality of data portions each of fixed length as shown in FIG. 8, or of arbitrary length as shown in FIG. 9 or FIG. 10*. The plurality of data portions are then arranged into two data fragments, data fragment X and data fragment Y, according to a set method of arrangement, and *data fragment X is then registered with database 41a of*

certificate authority 40a and data fragment Y is registered with database 41b of certificate authority 40b.”).

199. Given these parallels, a POSITA would have recognized that Tajima’s methods could be readily adapted within Torre’s shredder architecture.

200. **Second**, Torre explains that the tag produced by the first transformer is later used by the de-shredder to reverse the prior transformations and reconstruct the original data. *See* EX1003 (Torre), [0078] (“The tags 110 help keep track of what is been done to the pieces of the input data 139 as the pieces of the input data move through the shredder from the first-tier component 162 to the second-tier components 164 and to the third tier components 166. ***The tags 110 are used to indicate which functions should be performed in various stages of reconstructing the input data 139 from the shredded input data 132.*** Each of the tags 110 can contain information about its associated piece of the input data 139 describing such details as data geometries, data relationships, data nesting, structure, coding structure, data alignment, other data descriptions including the size and length, and data qualities such as hash.”), [0123] (“***Each of the tag readers 398 gets information from received tags to determine for the inverse transformer of the particular tiered component containing the particular tag reader what inverse transformations should be performed by the inverse transformer on shredded data at a point in the deshredding process associated with the particular tiered component.*** For

instance, if compression, encryption, or signature generation was done by a particular transformer of the shredder 160, then decompression, decryption, or signature verification would be performed by the inverse transformer of the deshredder 390 corresponding to the particular transformer of the shredder.”).

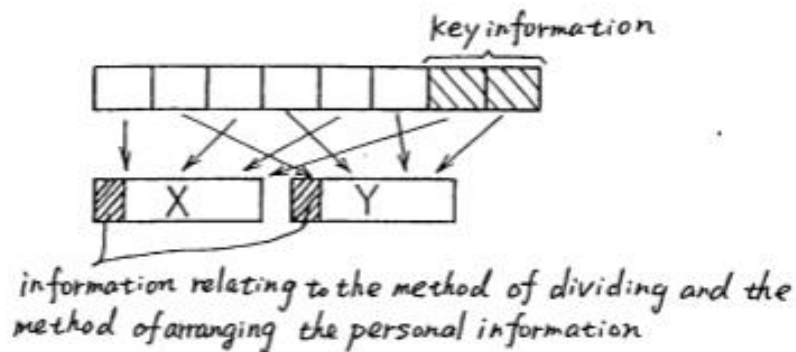
201. In scenarios where the first transformer applies encryption using a key for “the Data Encryption Standard (DES)” (EX1003 (Torre), [0085]), for example, a POSITA would have recognized that the same key is necessary for decryption, as DES is a symmetric encryption algorithm. *See* EX1023 (Downs), 12:44-58 (“In the Secure Digital Content Electronic Distribution System 100 the Content Provider(s) 101 encrypts the content using symmetric algorithms. They are called symmetric algorithms because the same key is used to encrypt and decrypt data. The data sender and the message recipient must share the key. The shared key is referred to here as the symmetric key. The Secure Digital Content Electronic Distribution System 100 architecture is independent of the specific symmetric algorithm selected for a particular implementation. Common symmetric algorithms are DES, RC2 and RC4. Both DES and RC2 are block cipher [sic]. A block cipher encrypts the data using a block of data bits at a time. DES is an official US government encryption standard, has a 64-bit block size, and uses a 56-bit key.”). *See also* §V.C above.

202. As a result, a POSITA would have been motivated to add the encryption key to the tag, enabling the de-shredder to perform the corresponding decryption.

This modification would have been especially apparent in light of Tajima's teaching of appending the encryption key to the encrypted data—it would have merely amounted to using a known technique (Tajima's technique for appending the encryption key to the encrypted data) to improve a similar device (Torre System) in the same way of providing the encryption key as a tag appended to the encrypted data so that the deshredder can use it to decrypt the encrypted data.

203. **Third**, Torre states that a tag “can be represented as metadata” and “can include the transformation steps taken, [and] input parameters used.” EX1003 (Torre), [0056]. Given this disclosure, a POSITA would have been inclined to include encryption-related input parameters—such as an encryption key—within the tag when the first transformer performs an encryption operation. Tajima provides a concrete example of this practice by teaching the inclusion of the encryption key as part of the metadata appended to the encrypted data. *See* EX1004 (Tajima), Figure 11, [0109]. Accordingly, a POSITA would have been motivated by Torre's express teaching to include input parameters of an encryption function in the tag of the pre-shred, such as the encryption key taught by Tajima.

Fig. 11



EX1004 (Tajima), Figure 11.

204. Next, a POSITA would have had a reasonable expectation that incorporating an encryption key into the tag generated by Torre's first transformer would function as intended for at least the following three reasons.

205. **First**, Torre describes the tag as carrying information necessary to reconstruct the original data, including instructions for reversing earlier transformations. Specifically, the tag is used to "indicate which functions should be performed in [the first transformer stage] of reconstructing the input data [] from the shredded input data []." EX1003 (Torre), [0078]. Therefore, when encryption is applied, the tag would reflect how that encryption was performed—so the corresponding decryption can occur during deshredding. EX1003 (Torre), [0078]; [0123].

206. Indeed, Tajima teaches the same principle: Tajima teaches appending the encryption key to the encrypted data, allowing it to be used later to “decrypt the encrypted [data],” which serves the same purpose as Torre’s tag. EX1004 (Tajima), [0110] (“the key information that was attached to the encrypted personal information is then used to decrypt the encrypted personal information.”). Thus, the operation of the combination would have been predictable.

207. **Second**, Torre states that “the embodiments disclosed herein ... can be equivalently implemented ... as one or more computer programs running on one or more computers (e.g., as one or more programs running on one or more data processing systems).” EX1003 (Torre), [0141]. Therefore, adapting the Torre System to incorporate Tajima’s teachings would simply involve implementing the corresponding functionality in software, by writing program code. That task would have been well within the abilities of a POSITA. *See* EX1003 (Torre), [0141] (“designing the circuitry and/or writing the code for the software and or firmware [described by the embodiments] would be well within the skill of one of ordinary skill in the art in light of this disclosure”).

208. **Third**, modifying Torre with Tajima’s tag generated by Torre’s first transformer would have been simple and would not have required extensive experimentation or impose undue burden, because both already disclose tags that carry transformation or encryption-related information. For the same reasons, the

combination would have yielded predictable results (because using such tags to guide data reconstruction or decryption is a well-established practice). Lastly, a POSITA would have had a reasonable expectation of success because Tajima does nothing more than provide additional detail on the same tag functionality already described by Torre.

ii. Torre/Tajima System-1 In View Of Tajima's Data Remapping And Splitting Method ("Torre/Tajima System-2")

209. Torre's shredder design includes an allocator responsible for dividing the incoming data and breaking it into separate data blocks. EX1003 (Torre), [0099] ("An exemplary allocator method 260 is shown in FIG. 13 as receiving data from the first transformer 174 (step 262) and partitioning the received data (step 264) to be sent to either multiple processors to process the partitioned data in parallel or to a single processor with limited resources to process the partitioned data sequentially."), [0102]. As depicted in Figures 15 and 17 below, this allocator features a "permutator 282 [that] re-maps the ordering of the transformed [received] data 274 using" (EX1003 (Torre), [0102]) techniques such as "chunking [and] interleaving." EX1003 (Torre), [0100]. Additionally, the allocator incorporates a "splitter 284 [that] splits the re-mapped data into data blocks 286 that are part of pre-shreds 287 outputted by the first-tier component 162." EX1003 (Torre), [0102]. Torre/Tajima System-1 would use these partitioning mechanisms from Torre—such

as chunking and interleaving—to divide encrypted data into multiple blocks as part of the shredding process.

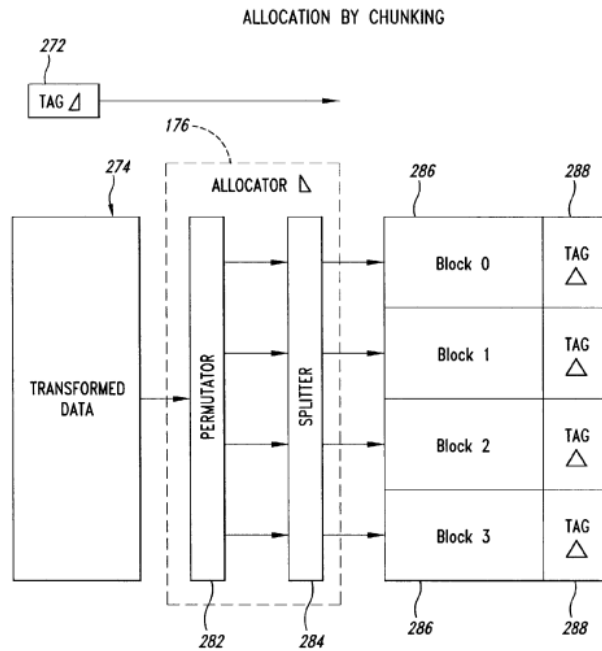


Fig. 15

EX1003 (Torre), Figure 15.

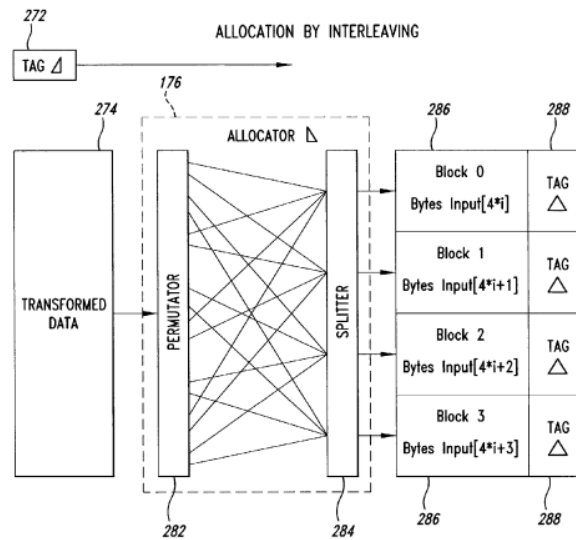


Fig. 17

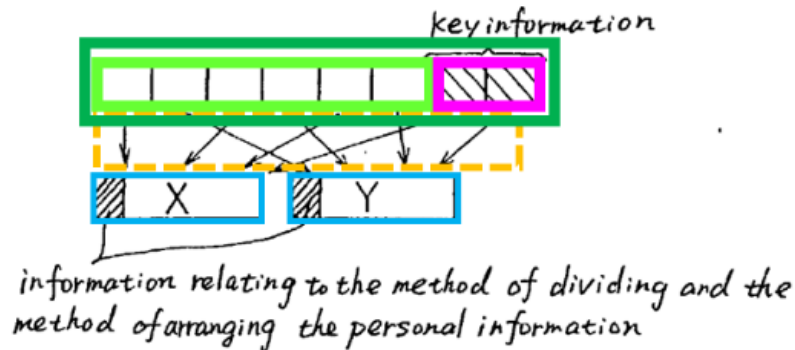
EX1003 (Torre), Figure 17.

210. Torre further notes that the allocator’s remapping of incoming data can be performed using various techniques, “dependent upon particular situations.” See EX1003 (Torre), [0101] (“the general scheme of remapping is not limited and specific remapping is *dependent upon particular situations*”). Approaches for remapping data in this way were well established in the field at the time.

211. For example, Tajima provides an illustrative example, shown in connection with Figure 11 (annotated with the color-coding scheme described in §IX.A.1 above), that involves: (1) acquiring input to be protected, including the encrypted content and the “encryption key used in the encryption”; (2) dividing that input into “a plurality of data portions each of ... arbitrary length as shown in ...

FIG. 10”; and (3) separating the resulting data portions into multiple fragments. EX1004 (Tajima), [0109]; [0156] (“it is also possible for the personal information to be divided among three or more data fragments”); Claim 8 (“... divides the encrypted personal information and the encryption key used in the encryption into a plurality of data portions”). See §IX.A.3 above [Tajima].

Fig. 11



EX1004 (Tajima), Figure 11 (annotated).

212. It would have been obvious to adapt the allocator in Torre/Tajima System-1 using the division and arrangement approach disclosed in Tajima’s Figure 11. In particular, implementing allocator 176 to apply Tajima’s method of segmenting and organizing received data into data fragments—corresponding to the pre-shreds 287 in Torre/Tajima System-1—would have been a natural and straightforward modification. EX1004 (Tajima), [0109]. The operation of the Torre/Tajima System-1 in the context of Figure 14 of Torre is illustrated below and annotated with the color-coding scheme described in §IX.A.1 above including the

annotation showing the encryption key and the SHA-1 Digital Signature (in pink)
inside the tag (in purple).

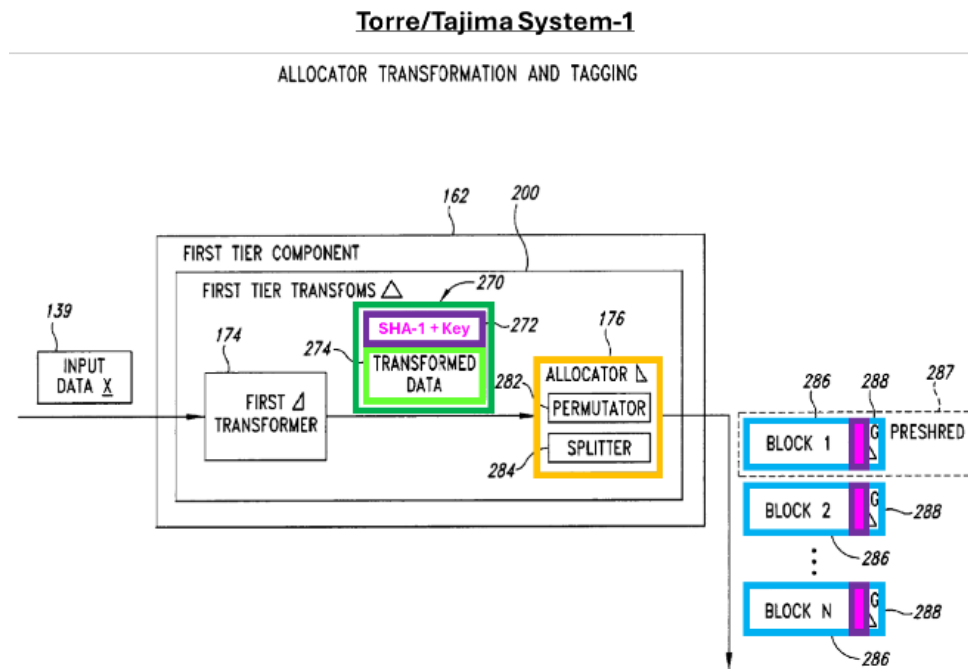


Fig. 14

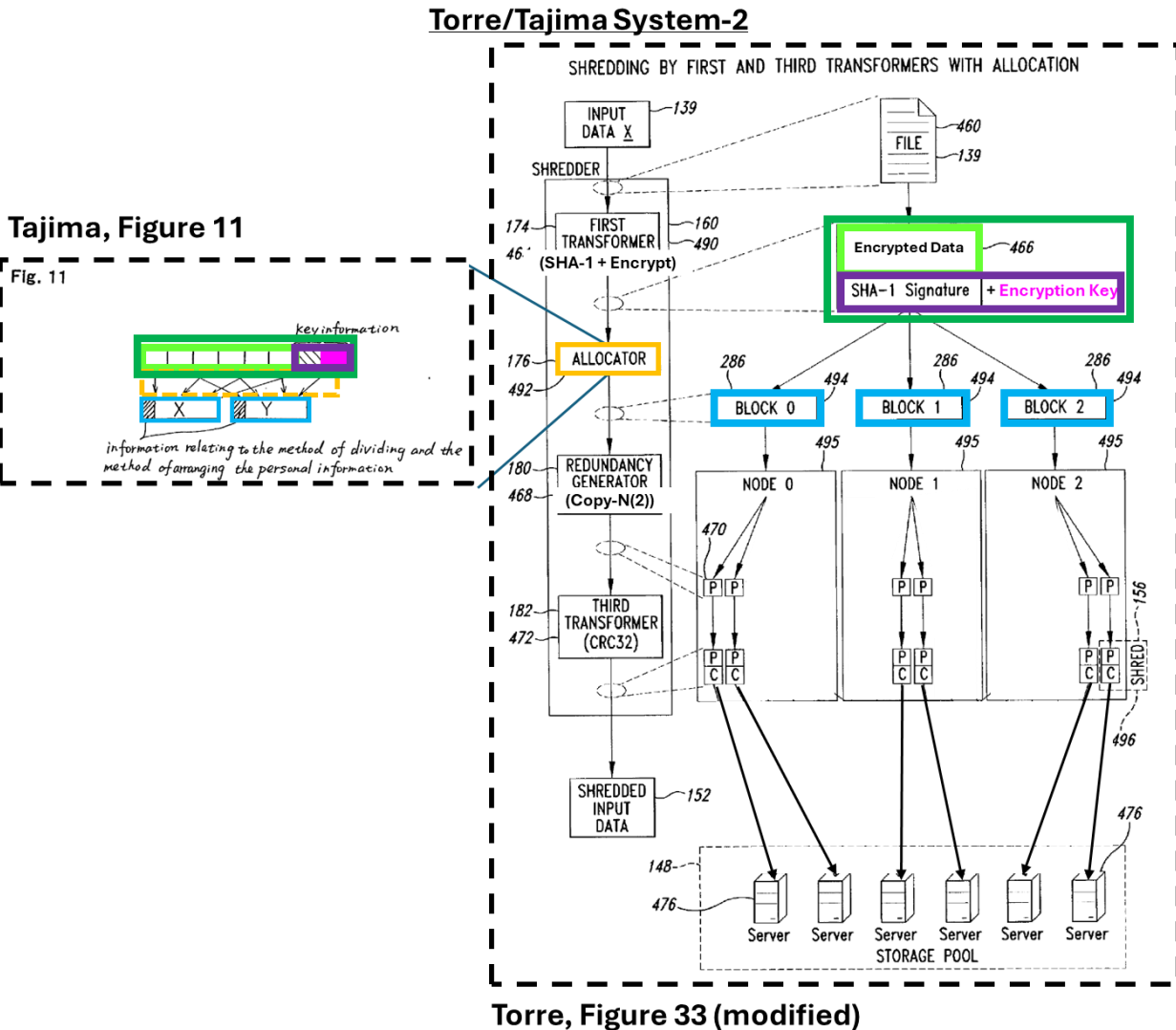
EX1003 (Torre), Figure 14 (annotated).

213. In this combination—referred to as “Torre/Tajima System-2”—the allocator operates by determining “arbitrary length[s] according to a function of, for example, random numbers, time, or file capacity,” and then uses those values to divide the combined set of transformed data and the encryption key into data portions that match the assigned arbitrary lengths. EX1004 (Tajima), [0106], [0109].

214. Following this division, each resulting portion may be placed into a separate pre-shred, consistent with the method illustrated in Tajima’s Figure 11

(annotated with the color-coding scheme described in §IX.A.1 above). EX1004 (Tajima), [0106], [0109], [0156].

215. A composite, annotated diagram of Torre/Tajima System-2 is shown below; it combines Torre's Figure 34 with Tajima's Figure 11, annotated with the color-coding scheme described in §IX.A.1 above including the annotation showing the encryption key and the SHA-1 Digital Signature (in pink) inside the tag (in purple).



a. Reasons to Combine Torre/Tajima System-1 Further with Tajima As Provided In Torre/Tajima System-2

216. A POSITA would have been motivated to combine Torre and Tajima as reflected in the Torre/Tajima System-2 figures above for at least the following three reasons.

217. **First**, a POSITA would have found it logical to integrate the method from Tajima's Figure 11 into Torre/Tajima System-1, given that both Torre and Tajima are focused on addressing the same fundamental challenge using closely related techniques. *See* §IX.A.4.i above [Torre/Tajima System-1].

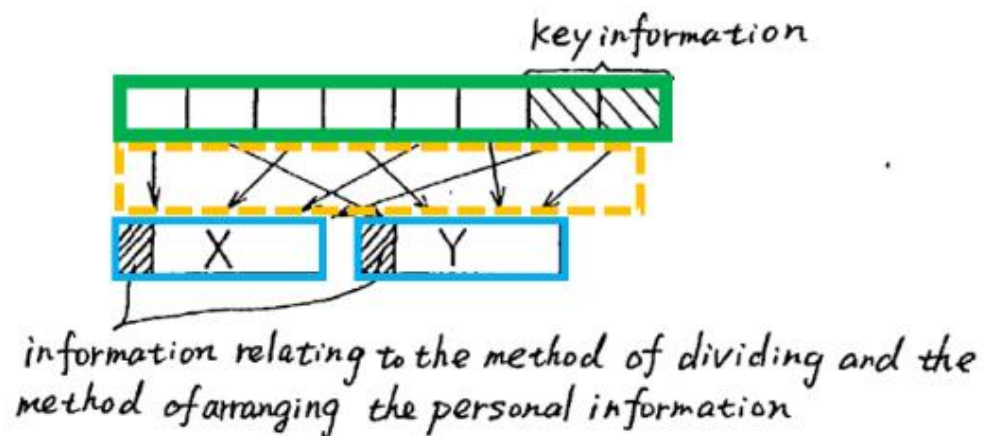
218. **Second**, a POSITA would have also been particularly motivated to incorporate the method from Tajima's Figure 11 into Torre/Tajima System-1 based on Torre's own guidance. Torre explicitly states that the "scheme of remapping" is "not limited," and that both the transformed data and associated tags can be reorganized using any suitable approach that "depend[s] upon particular situations." EX1003 (Torre), [0101] ("the general scheme of remapping is not limited and specific remapping is dependent upon particular situations").

219. This express flexibility would have prompted a POSITA to explore alternative remapping and splitting techniques. Tajima provides precisely such an approach, one that adjusts based on factors like random values, timing, or data size. EX1004 (Tajima), [0106] ("personal information ... is first divided into a plurality of data portions each of different arbitrary length according to a function of, for example, random numbers, time, or file capacity, and this plurality of data portions is then arranged into two data fragments.").

220. **Third**, A POSITA would have also recognized that Tajima's approach enhances data security by introducing unpredictability into the way data is

partitioned. Specifically, by distributing data portions of “arbitrary length[s]” across pre-shreds in a round-robin pattern—as depicted in Tajima’s Figure 11 (annotated with the color-coding scheme described in §IX.A.1 above)—it becomes more difficult for an unauthorized party to reassemble the original transformed data. Thus, implementing Torre’s allocator using Tajima’s method would have amounted to using a known technique of rearranging and splitting received information (Tajima’s Figure 11) to improve a similar device (Torre/Tajima’s allocator) in the same way (increasing arbitrariness in pre-shreds/data fragments generated by the allocator).

Fig. 11



EX1004 (Tajima), Figure 11.

221. Next, A POSITA would have had a reasonable expectation of success in applying Tajima’s method to implement the allocator in Torre/Tajima System-1 for at least the following four reasons.

222. **First**, the combination would have produced no surprising or unpredictable results. As discussed above, the allocator in Torre/Tajima System-1 is responsible for remapping and segmenting the transformed data along with the tag. EX1003 (Torre), [0099], [0102]. Tajima performs the same function—remapping and dividing transformed content (such as encrypted data) along with associated metadata (such as the encryption key). EX1004 (Tajima), [0109]; [0156]; Claim 8. A POSITA would have thus recognized that the combination amounts to a simple substitution of one known technique (Torre’s data allocation technique) for another (Tajima’s method) to obtain predictable results of rearranging and splitting the received data and its tag according to Tajima’s method.

223. **Second**, Torre notes that “the embodiments disclosed herein, in whole or in part, can be equivalently implemented ... as one or more computer programs running on one or more computers (e.g., as one or more programs running on one or more data processing systems).” EX1003 (Torre), [0141]. As such, incorporating Tajima’s method into the Torre/Tajima System-1 architecture would have simply involved implementing the corresponding functionality in software, by writing program code. That task would have been well within the abilities of a POSITA. *See* EX1003 (Torre), [0141].

224. **Third**, the allocator in Torre/Tajima System-1 and the method described in Tajima both operate on substantially similar types of data—specifically,

encrypted content along with associated metadata such as the encryption key. In Torre, this includes the transformed (encrypted) data and its tag; in Tajima, the encrypted data and the key used for encryption. The output in both systems is likewise analogous: pre-shreds in Torre and data fragments in Tajima. Given this structural and functional similarity, a POSITA would have readily appreciated that Tajima's method could be substituted for Torre's existing allocation technique to achieve the same functional result—dividing incoming data—using a predictable, well-understood approach such as assigning arbitrarily sized portions to output fragments.

225. **Fourth**, incorporating Tajima's method into Torre/Tajima System-1 would have been simple and would not have required extensive experimentation or imposed undue burden, because both systems already disclose remapping and allocation of encrypted data along with associated metadata. For the same reasons, the combination would have yielded predictable results (because substituting one known allocation technique for another to divide and rearrange data is a well-understood design choice). Lastly, a POSITA would have had a reasonable expectation of success because Tajima does nothing more than describe the allocation process in more detail than Torre.

226. At a minimum, it would have been obvious to try Tajima's method of dividing and rearranging data. As Torre itself recognizes, the "scheme of

remapping” is “not limited,” and the transformed data and the tags can be “remapped” using any suitable method that “depend[s] upon particular situations.” EX1003 (Torre), [0101].

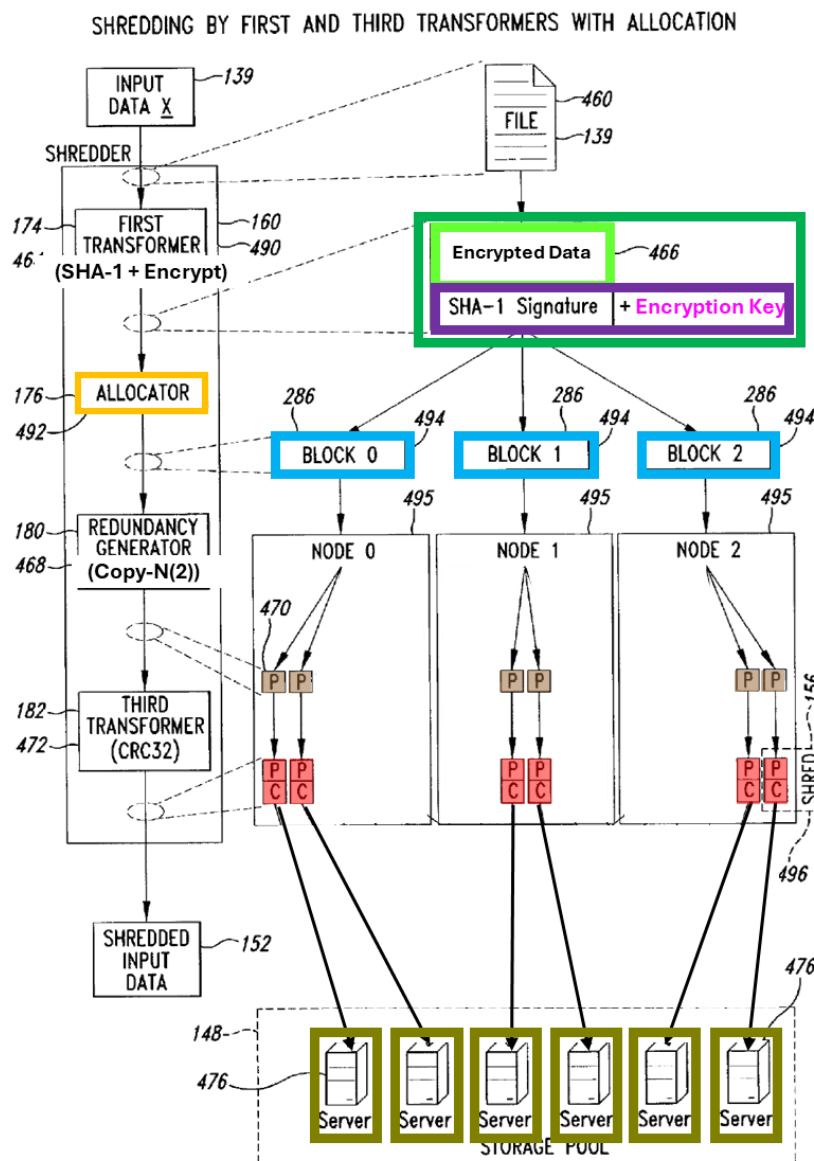
227. This teaching highlights a design need to implement a remapping approach that can adapt based on contextual requirements. A POSITA would have understood that there were a finite number of predictable options available to accomplish such remapping, including: (1) hybrid combinations of chunking and interleaving (EX1003 (Torre), [0100]); (2) “divid[ing] [the received data] into a plurality of data portions each of equal arbitrary length” and “arrang[ing]” the data portions to pre-shreds (EX1004 (Tajima), [0103]); and (3) “divid[ing] [the received data] into a plurality of data portions each of different arbitrary length according to a function of, for example, random numbers, time, or file capacity” and “arrang[ing]” the data portions to pre-shreds (EX1004 (Tajima), [0106]). Each of these alternatives would have been expected to succeed for the same reasons already discussed in §IX.A.4, for example, the combination would have merely amounted to a simple substitution of one known technique (Torre’s data allocation technique) for another (the identified solutions) to obtain predictable results of rearranging and splitting the received transformed data and its tag.

228. Torre/Tajima System-1 and Torre/Tajima System-2 are identical apart from their respective allocator implementations. Accordingly, throughout the

forthcoming analysis, where distinctions between allocator methods are not relevant, both systems are jointly referred to as the “Torre/Tajima System.”

229. The Torre/Tajima System is illustrated below using Figure 34 of Torre, annotated with the color-coding scheme described in §IX.A.1 above.

Torre/Tajima System



5. **Claim 1: A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.**

230. Claim 1 would have been obvious in view of the Torre/Tajima System.

- i. **[1Pre] A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising:**

231. To the extent the preamble is limiting, Torre discloses preamble [1Pre].

232. **First**, Torre discloses “[a] method for securing [input data].” Torre describes “[a] system and method for shredding and deshredding ... to *store and retrieve data* while emphasizing considerations such as redundancy, integrity, and *security of data*.” EX1003 (Torre), [0049]. Additionally, Torre notes that its “shredding system and method ... is envisioned to be used with any other encryption method that provides a *desired level of security* dependent upon such other factors

as proper key management.” EX1003 (Torre), [0085]. *See also* EX1003 (Torre), [0095]–[0096], [0109].

233. Torre further explains that “during allocation, certain particular bytes or other groupings of data may be rearranged based upon particular processing to occur in subsequent steps or for certain portions of the data to be secretly mapped to certain storage units for reasons such as *enhanced security*.” EX1003 (Torre), [0109].

234. *Second*, the input data handled by Torre—and therefore by the Torre/Tajima System—corresponds to the claimed “data set.” Torre states that “[t]he input data can be a *file or another form*” and may be produced “by operating software programs and by receiving or recording data such as live audio or *video*, or audio or *video* broadcast streams.” EX1003 (Torre), [0050]. This disclosure encompasses input data such as video content composed of a series of video frames. *See* §V.A above. *See also* EX1024 (Rangan), 2 (“Digitization of motion video yields a sequence of frames, and that of audio yields a sequence of samples Frame is the basic unit of video. Sample is the basic unit of audio.”). A sequence of video frames constitutes a “data set” because it is a group of related data elements (*i.e.*, the frames).

235. Notably, in its litigation with Google, Patent Owner interpreted the term “*data set*” broadly to mean “collection of information for storage.” EX1013 (PO’s

Google Litigation Construction), p. 8. Under that construction, a set of video frames qualifies as a “data set.”

236. Indeed, a set of video frames 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 ’802 Patent and shares its specification—Patent Owner adopted a broad construction of “data set” as meaning a “collection of information for storage.” EX1013 (PO’s Google Litigation Construction), 8. The “input data” in the Torre/Tajima system meets that definition. As discussed in just above, Torre teaches that the input can be a video file composed of multiple video frames. As each frame constitutes “information for storage,” the full video file represents a “collection of information for storage” as claimed.

237. **Third**, Torre teaches that “*the method steps [are] implemented by a programmed computer system.*” EX1003 (Torre), [0141]. Torre further explains that the disclosed functionality may be realized “as one or more **computer programs running on one or more computers** (e.g., as one or more programs running on one or more data processing systems), as one or more programs running on one or more controllers (e.g., microcontrollers)[,] as one or more programs running on one or more processors e.g., microprocessors, as firmware, or as virtually any combination thereof.” EX1003 (Torre), [0141]. *See also* EX1003 (Torre), [0142].

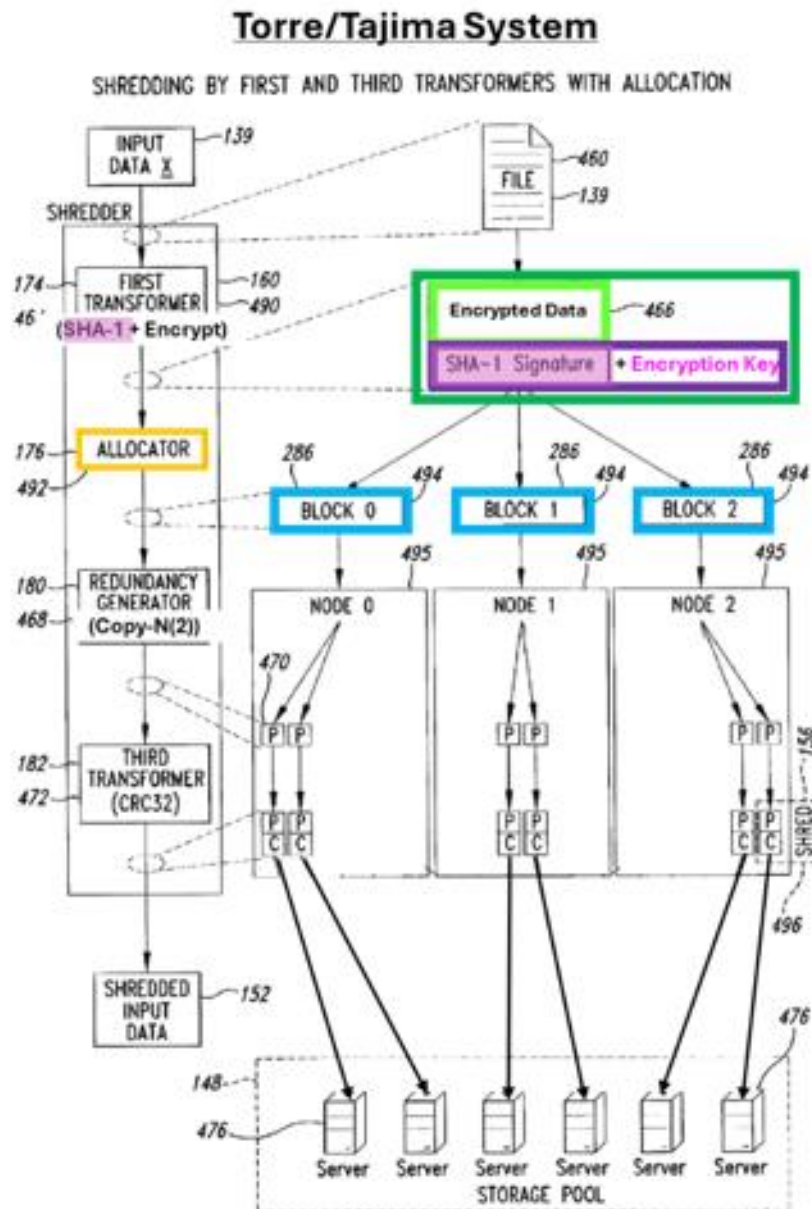
238. **In sum**, Torre discloses “[a] *method for securing a data set* [Torre’s method for shredding and deshredding input data for secure storage], *the method steps implemented by a programmed computer system* [Torre’s computer programs running on one or more computers].”

ii. [1A] creating integrity information using the data set;

239. Torre discloses claim limitation [1A].

240. Torre explains that “[i]n the process of shredding, input data [*data set*] is generally transformed into many smaller pieces of data called shreds.” EX1003 (Torre), [0051]. During this process, “a first transformer 174 perform[s] the first transformation functions” on the input data. EX1003 (Torre), [0073], [0062] (“data transformed by the first transformations”), Figure 34. One such transformation function includes “digital signature generation.” EX1003 (Torre), [0084] (“The types of transformation functions selected by the first transformer method 230 include ... *digital signature generation*.”). For example, Torre describes a first transformer 464 that “receives the input data 139 ... and uses a *SHA-1 signature generation function* to generate a *signature* associated with the received data to generate a pre-shred 466 containing the received data and a tag of the *SHA-1 signature*.” EX1003 (Torre), [0131].

241. The described system and its operation are illustrated below, which annotates Figure 34 of Torre with the color-coding scheme described in §IX.A.1 above.



EX1003 (Torre), FIG. 34 (annotated).

242. The “**digital signature**” meets the claimed “**integrity information**” limitation. Torre explains that a digital signature—such as SHA-1—can be used to validate the “**data integrity**” of reconstructed information. EX1003 (Torre), [0086] (“Implementations using **signature generation allows for data integrity checking** to verify the validity of reconstruction of the input data 139 from the shredded input data 156.”), [0088]. This aligns with the disclosure in the ’802 Patent, which states that “[e]xamples of integrity information may include ... **digital signature** information.” EX1001 (’802 Patent), 64:30–35. *See also* §V.C.1 above.

243. Accordingly, Torre describes using a first transformer to generate a **digital signature**—i.e., “**integrity information**”—from the input data, which corresponds to the claimed “**data set**.”

244. **In sum**, Torre discloses “**creating integrity information** [Torre’s digital signature information] *using the data set* [Torre’s input data].”

iii. [1B] encrypting the data set based on an encryption key to produce an encrypted data set;

245. Torre discloses claim limitation [1B].

246. As an initial matter, encrypting with an encryption key was well known in the art long before the priority date of the ’802 patent. *See* §V.C above; EX1023 (Downs), 12:44-58 (“In the Secure Digital Content Electronic Distribution System 100 the Content Provider(s) 101 encrypts the content using symmetric algorithms.

They are called symmetric algorithms because the same key is used to encrypt and decrypt data. The data sender and the message recipient must share the key. The shared key is referred to here as the symmetric key. The Secure Digital Content Electronic Distribution System 100 architecture is independent of the specific symmetric algorithm selected for a particular implementation. Common symmetric algorithms are DES, RC2 and RC4. Both DES and RC2 are block cipher [sic]. A block cipher encrypts the data using a block of data bits at a time. DES is an official US government encryption standard, has a 64-bit block size, and uses a 56-bit key.”).

247. First, Torre teaches “*encrypting the data set ... to produce an encrypted data set*.” As discussed in §IX.A.2 above [Torre], the first transformer is capable of performing encryption on the input data (“*data set*”) in addition to generating a digital signature (“*integrity information*”).

248. For instance, Figure 11 in Torre—reproduced and annotated below—illustrates “[a] table of allowed transformation combinations” involving the first and second transformers. EX1003 (Torre), [0091], Figure 11. The figure includes a “Fpre column [that] ... identifies which functions are performed by the first transformer 174 [colored in yellow] and the second transformer 178 [colored in green].” EX1003 (Torre), [0091]. Within that table, “‘E’ represents the encryption functions, and ‘S’ represents the signature generation functions.” EX1003 (Torre), [0091]. Based on this, row 9—highlighted in blue—shows that the first transformer

may apply a signature generation function followed by **encryption**, thereby producing **encrypted data set**. This interpretation is further supported by Torre's statement that "any of the entries in the table of FIG. 11 that had **signature generation** ... before ... **encryption** would be a candidate for high data integrity." EX1003 (Torre), [0092].

1st Transformer 2nd Transformer Advantages

F_{pre}

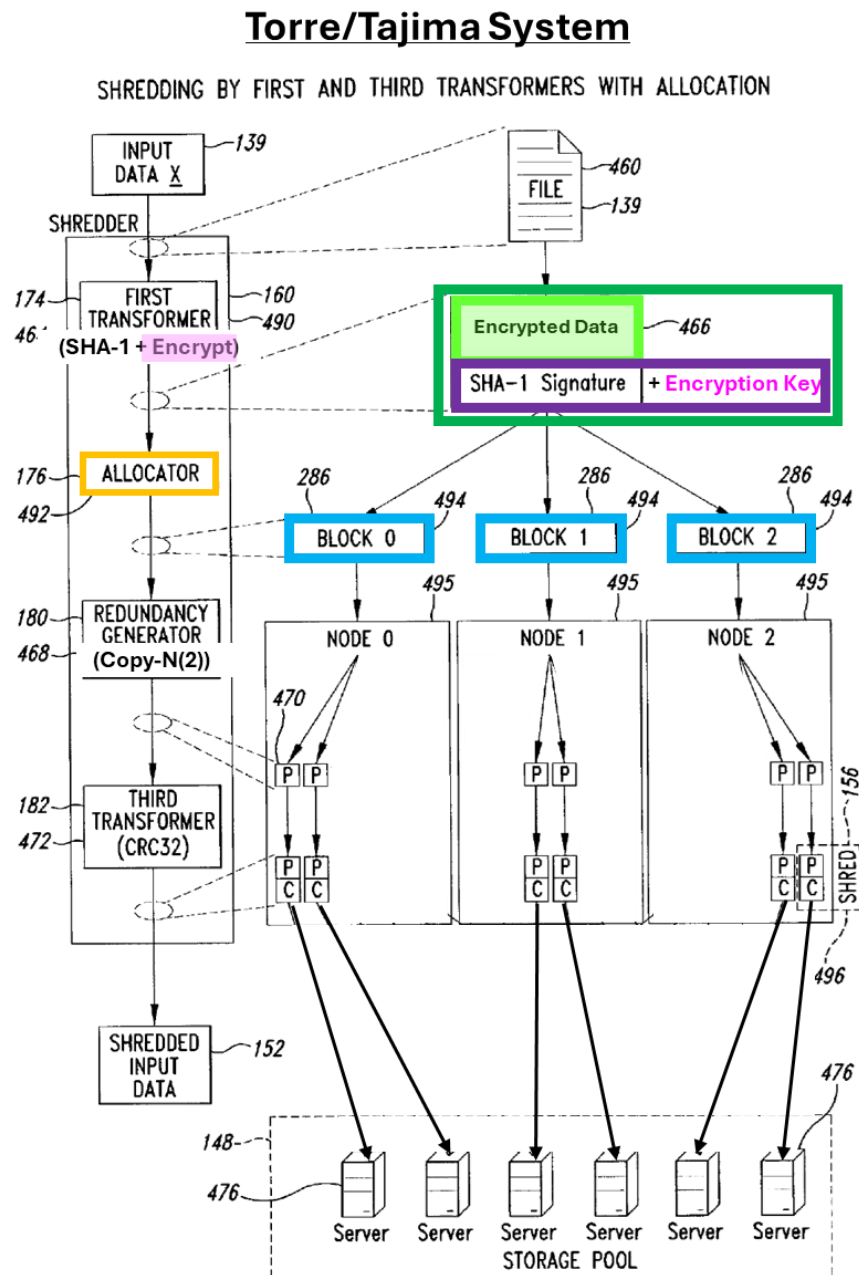
TABLE OF ALLOWED TRANSFORM COMBINATIONS
FOR FIRST AND SECOND TRANSFORMERS

RULE	1st Transformer	2nd Transformer	1	2	3	4	5	6	7	8	9	10	11	12
1	E								7	8				
2	S		1											
3	A			2										
4	C				3									
5	E A		2						7	8				
6	E S							6	7	8				
7	S A		1	2										12
8	S C		1		3									
9	S E		1						7	8				
10	A C			2	3								11	
11	A E			2		4			7		9			
12	A S			2				6				10		
13	C A			2	3		5							
14	C E				3				7	8				
15	C S				3			6						
16	E A S		2					6	7	8				
17	E S A		2					6	7	8				12
18	S A C		1	2	3								11	12
19	S A E		1	2		4			7		9			12
20	S C A		1	2	3		5							12
21	S C E		1		3				7	8				
22	S E A		1	2					7	8				12
23	A C E			2	3	4			7		9		11	
24	A C S			2	3			6				10	11	
25	A E S			2		4		6	7		9	10		
26	A S C			2	3			6					11	
27	A S E			2		4		6	7		9	10		
28	C A E			2	3	4	5		7		9			
29	C A S			2	3		5	6				10		
30	C E A			2	3		5		7	8				
31	C E S				3			6	7	8				
32	C S A			2	3		5	6						12
33	C S E				3			6	7	8				
34	S A C E		1	2	3	4			7		9		11	12
35	S C A E		1	2	3	4	5		7		9			12
36	S C E A		1	2	3		5		7	8				12
37	A C E S			2	3	4		6	7		9	10	11	
38	A C S E			2	3	4		6	7		9	10	11	
39	A S C E			2	3	4		6	7		9	10	11	
40	C A E S			2	3	4	5	6	7		9	10		
41	C A S E			2	3	4	5	6	7		9	10		
42	C E A S			2	3		5	6	7	8		10		
43	C E S A			2	3		5	6	7	8				12
44	C S A E			2	3	4	5	6	7		9			12
45	C S E A			2	3		5	6	7	8				12

Fig. 11

EX1003 (Torre), Figure 11 (annotated).

249. The encryption performed by the first transformer, along with the resulting encrypted data set, is depicted in Torre's annotated FIG. 34 below, annotated with the color-coding scheme described in §IX.A.1 above.



EX1003 (Torre), FIG. 34 (annotated).

250. Second, Torre also explains that the **encryption** step secures the input data “*based on an encryption key.*” EX1003 (Torre), [0085]. Torre notes that “[t]he types of transformation functions selected by the first transformer method 230 include ... **encryption.**” EX1003 (Torre), [0084]. Further, it describes that “[i]mplementations of encryption include ... those that use **block ciphers,**” including “the **Data Encryption Standard (DES).**” EX1003 (Torre), [0085].

251. It was well known in the art that a “block cipher” refers to a technique for encrypting blocks of data using an encryption key, and that DES specifically is a symmetric block cipher—meaning the same key is used for both encryption and decryption. *See* EX1023 (Downs), 12:44-58 (“In the Secure Digital Content Electronic Distribution System 100 the Content Provider(s) 101 encrypts the content using symmetric algorithms. They are called symmetric algorithms because the same key is used to encrypt and decrypt data. The data sender and the message recipient must share the key. The shared key is referred to here as the symmetric key. The Secure Digital Content Electronic Distribution System 100 architecture is independent of the specific symmetric algorithm selected for a particular implementation. Common symmetric algorithms are DES, RC2 and RC4. Both DES and RC2 are block cipher [sic]. A block cipher encrypts the data using a block of data bits at a time. DES is an official US government encryption standard, has a 64-bit block size, and uses a 56-bit key.”). *See also* §V.C above.

252. **In sum**, Torre discloses “*encrypting the data set* [Torre’s input data] *based on an encryption key* [Torre’s key] *to produce an encrypted data set* [Torre’s encrypted data set] 456].”

- iv. [1C] generating **data splitting information**, wherein the data splitting information is usable to determine into which of a **plurality of shares of data** a unit of data of the encrypted data set will be placed;

- a. **Torre/Tajima System-1**

253. Torre/Tajima System-1 satisfies claim limitation [1C] because Torre discloses or renders obvious claim limitation [1C].

254. **First**, Torre discloses—or at a minimum renders it obvious—“*generating data splitting information.*” As discussed in §IX.A.4 above [Torre/Tajima], the allocator is responsible for operations that “partition[s] the received data” (including encrypted data set) and “split[s]” that data into data blocks. EX1003 (Torre), [0099], [0102].

255. For instance, the “permutator 282 [in the allocator] re-maps the ordering of the transformed data 274 using various partitioning methods” (EX1003 (Torre), [0102]), such as “**chunking, interleaving, and hybrid combinations** of these two types” (EX1003 (Torre), [0100]). Additionally, the “splitter 284 [in the allocator] **splits** the re-mapped data into data blocks 286” (EX1003 (Torre), [0102]).

Torre further states that “the general scheme of remapping is not limited and specific remapping is *dependent upon particular situations*.” EX1003 (Torre), [0101].

256. A POSITA would have understood that Torre adapts its remapping behavior to “particular situations” by choosing among “various partitioning methods.” EX1003 (Torre), [0102].

257. This process of selecting between different remapping techniques would inherently involve generating corresponding [selection information](#)—*i.e.*, “*data splitting information*.” Without the [selection information](#), it would not be feasible to adjust remapping logic based on context, as Torre indicates is required “dependent upon particular situations.” EX1003 (Torre), [0101].

258. For example, in computer languages, one of the most basic programming primitives (often taught to first year Computer Science majors) is a “switch” or “case” statement, where an input data comprises the information used to select among several choices of actions. A “case statement” is, “[i]n programming languages such as Ada, Pascal, and C, a type of control statement that executes one of several sets of instructions based on some key value. Case statements are used in evaluating situations that can have a number of different results. ‘Case’ in this sense refers to a refinement of a basic IF-THEN type of conditional statement (if A is true, then do B), but a case statement functions more like a series of nested IFs (if A, then do this; else if B, then do that; else ...). In a case evaluation, a variable (such as a

number or a string of characters) is compared against one after another of a series of constants assigned by the programmer. Each constant represents a different case and defines an action to be carried out. When the program finds a constant that matches the variable, it carries out whatever action is dictated by the case in which the match occurs. *See also* constant, control statement, variable.” EX1016 (Microsoft Computer Dictionary), 87.

259. If Patent Owner argues that Torre does not disclose generating such [selection information](#), it would have been, at a minimum, obvious to generate such [selection information](#) to enable switching among “various partitioning methods,” including “*chunking, interleaving, and hybrid combinations* of these two types,” based on “particular situations.” EX1003 (Torre), [0100], [0102]. The concept of using selection information to manage mode switching was well established in the field. *See* EX1016 (Microsoft Computer Dictionary), 87.

260. Moreover, Torre already teaches “generat[ing] information describing [the allocator’s] processing,” which would have necessarily included data reflecting the choice of remapping method. EX1003 (Torre), [0103]. Therefore, this modification would simply involve first “generat[ing] information describing [the allocator’s] processing”—*i.e.*, the [selection information](#) for the remapping strategy—and then applying that information to guide the remapping selection. EX1003 (Torre), [0103].

261. Torre itself provides a clear rationale for the proposed modification by indicating that the allocator may parse and divide the received data “dependent on particular situations.” EX1003 (Torre), [0101], [0109]. “For example[,] during allocation, certain particular bytes or other groupings of data *may be rearranged based upon particular processing* to occur in subsequent steps” EX1003 (Torre), [0109]. This teaching reinforces that remapping behavior can vary dynamically, supporting the generation of selection information as part of the allocator’s operation.

262. A POSITA would have reasonably expected success in generating selection information within the Torre/Tajima System-1, as doing so would simply involve choosing and applying one remapping scheme over another based on the provided selection criteria.

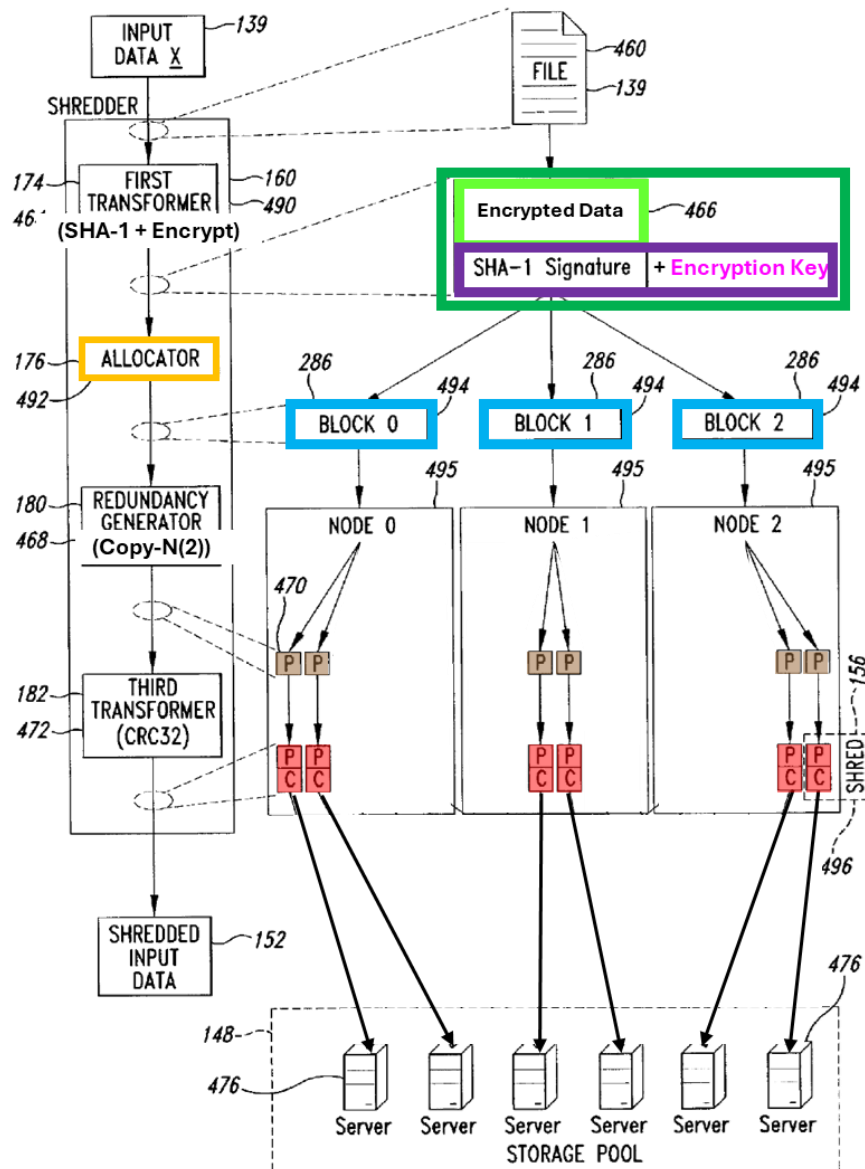
263. The trivial modification would have merely amounted to applying a known switching technique (*e.g.*, generating and using selection information for mode switching) to a known device ready for improvement (Torre’s allocator that may implement different remapping schemes depending on particular situations) to yield predictable results of executing the remapping mode selected by the selection information.

264. *Second*, Torre teaches that such **selection information** “*is usable to determine into which of a plurality of shares of data a unit of data of the encrypted*

data set will be placed.” Once a pre-shred 466 is formed by the first transformer 174 through its transformations, “[t]he allocator 492 *splits* the pre-shred 466 into *three data blocks* 494, which are sent on to three redundancy generators 468 each located in three of the second-tier components 164.” EX1003 (Torre), [0135]. *See also* EX1003 (Torre), [0061] (“The shredding step 124 performs first transformations (step 132), *allocates data transformed by the first transformations into various different blocks of data (step 133)*, and performs second transformations on these blocks of data (step 134).”). And the allocator then uses *selection information* to determine which partitioning technique is used to allocate a unit of data of the *encrypted data set* to one of the *data blocks*. EX1003 (Torre), [0100]-[0102]. This is shown in Figure 34 of Torre, annotated below with the color-coding scheme described in §IX.A.1 above.

Torre/Tajima System-1

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



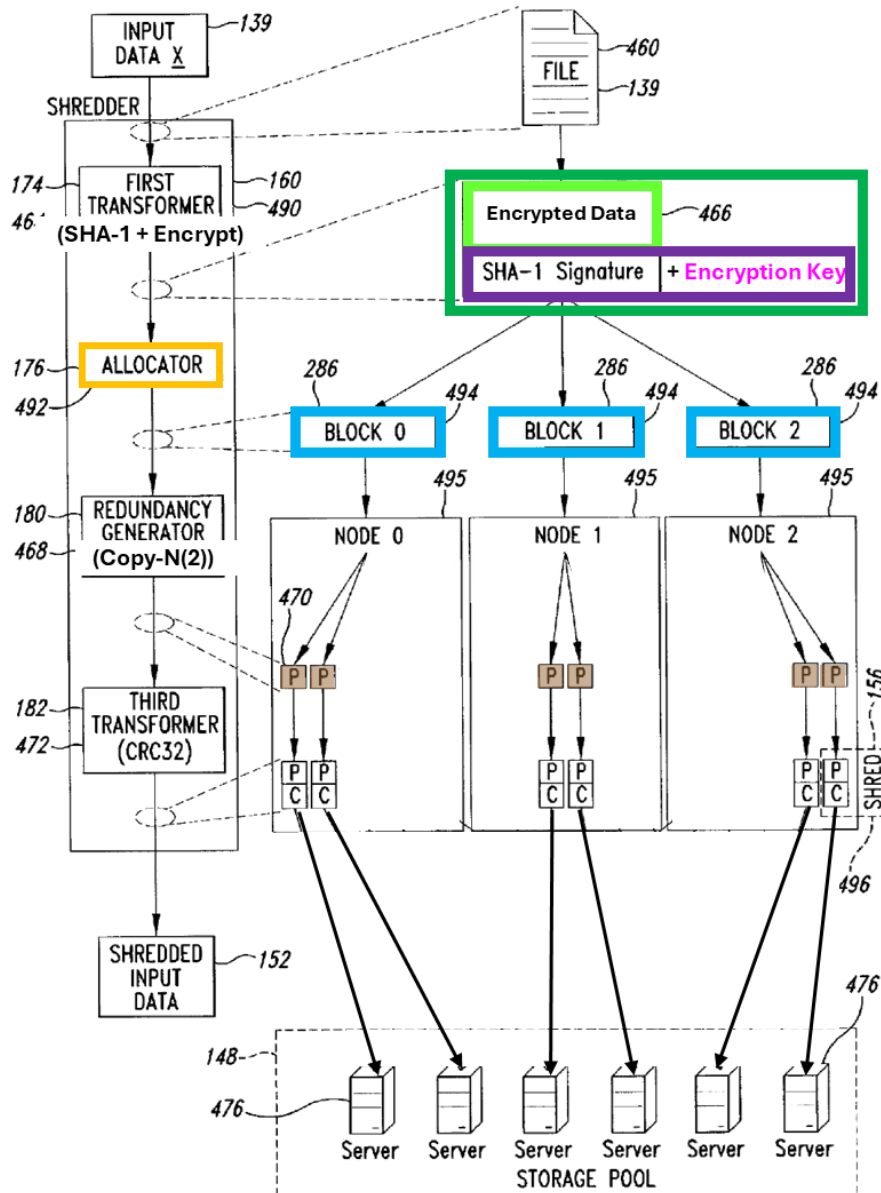
EX1003 (Torre), FIG. 34 (annotated).

265. By assigning a unit of the **encrypted data set** to one of the **data blocks**, the allocator effectively “*determine[s] [using the selection information] into which of a plurality of shares of data a unit of data of the **encrypted data set** will be placed.*”

Once the allocator outputs the **data blocks**, the redundancy generator 180 applies the Copy-N(2) redundancy method to produce **two copies of each data unit** from the received **data block** and passes those **two copies** to the third transformer. EX1003 (Torre), [0113]. The redundancy generator under the Copy-N(2) scheme ensures that **each copy** is an “exact copy” of the **received data block**, which is from the **encrypted data set**. See EX1003 (Torre), [0113] (“A redundancy transformation called Copy-N can be used by some implementations to simply make *multiple exact copies of existing data*.”). This is shown in Figure 34 of Torre, annotated below with the color-coding scheme described in §IX.A.1 above and the copies of each data unit shown in brown.

Torre/Tajima System-1

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION

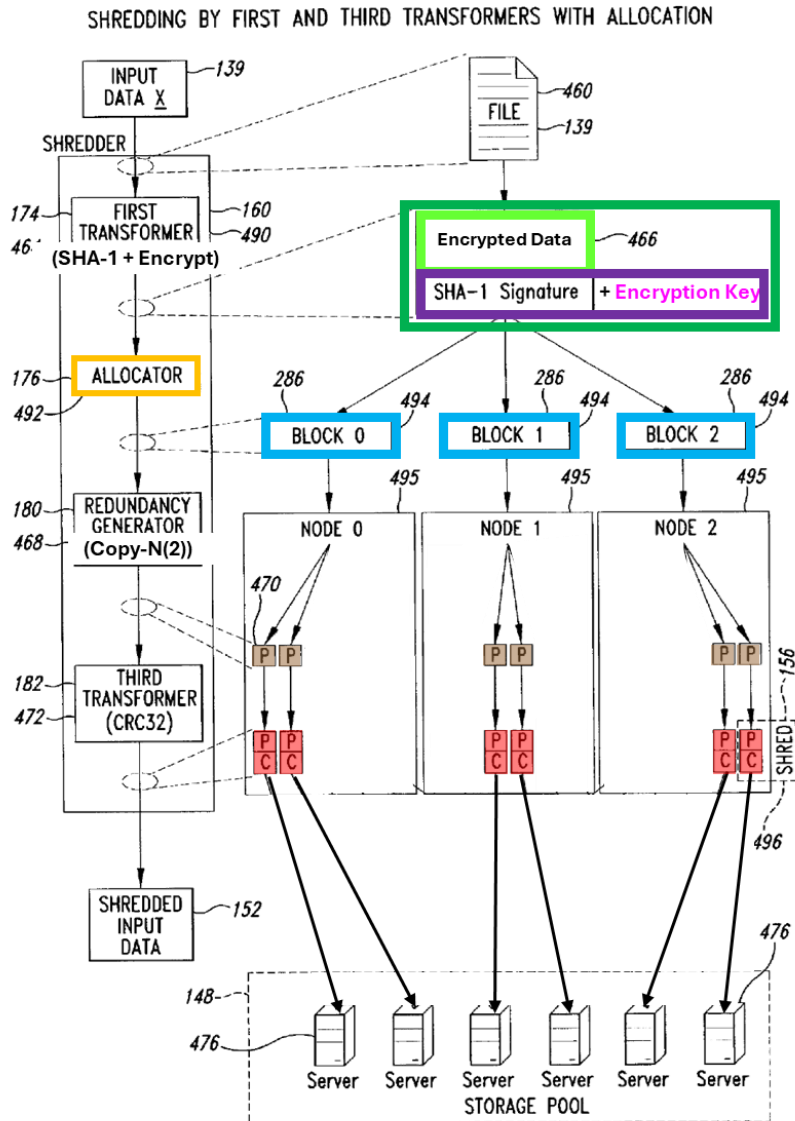


EX1003 (Torre), FIG. 34 (annotated).

266. The third transformer then processes **each unit of data** to generate a corresponding **shred**, as illustrated in Torre's annotated FIG. 34 below. EX1003

(Torre), [0132]. Each resulting shred contains a **unit of data** from its associated **data block**. Therefore, by utilizing the **selection information** (“*data splitting information*”), the allocator determines into which **data block**—and thus into which shred—*i.e.*, “*which of a plurality of shares of data*,” each unit of the **encrypted data set** 466 (“*encrypted data set*”) is assigned. This is shown in Figure 34 of Torre, annotated below with the color-coding scheme described in §IX.A.1 above, with copies of data units shown in brown and the shreds shown in red.

Torre/Tajima System-1



EX1003(Torre), FIG. 34 (annotated).

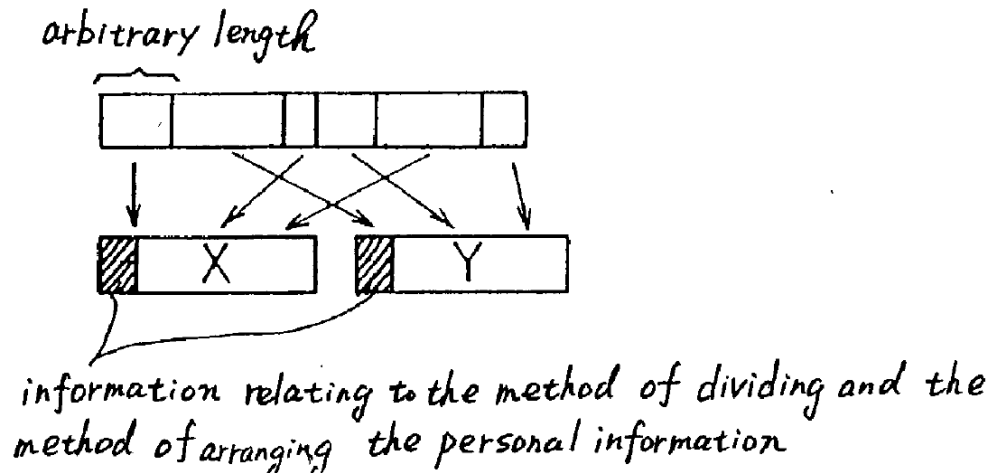
267. **In sum**, Torre discloses or renders obvious “generating *data splitting information* [Torre’s selection information], wherein the *data splitting information* is usable to determine into which of a *plurality of shares of data* [Torre’s shreds 156] a unit of data of the *encrypted data set* [Torre’s encrypted data 466] will be placed.”

b. Torre/Tajima System-2

268. Torre/Tajima System-2 satisfies claim limitation [1C] because Torre in view of Tajima teaches claim limitation [1C].

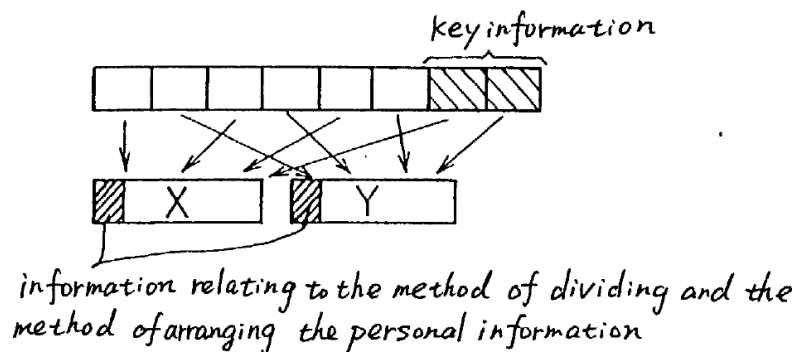
269. **First**, Torre in view of Tajima teaches “*generating data splitting information.*” As discussed in §§IX.A.3 above [Tajima] and IX.A.4 above [Torre/Tajima], Tajima “divide[s]” “the *encrypted [data]* and information regarding the key [*i.e., encryption key*] that is used in the encryption ... into a plurality of data portions each of ... *arbitrary length* as shown in ... FIG. 10.” EX1004 (Tajima), [0109]; Claim 8 (clarifying that “information regarding the key” is “encryption key”). Tajima further explains that “[T]he *arbitrary lengths*” constitute “information relating to the method of *dividing* and method of *arranging*” the data, as shown in Figure 10 of Tajima, below (which is also used in the embodiment of Figure 11 of Tajima) and thus corresponds to the claimed “*data splitting information.*” EX1004 (Tajima), [0107].

Fig. 10



EX1004 (Tajima), FIG. 10.

Fig. 11



EX1004 (Tajima), FIG. 11.

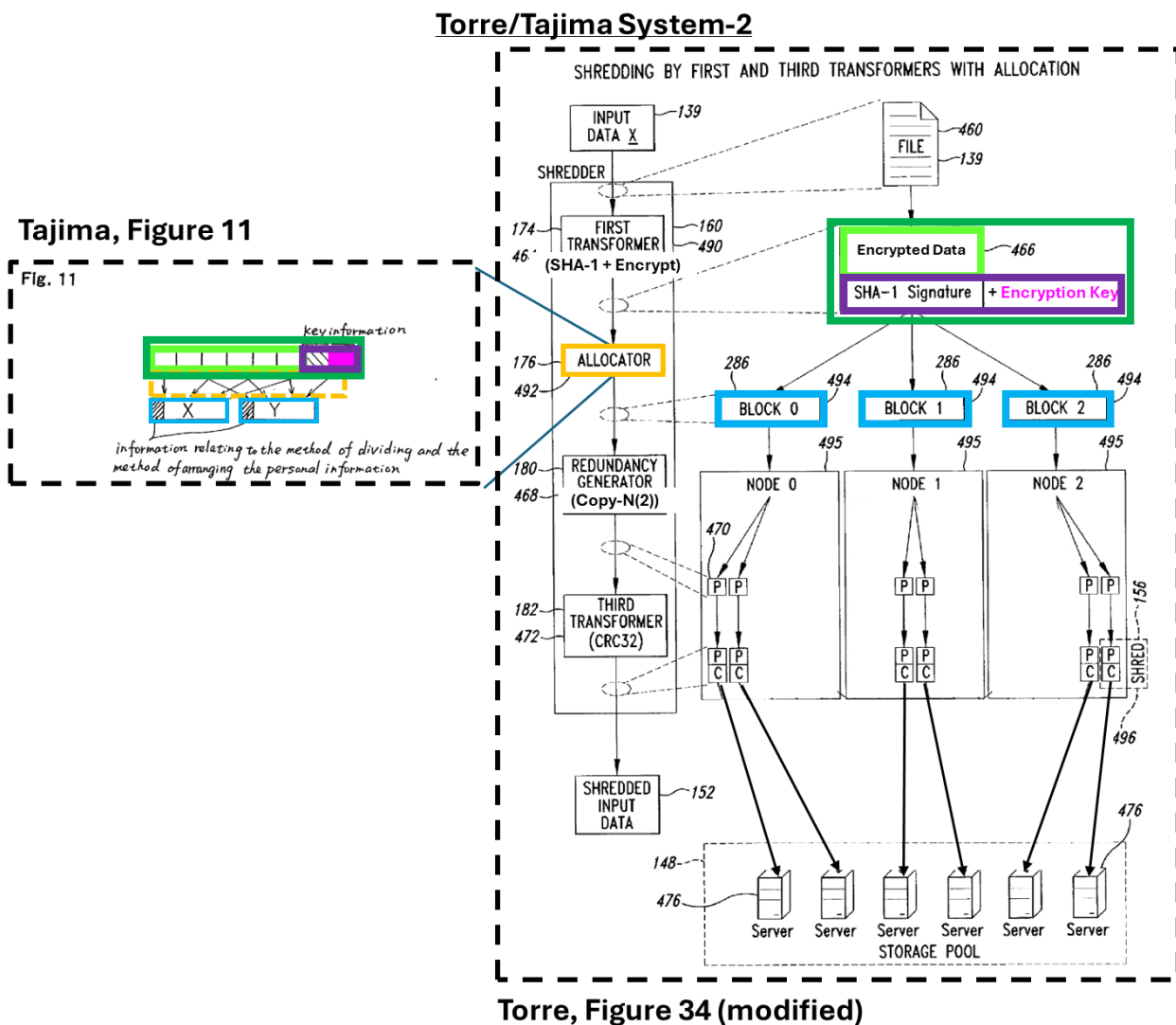
270. Tajima further teaches that the **arbitrary length information** is “generat[ed].” In Tajima, data is “divided into a plurality of data portions each of different **arbitrary length according to a function of, for example, random numbers, time, or file capacity.**” EX1004 (Tajima), [0106]. Therefore, the **arbitrary length**

information depends on parameters that are not predetermined at the time of implementing the method of splitting/arranging; rather, the arbitrary length information depends on parameters that are determined at the time of executing the method of splitting/arranging, such as random numbers or time. Thus, the arbitrary length information is **generated** at the time of executing the method.

271. This understanding of Tajima aligns with additional disclosures in Tajima. Tajima explicitly distinguishes between dividing data into segments of “arbitrary length” and dividing data into segments of “**predetermined** fixed length.” EX1004 (Tajima), [0101], [0106]. This distinction confirms that the arbitrary lengths are not fixed in advance and, instead, are determined dynamically—*i.e.*, generated during execution. Tajima further supports this by explaining that the arbitrary length information “must be attached” to the resulting data fragments (EX1004 (Tajima), [0107]) so that it can later be used to accurately reconstruct the original input data from those fragments (EX1004 (Tajima), [0108]).

272. **Second**, Torre/Tajima System-2 satisfies the requirement that the arbitrary length information “*is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed,*” for the same reasons that the selection information in Torre/Tajima System-1 satisfies this aspect of the limitation. See §IX.A.5.iv.a above [Torre/Tajima System-1]. When the allocator assigns a unit of the encrypted data set to one of the data blocks, it has

“determine[d] [using the arbitrary length information] into which of a plurality of shares of data a unit of data of the encrypted data set will be placed.” This is shown in the composite figure Torre/Tajima System-2 introduced in §IX.A.4.ii, annotated with the color-coding scheme introduced therein and depicted below in FIG. 34 (modified and annotated).



EX1003(Torre), FIG. 34 (modified and annotated); EX1004 (Tajima), FIG. 11 (annotated).

273. **In sum**, Torre in combination with Tajima teaches “*generating **data splitting information** [Tajima’s arbitrary length information], wherein the **data splitting information** is usable to determine into which of a **plurality of shares of data** [Torre’s shreds 156] a unit of data of the **encrypted data set** [Torre’s encrypted data 466] will be placed.*”

- v. [1D] separating the **encrypted data set** into the **plurality of shares** based on the **data splitting information**;

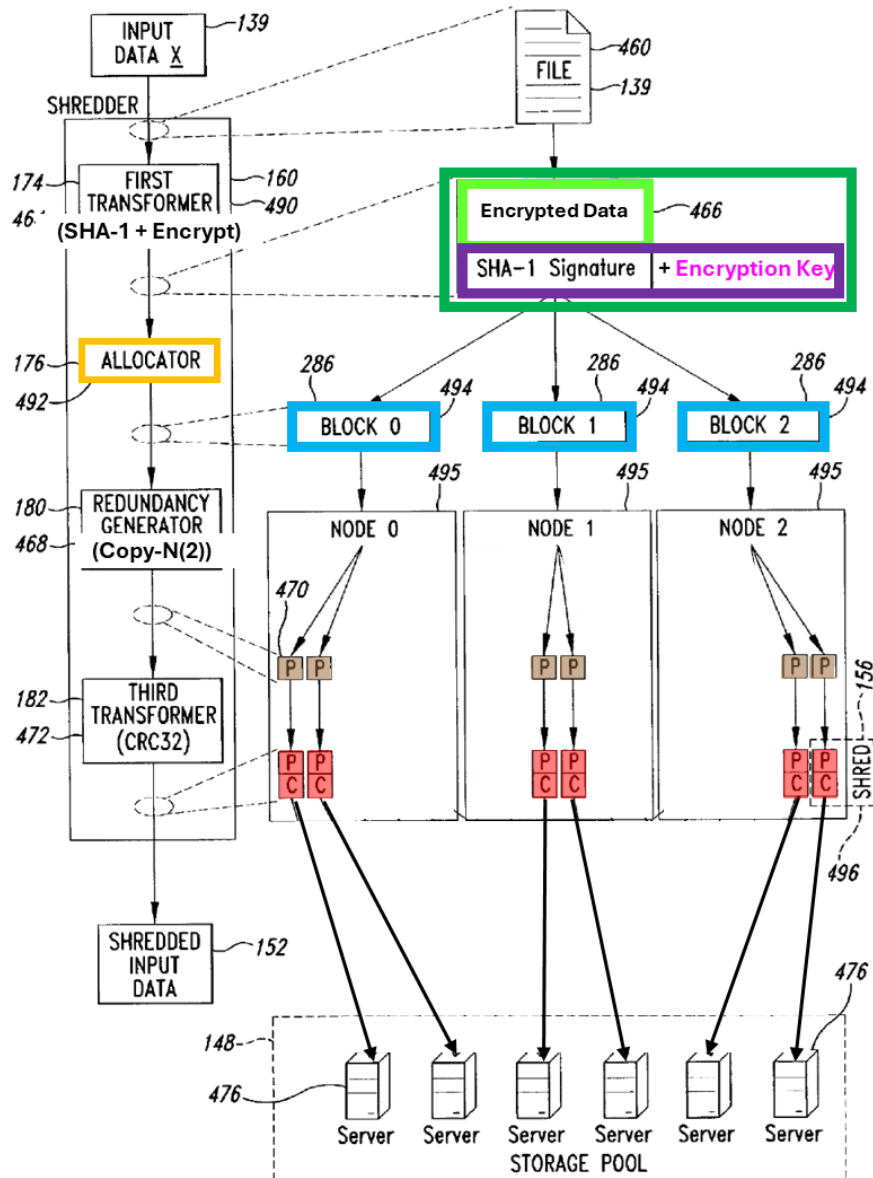
- a. **Torre/Tajima System-1**

274. Torre discloses or renders obvious claim limitation [1D].

275. As discussed in §IX.A.5.iv.a above [claim limitation 1C, Torre/Tajima System-1], Torre discloses or renders obvious the use of **selection information** to divide the **encrypted data set** into **separate data blocks**, which are then processed into **multiple shreds**—*i.e.*, a “*plurality of shares.*” This is further supported by Torre’s explanation that “[t]he splitter 284 [in the allocator] then *splits* the re-mapped data into data blocks 286” EX1003 (Torre), [0100]–[0102].

Torre/Tajima System-1

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (modified and annotated).

276. In sum, Torre discloses or renders obvious “separating the **encrypted data set** [Torre’s encrypted data 466] into the **plurality of shares** [Torre’s shreds 156] based on the **data splitting information** [Torre’s selection information].”

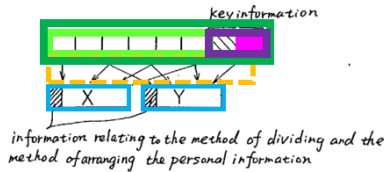
b. Torre/Tajima System-2

277. Torre in view of Tajima teaches claim limitation [1D].

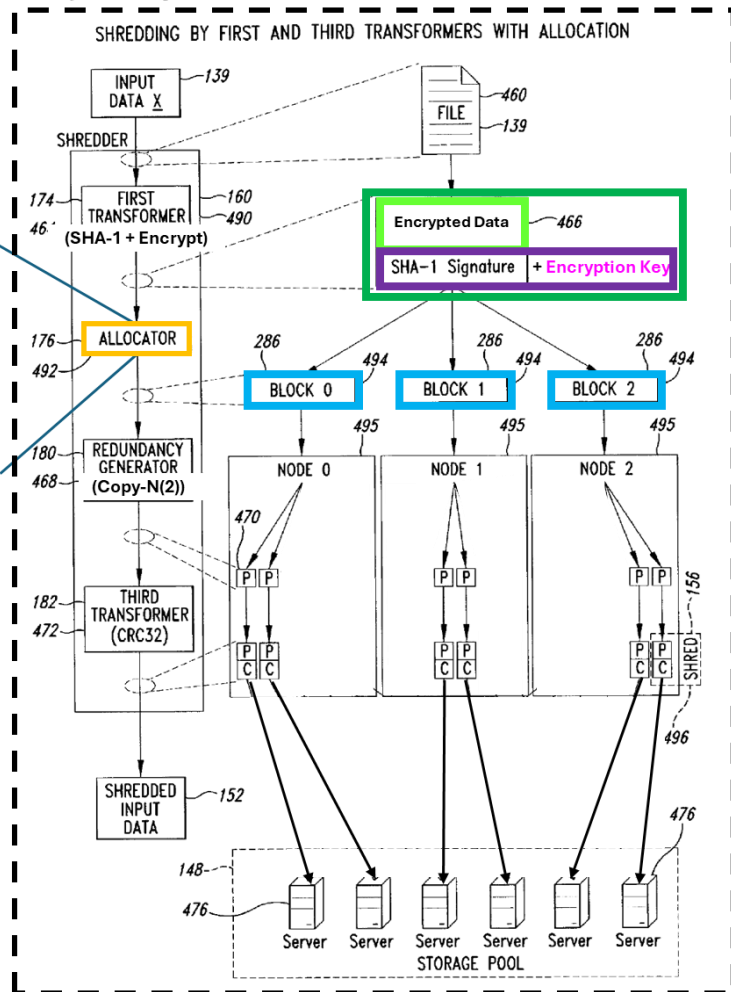
278. As detailed in §IX.A.5.iv.b above [claim limitation 1C, Torre/Tajima System-2], the allocator in Torre—using the generated arbitrary length information (“*data splitting information*”)—divides the encrypted data set into data blocks. These blocks are then further processed into multiple shreds, corresponding to the claimed “plurality of shares,” as shown below. EX1003 (Torre), [0131]–[0136], Figure 34; EX1004 (Tajima), [0106]–[0107], Figure 11 (annotated below with the color-coding scheme described in §IX.A.1 above).

Tajima, Figure 11

Fig. 11



Torre/Tajima System-2



Torre, Figure 34 (modified)

EX1003 (Torre), FIG. 34 (modified and annotated).

279. **In sum**, Torre in combination with Tajima teaches “*separating the encrypted data set* [Torre’s encrypted data 466] *into the plurality of shares* [Torre’s shreds 156] *based on the data splitting information* [Tajima’s arbitrary length information].”

- vi. [1E] including in the **plurality of shares** data indicative of (a) the **encryption key** and (b) the **integrity information**; and

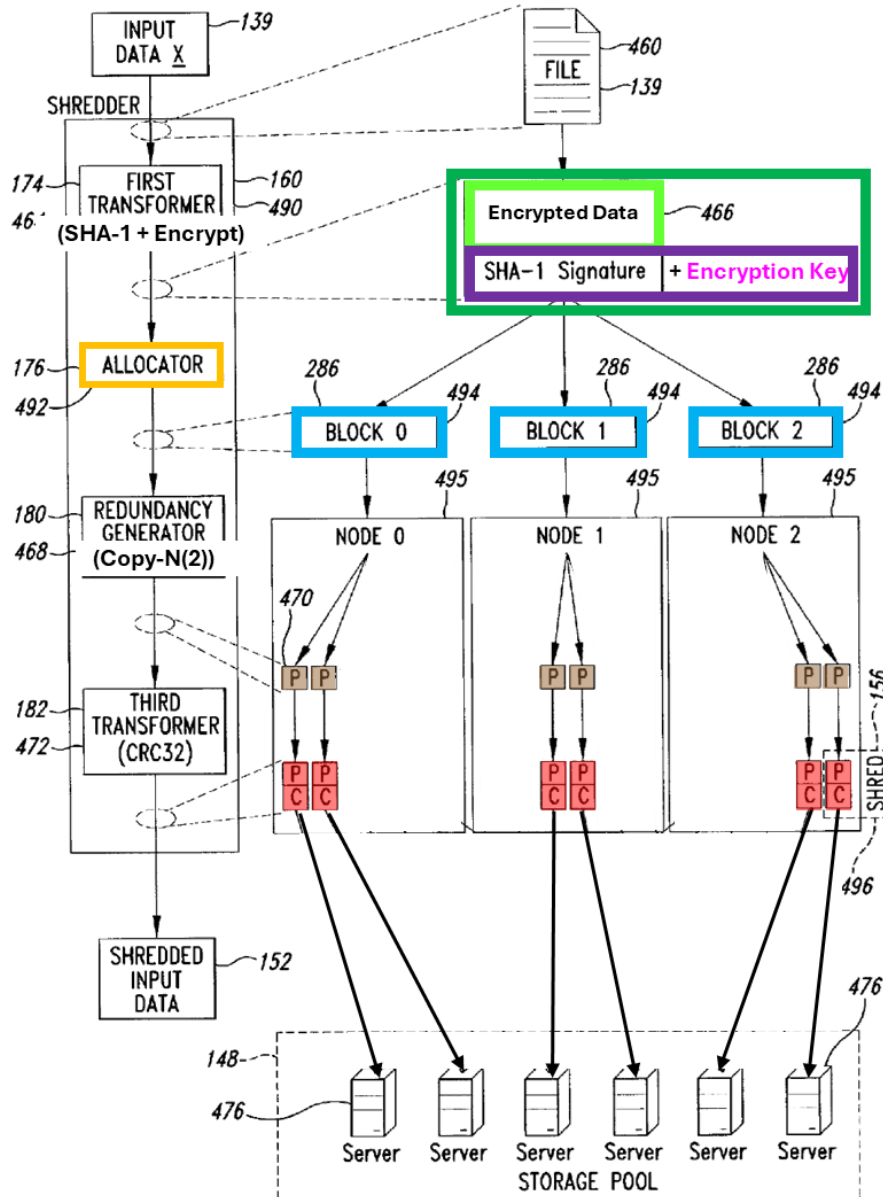
280. Torre in view of Tajima teaches claim limitation [1E].

281. As explained in §IX.A.4 above [Torre/Tajima], in the Torre/Tajima System, the first transformer provides a **pre-shred output** that contains: (1) the **encrypted data** and (2) a tag that includes **integrity information** (such as a **SHA-1 signature**) (as taught by Torre) along with the **encryption key** (as taught by Tajima), as shown below. See §IX.A.4 above [Torre/Tajima]; EX1003 (Torre), Figure 11 (Rule 9), [0091], [0131] (“a pre-shred 466 containing the received data and a tag of the **SHA-1 signature**”), [0134], Figure 34; EX1004 (Tajima), [0078], [0085], [0123], Claim 8 (“encrypts said personal information and divides the **encrypted personal information** and the **encryption key** used in the encryption into a plurality of data portions” that form **data fragments**).

282. Next, as discussed in §IX.A.5.iv above for claim limitation [1C], the allocator in Torre/Tajima System segments and arranges the **pre-shred output**—containing both the **SHA-1 signature** and **encryption key**—into **multiple pre-shreds** 286/494. EX1003 (Torre), [0135]; EX1004 (Tajima), [0106], [0109]. These are further processed by the redundancy generator 180/468 and third transformer 182/472, which divide the **multiple pre-shreds** 286/494 into a **plurality of shreds** 156/496. EX1003 (Torre), [0131]–[0132], [0135].

Torre/Tajima System

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (modified and annotated).

283. Because the output of the first transformer—the **pre-shred output 466** containing the **SHA-1 signature** and **encryption key**—is rearranged and divided into

the **shreds**, the **shreds** contain “data indicative of (a) the **encryption key** and (b) the **integrity information**.”

284. The **shreds** are subsequently used by a deshredder which can recover the encrypted data, the **encryption key**, and the **SHA-1 signature** from the **shreds**. EX1003 (Torre), [0053] (“Information describing how the shreds were created is used to *reassemble* the shreds through processes inverse to those used to create the shreds. *This information is stored in tags typically found within each shred.*”), [0054], [0134] (“*reconstruct the pre-shred* 466 containing data and a tag indicating that *SHA-1 signature* generation was performed by the first transformer 464 during shredding.”); EX1004 (Tajima), [0110] (during data reconstruction, “data fragment X and data fragment Y are *combined* based on information relating to the method of dividing and the method of arranging if information relating to the method of dividing and the method of arranging has been attached to data fragments X and Y, and the *key information that was attached to the encrypted personal information is then used to decrypt the encrypted personal information.*”).

285. **In sum**, Torre in combination with Tajima teaches “including in the **plurality of shares** [Torre’s shreds 156] data indicative of (a) the **encryption key** [Tajima’s encryption key] and (b) the **integrity information** [Torre’s SHA-1 digital signature].”

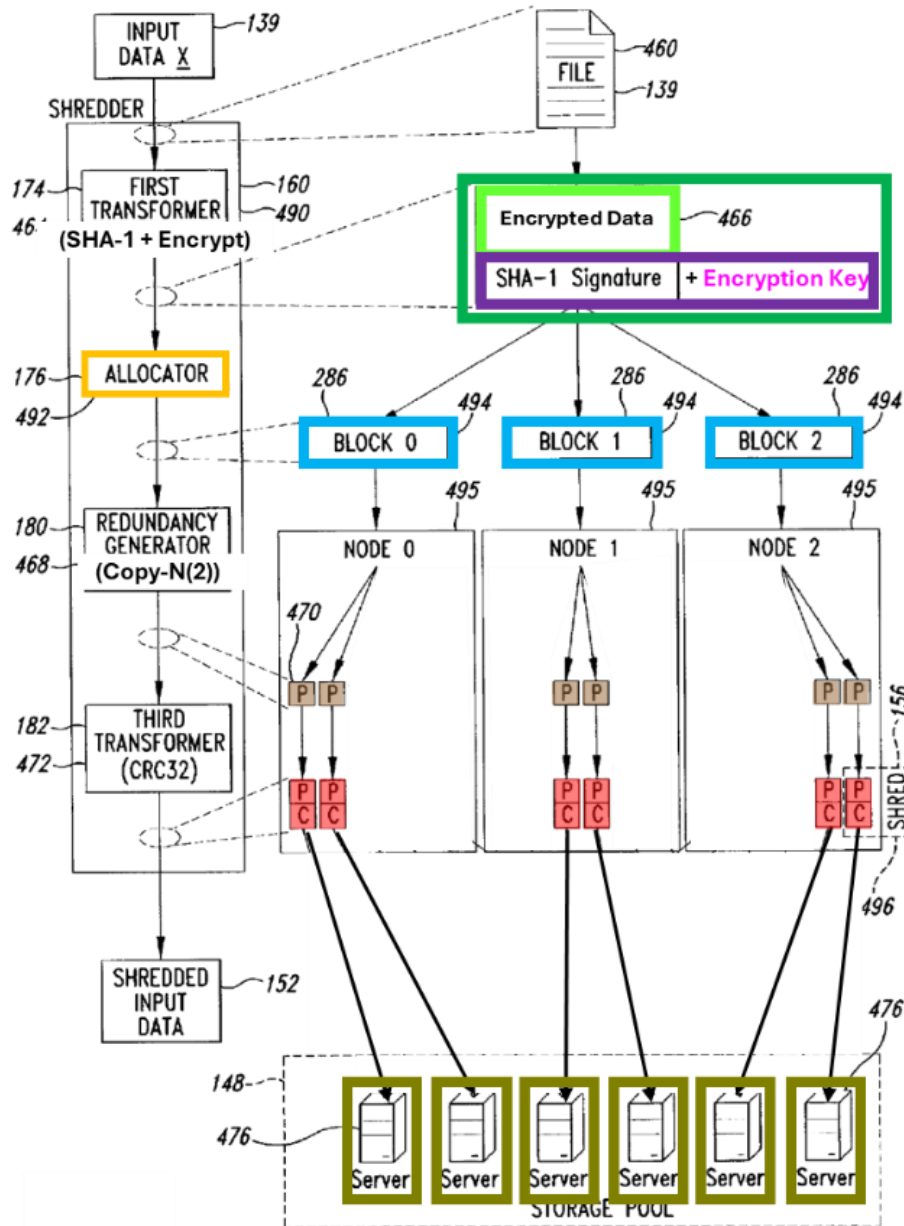
vii. [1F] causing the **plurality of shares** to be stored in **respective separate storage locations**;

286. Torre discloses claim limitation [1F].

287. Torre teaches that **shreds** can be stored in **storage devices** that are located in “*respective separate physical locations.*” Within the Torre/Tajima System, “[e]ach of the **three data blocks 494** results in **[two] shreds 156** being stored on the **[respective two] servers 476**” within a storage pool 148, as depicted below. EX1003 (Torre), [0135]. Torre further notes, “[f]or higher reliability, the storage units [such as the servers 476 in the storage pool 148] could ***be physically located separately from one another***, typically linked to the shredder by one of the interconnects 144 that is a communication network.” EX1003 (Torre), [0069]. *See also* EX1003 (Torre), [0052] (“Storage of the shredded input data ... may involve sending it ... to ***multiple storage units*** such as ... multiple hard disks on a single server or ***multiple servers***, or multiple storage devices, other than servers, directly attached to a network.”). Therefore, the Torre/Tajima System stores the **plurality of shares** on **respective servers** that are “physically located separately from one another.” EX1003 (Torre), [0069].

Torre/Tajima System

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (modified and annotated).

288. **In sum**, Torre discloses “*causing the plurality of shares* [Torre’s shreds] *to be stored in respective separate storage locations* [Torre’s servers in separate locations].”

viii. [1G] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.

289. Torre discloses claim limitation [1G].

290. Torre states that “[i]n the process of shredding, input data is generally transformed into many smaller pieces of data called shreds.” EX1003 (Torre), [0051]. It further teaches that “[s]hredding is done in a manner that adds redundancy to the shreds so that the *input data* can be recovered from a *fewer number of shreds* than the total number stored.” EX1003 (Torre), [0051].

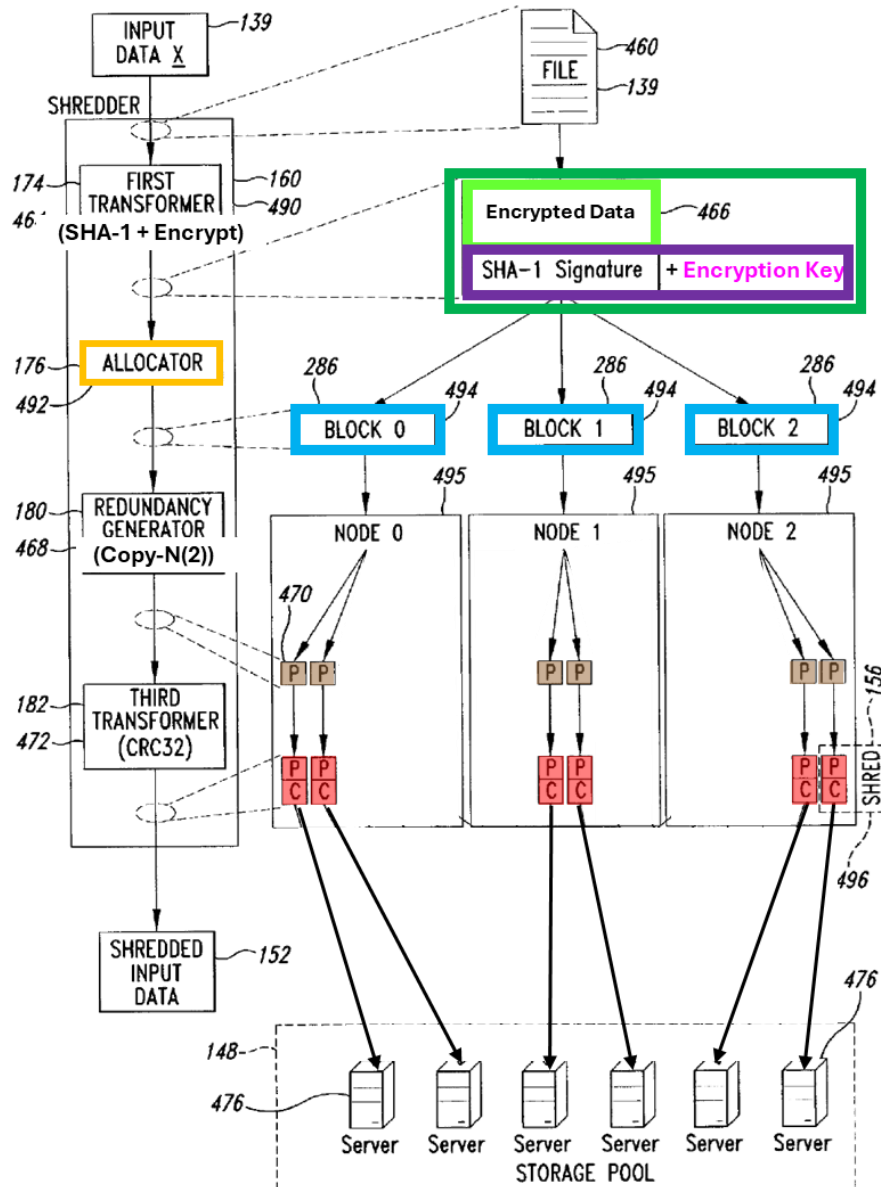
291. For example, Torre describes using a deshredder to reconstruct the original input data from the shreds. EX1003 (Torre), [0121] (“The deshredder 390 processes the received shreds 156 to output a recovered version of the input data 139, which is the same as the original input data.”). At the beginning of the deshredding process, “[t]he deshredder obtains a regeneration *subset of shreds* 156 (step 411) being *enough shreds 156 from the shredded input data 152* to be able to reconstruct the input data 139.” EX1003 (Torre), [0124]. If those obtained shreds are deemed valid, “the deshredder ... performs functions inverse to those functions

performed by the shredder” to recover the original data. EX1003 (Torre), [0121].
See also EX1003 (Torre), [0125]–[0126].

292. Thus, Torre teaches that the input data (*i.e.*, the “*data set*”) can be restored by accessing a “*subset of shreds* 156 (step 411) being *enough shreds 156 from the shredded input data 152* to be able to reconstruct the input data 139” (EX1003 (Torre), [0124]), which corresponds to “*accessing less than all, but at least a threshold number of, the plurality of shares.*” *See also* EX1003 (Torre), [0128] (“A third inverse transformer method 425, shown in FIG. 29, detects best shreds (step 426), ... [and] then reads in the chosen of the shreds 156.”). In the Torre/Tajima System illustrated below, the input data can be reconstructed by accessing 5 (a threshold number) of the 6 *shreds*.

Torre/Tajima System

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (modified and annotated).

293. Specifically, in Torre/Tajima System, the input data is restorable by accessing 5 of the 6 **shreds**. As explained in §IX.A.2 above (Torre), the redundancy

generator uses the Copy-N(2) technique to form **two copies of each unit of data** from each **data block** and the third transformer, in turn, uses the **two copies** to form **two separate shreds** that are then stored in respective **storage units**. Under this Copy-N(2) scheme, the input data would be recoverable from 5 **shreds**—which is less than the total number of **shreds** (*i.e.*, 6)—because even with only 5 **shreds**, the deshredder is guaranteed to get at least one copy of **each unit of data** from each **data block**.

294. **In sum**, Torre discloses “*wherein the data set [Torre’s input data] is restorable by accessing less than all, but at least a threshold number of, the plurality of shares* [Torre’s restorable by accessing 5 of the 6 shreds based on the Copy-N redundancy scheme].”

6. Claim 2: The method of claim 1, wherein the step of separating the encrypted data set comprises using a deterministic technique.

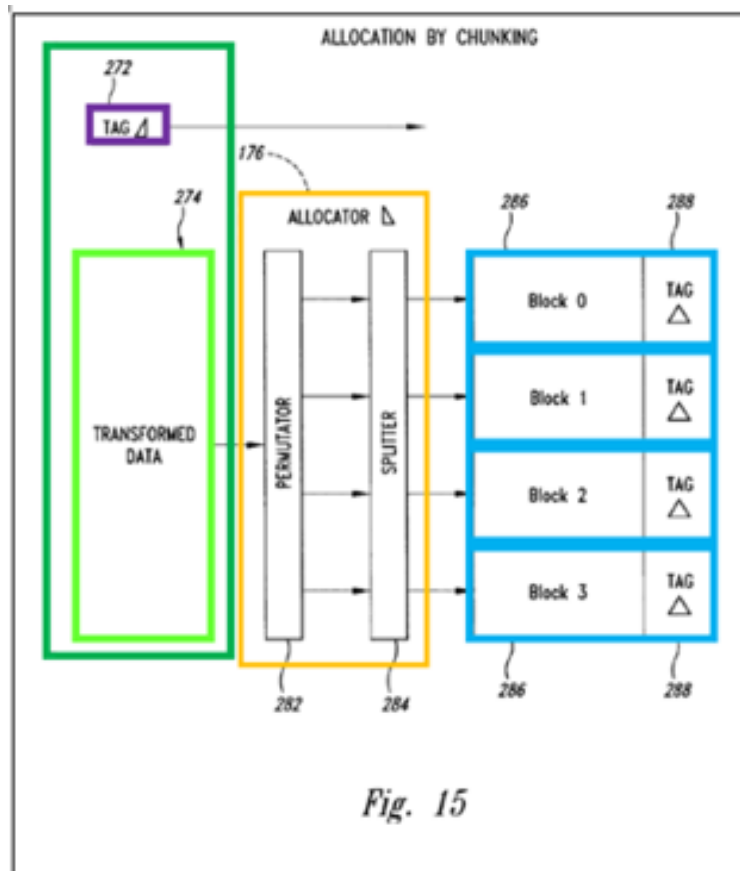
i. Torre/Tajima System-1

295. Torre discloses the additional limitation of Claim [2]. In the Torre/Tajima System-1, the allocator applies Torre’s partitioning techniques to divide the **pre-shred** into multiple **data blocks**. EX1003 (Torre), [0099]–[0102]. See §§IX.A.4.i above and IX.A.4.ii above.

296. As an initial matter, a POSITA would have known that for a technique to be deterministic, it has to follow the principle of “determinism”: “[i]n computing, the ability to predict an outcome or to know in advance how data will be manipulated

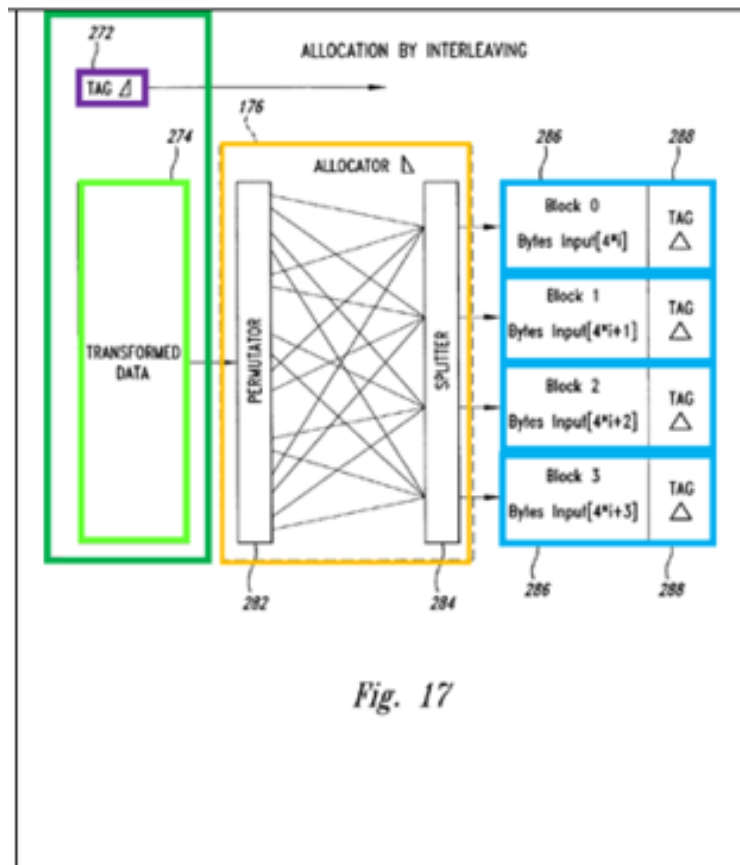
by a processing system. A deterministic simulation, for example, is one in which a certain input always produces the same output.” EX1016 (Microsoft Computer Dictionary), 154.

297. One example of a partitioning technique used in Torre is chunking. As illustrated in Figure 15 (annotated below), Torre describes “[a]n exemplary depiction of chunking ... in which the permutator 282 uses a pass-through re-map, which sequentially grabs blocks of data and passes the sequential blocks of data through the permutator without altering the order of the data.” EX1003 (Torre), [0104]. Following this, “[t]he splitter 284 then splits the sequential blocks of data into the data blocks 286.” EX1003 (Torre), [0104]. *See also* EX1003 (Torre), [0105], Figure 16. This is shown in the figure below, annotated using the same color-coding scheme as that described in the table in §IX.A.1 above.



EX1003 (Torre), Figure 15.

298. Another partitioning approach disclosed in Torre is interleaving. As shown in Figure 17 (annotated below), Torre describes “[a]n exemplary depiction of interleaving ... in which the permutator 282 uses an interleaving re-map, which in some implementations is round-robin distributing of the data at the bit or byte level into blocks of data and passes the distributed blocks of data through the permutator on to the splitter 284.” EX1003 (Torre), [0107]. This is shown in the figure below, annotated using the same color-coding scheme as that described in the table in §IX.A.1 above.



EX1003 (Torre), Figure 17.

299. Both chunking and interleaving, as described in Torre, are “*deterministic technique[s]*” for dividing the **encrypted data** (“*encrypted data set*”) into **data blocks**. These methods assign each unit of data in the **encrypted data** to a **data block** based on its relative position within the input—meaning the mapping is predetermined and not random.

300. In fact, the ’802 Patent explicitly identifies chunking and interleaving as examples of a “deterministic algorithm.” See EX1001 (’802 Patent), 72:65–73:4 (“[A]ny suitable deterministic algorithm for parsing and splitting data may be used.

For example, breaking up the original data into sequential shares [*i.e.*, chunking] may be employed as a parsing and splitting algorithm. Another example is to parse and split the original data bit by bit, appending each respective bit to the data shares sequentially in a round-robin manner [*i.e.*, interleaving].”).

301. **In sum**, Torre discloses “*wherein the step of separating the encrypted data set comprises using a deterministic technique* [Torre’s chunking or interleaving technique].”

ii. Torre/Tajima System-2

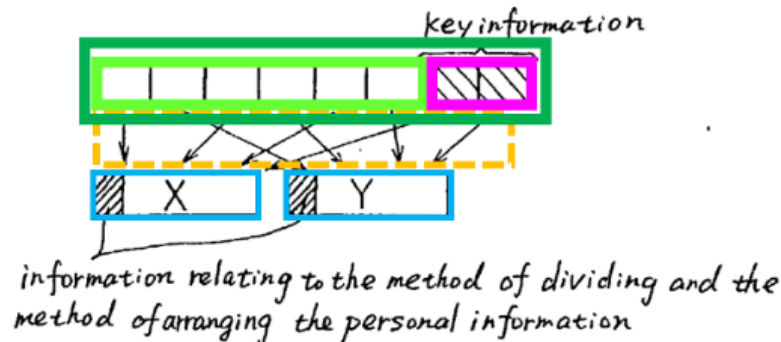
302. Torre in view of Tajima teaches the additional limitation of Claim [2].

303. As discussed in §IX.A.4.ii above [Torre/Tajima System-2], the allocator in Torre/Tajima System-2 identifies “arbitrary length[s]” and uses those values to divide the collection of transformed data (*i.e.*, the “encrypted data set”) and the encryption key into multiple data portions. EX1004 (Tajima), [0106], [0109]. These data portions are then distributed by the allocator to different data blocks in a **round-robin manner**, as illustrated in Tajima’s Figure 11 (annotated with the color-coding scheme described in §IX.A.1 above).

304. A POSITA would have known that “round robin” is a deterministic technique because it is “[a] sequential, cyclical allocation of resources to more than one process or device.” EX1016 (Microsoft Computer Dictionary), 458. As a result, Torre/Tajima System-2 partitions the encrypted data (*i.e.*, “encrypted data set”) by

assigning the portions in a round-robin pattern—i.e., “using a deterministic technique.”

Fig. 11



EX1004 (Tajima), Figure 11 (annotated).

305. **In sum**, Torre in combination with Tajima teaches “wherein the step of separating the encrypted data set comprises using a deterministic technique [Tajima’s round robin].”

7. Claims 3 and 5

- i. [3] The method of claim 1, wherein the step of separating the encrypted data set comprises using a substantially random technique.

[5] The method of claim 1, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially randomly [sic] distribution of the encrypted data set.

306. Torre/Tajima System-2 satisfies Claims [3] and [5] because Torre in view of Tajima discloses the additional limitations of Claims [3] and [5].

307. As an initial matter, I note that the term “encrypted data” term appearing first in Claim 5 lacks antecedent basis. For purposes of this IPR only, I have been asked to interpret “encrypted data” to mean “encrypted data *set*.”

308. As detailed in §IX.A.4.ii above [Torre/Tajima System-2], the **allocator** in Torre/Tajima System-2 determines “***arbitrary length[s]*** according to a function of, for example, **random numbers**,” and then uses these lengths to divide the **collection** of the **transformed data** (*i.e.*, “*encrypted data set*”) and the encryption key into multiple data portions. These portions are subsequently assigned to various **data blocks**. EX1004 (Tajima), [0106], [0109].

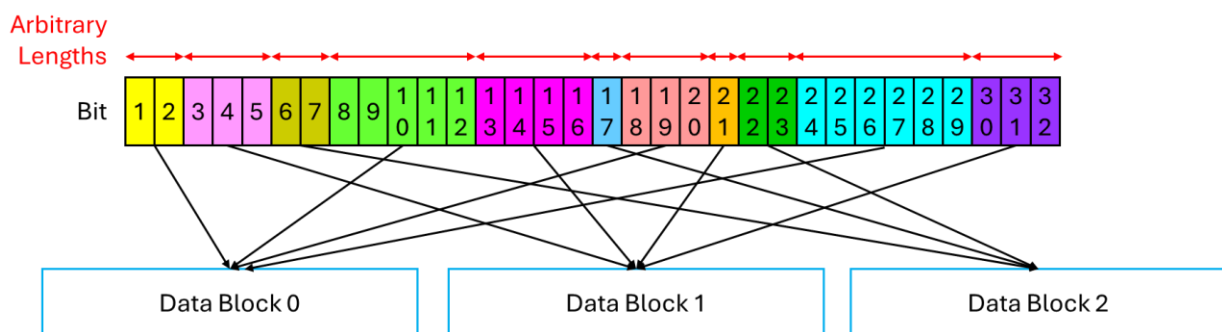
309. Accordingly, the **allocator** divides the **transformed data** (*i.e.*, “*encrypted data set*”) into **data blocks** using a “*substantially random technique*” based on “random numbers,” thereby satisfying the additional limitation of Claim [3].

310. **In sum**, for Claim 3, Torre in combination with Tajima discloses “*wherein the step of separating the encrypted data set* [Torre’s **transformed data**] *comprises using a substantially random technique* [Tajima’s technique of using generating and using arbitrary lengths based on random numbers].”

311. The **allocator** also satisfies the additional limitation of Claim [5]. Because the arbitrary lengths—which correspond to the sizes of the (plurality of) data portions—are determined using “random numbers”; each bit from the **collection** (which includes the **transformed data**) is ***randomly assigned*** to data portions, and

these (plurality of) data portions are then distributed into **data blocks**. Therefore, each bit from the **collection** (which includes the **transformed data**) is randomly assigned to the (plurality of) **data blocks**.

312. A POSITA would have understood this process of randomly assigning each bit to a portion, followed by assigning each portion to a **data block**, as illustrated below in a figure I created. For instance, each black outline box represents a bit from the 32-bit **collection** (bits 1 ... 32); bit-boxes sharing the same color denote bits grouped into a data portion determined by an arbitrary length; and the data portions are assigned to data blocks 0-2 in a round-robin fashion, consistent with Tajima's Figure 11. As noted in §IX.A.5.iv above for claim limitation [1C], the **data blocks** are subsequently divided into multiple **shreds** (which is not shown below for brevity).



313. Because each bit from the **collection** (which includes the **transformed data**) is randomly assigned to one of the data portions—which are then interleaved across multiple **data blocks** and ultimately into multiple **shreds** (*i.e.*, the “*plurality of*

shares”)—the resulting **shreds** exhibit “*a substantially randomly [sic] distribution of the encrypted data set*,” thereby satisfying the additional limitation of Claim [5].

314. **In sum**, for Claim 5, Torre in combination with Tajima discloses “*wherein the step of separating the encrypted data [Torre’s transformed data] into the plurality of shares [Torre’s shreds] comprises causing the plurality of shares [Torre’s shreds] to have a substantially randomly [sic] distribution of the encrypted data set [using Tajima’s technique of using generating and using arbitrary lengths based on random numbers].*”

8. Claim 9: The method of claim 1, wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices.

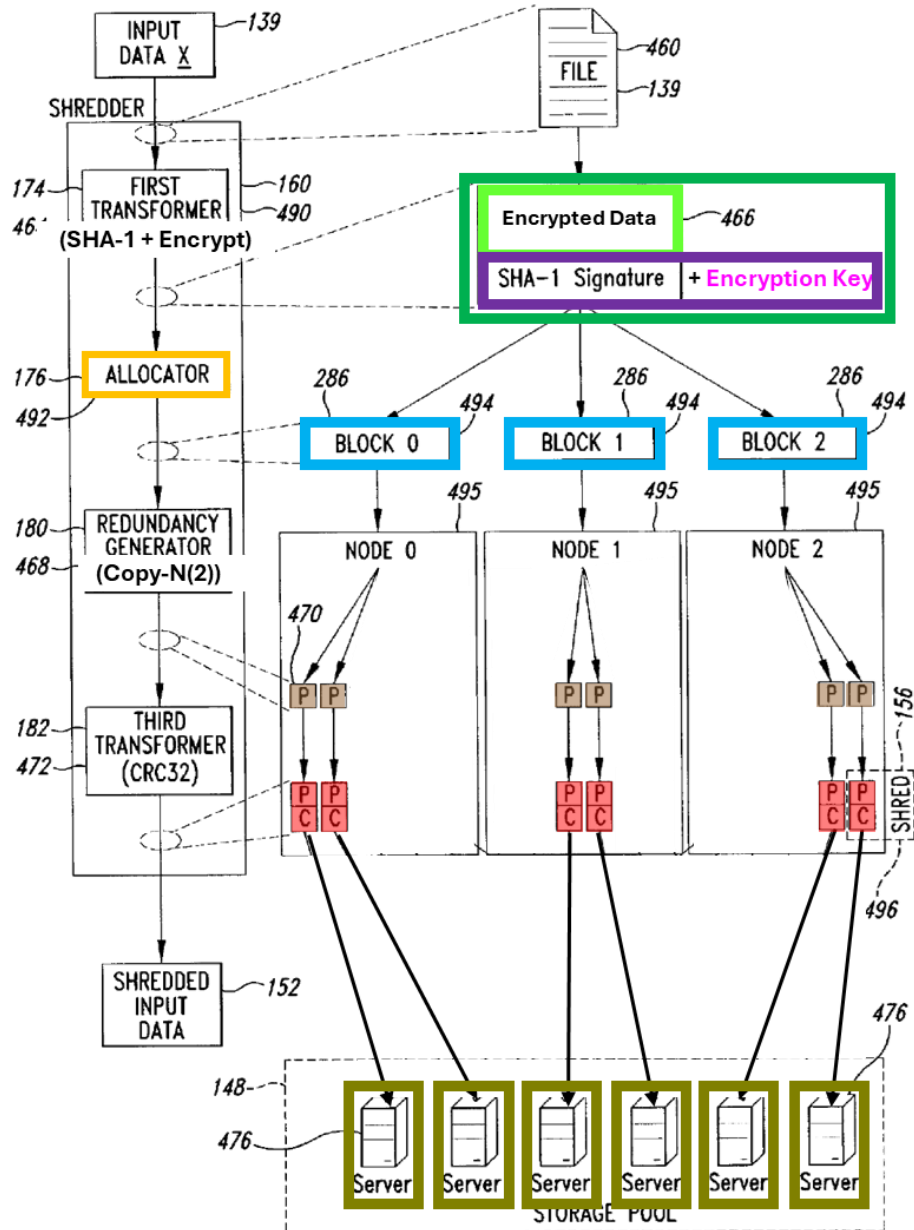
315. Torre discloses the additional limitation of Claim [9].

316. As discussed in §IX.A.5.vii above for claim limitation [1F], the **multiple shreds** are stored in their **respective storage units**, which may be located in “*separate physical locations.*”

317. In the Torre/Tajima System, as illustrated in Torre’s annotated FIG. 34 below, the **shreds** (*i.e.*, “*shares*”) are stored across at least **two separate servers**. EX1003 (Torre), [0135]. *See also* EX1003 (Torre), [0052] (“Storage of the shredded input data ... may involve sending it ... to **multiple storage units** such as ... multiple hard disks on a single server or **multiple servers**, or multiple storage devices, other than servers, directly attached to a network.”).

Torre/Tajima System

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (annotated).

318. **In sum**, Torre discloses “*wherein the step of storing the **shares** [Torre’s shreds] in respective separate storage locations comprises storing the **shares** [Torre’s shreds] on at least two **separate storage devices** [Torre’s servers].*”

9. **Claim 10: A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.**

319. Claim 10 is identical to claim 1 except for claim preamble [10Pre].

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

Claim 1	Claim 10
[1A]	[10A]
[1B]	[10B]
[1C]	[10C]
[1D]	[10D]
[1E]	[10E]
[1F]	[10F]
[1G]	[10G]

321. Therefore, for the same reasons discussed in §§IX.A.5.ii above—IX.A.5.viii above regarding claim limitations [1A]–[1G], the Torre/Tajima System also satisfies claim limitations [10A]–[10G], respectively.

- i. **[10Pre] A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of**

322. To the extent the preamble is limiting, Torre discloses claim preamble [10Pre].

323. As discussed in §IX.A.5.i above for claim preamble [1Pre], Torre discloses “*a method for securing a data set.*”

324. Torre also discloses that the described method is implemented as “[*a non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out*]” the method. For instance, Torre explains that its teachings “can be implemented, individually and/or collectively, by a wide range of hardware, **software**, firmware, or virtually any combination thereof.” EX1003 (Torre), [0141]. The software is implemented as a “code” (*i.e.*, “*computer executable instructions*”) that, “*when executed by at least one processor, cause[s] a computer system*” to execute the method. See EX1003 (Torre), [0141] (explaining that its teachings are implemented

“as one or more *computer programs* running on *one or more computers* ..., as one or more *programs* running on *one or more processors e.g., microprocessors*, as firmware, or as virtually any combination thereof”); [0142]; [0064]. Torre further states that such code may be stored in “signal bearing media” including “**floppy disks, hard disk drives, CD ROMs, digital tape, and computer memory,**” thereby showing “[a] *non-transitory computer readable medium.*” EX1003 (Torre), [0141]; [0064].

325. Therefore, claim 10 would have been obvious in view of the Torre/Tajima System.

326. **In sum**, Torre discloses “[a] *non-transitory computer readable medium* [Torre’s signal bearing media] *storing computer executable instructions* [Torre’s computer programs] *that, when executed by at least one processor* [Torre’s one or more processors], *cause a computer system* [Torre’s one or more computers] *to carry out a method for securing a data set* [Torre’s method for shredding and deshredding input data for secure storage].”

10. **Claim 19: A computer system for securing a data set, the system comprising: at least one processor; a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.**

327. Claim 19 is substantively the same as claim 1, except for claim preamble [19Pre] and claim limitations [19A]–[19B], discussed next.

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

Claim 1	Claim 19
[1A]	[19C]
[1B]	[19D]
[1C]	[19E]
[1D]	[19F]
[1E]	[19G]
[1F]	[19H]
[1G]	[19I]

329. Therefore, for the same reasons discussed in §§IX.A.5.ii-IX.A.5.viii regarding claim limitations [1A]–[1G], the Torre/Tajima System also satisfies claim limitations [19C]–[19I], respectively.

i. [19Pre] A computer system for securing a data set, the system comprising:

330. To the extent the preamble is limiting, Torre discloses claim preamble [19Pre]. As explained in §IX.A.5.i above for claim preamble [1Pre], Torre discloses a method “*for securing a data set.*”

331. Torre also teaches that the disclosed method is executed by a “*computer system.*” EX1003 (Torre), [0141] (explaining that its teachings are implemented “as one or more *computer programs running on one or more computers* (e.g., as one or more programs running on one or more *data processing systems*), ..., as one or more *programs running on one or more processors* e.g., microprocessors, as firmware, or as virtually any combination thereof.”); [0142]; [0064].

332. **In sum**, Torre discloses “[*a*] *computer system* [Torre’s one or more computers] *for securing a data set* [Torre’s method for shredding and deshredding input data for secure storage].”

ii. [19A] at least one processor;

333. Torre also teaches that the disclosed method is executed by the “*computer system*” that includes “*at least one processor.*” See EX1003 (Torre),

[0141] (explaining that its teachings are implemented “as one or more *computer programs running on one or more computers* (e.g., as one or more programs running on one or more *data processing systems*), ..., as one or more *programs running on one or more processors* e.g., microprocessors, as firmware, or as virtually any combination thereof.”), [0142], [0064].

334. **In sum**, Torre discloses “*at least one processor* [Torre’s one or more processors].”

- iii. **[19B] a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of:**

335. Torre/Tajima System satisfies claim limitation [19B] for the same reasons discussed in §IX.A.9.i above for claim preamble [10Pre]. *See* §IX.A.9.i above for claim preamble [10Pre].

336. Accordingly, the Torre/Tajima System renders claim 19 obvious.

11. Claims 11-12, 14, 18, 20-21, 23, and 27

337. The Torre/Tajima System renders claims 11-12, 14, 18, 20-21, 23, and 27 obvious.

- The Torre/Tajima System satisfies the additional limitation of claims 11 and 20 for the same reasons as discussed in §IX.A.6 above for [Claim 2];

- Torre/Tajima System-2 satisfies the additional limitation of claims 12, and 21 for the same reasons discussed in §IX.A.7 above for [Claim 3];
- Torre/Tajima System-2 satisfies the additional limitation of claims 14 and 23 for the same reasons discussed in §IX.A.7 above for [Claim 5]; and
- The Torre/Tajima System satisfies the additional limitation of claims 18 and 27 for the same reasons discussed in §IX.A.8 above for [Claim 9].

B. Ground II: Torre, Tajima, And Aldridge Render Obvious Claims 4, 13, and 22.

338. Claims 4, 13, and 22 would have been obvious in view of Torre, Tajima, and Aldridge.

339. In this Ground, I introduce illustrations to show how the Torre/Tajima/Aldridge combination renders Claims 4, 13, and 22 obvious.

1. Torre/Tajima/Aldridge Color-Coding Scheme

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

Color	Disclosure in Torre, Tajima, and/or Aldridge
violet	Torre's SHA-1 digital signature information
pink	Torre's encryption operation
lime green	Torre's encrypted data set
	Tajima's encrypted data
red	Torre's shreds
light blue	Torre's intermediate data blocks
	Tajima's data fragments
magenta	Tajima's encryption key

Color	Disclosure in Torre, Tajima, and/or Aldridge
olive	Torre's storage servers
brown	Torre's data unit from data blocks
yellow	Torre's allocator Tajima's allocation method
coral	Aldridge's scrambler
forest green	Collection of the encrypted data, SHA-1 digital signature information, and/or encryption key

2. Overview of Aldridge (EX1005)

341. Aldridge is directed to the “storage and transfer of large datasets with high security across existing networks.” EX1005 (Aldridge), [0013]. It explains that “[s]ecurity is achieved through two steps. In the first aspect, data to be transferred are obscured by scrambling the datasets, and then splitting them into virtually unidentifiable file segments of uniform size and characteristics.” EX1005 (Aldridge), [0013].

342. Figure 2 in Aldridge (below) illustrates this process. EX1005 (Aldridge), [0049]. Initially, “original data source 1 comprised of files and subdirectories 2-7” is “split (‘sliced’)” into “segments of fixed length 8-13.” EX1005 (Aldridge), [0056]. These segments are then grouped—“Adjacent segments may be then grouped in multiples of n (n=3 in the example in FIG. 2)” — and “the data in each group of n adjacent segments may then be ‘scrambled’ resulting in segments 14-19 of equal length.” EX1005 (Aldridge), [0057]. “‘Scrambling’ may

involve any process which obscures data, including but not limited to encryption, byte shuffling, byte and/or bit rotation.” EX1005 (Aldridge), [0057].

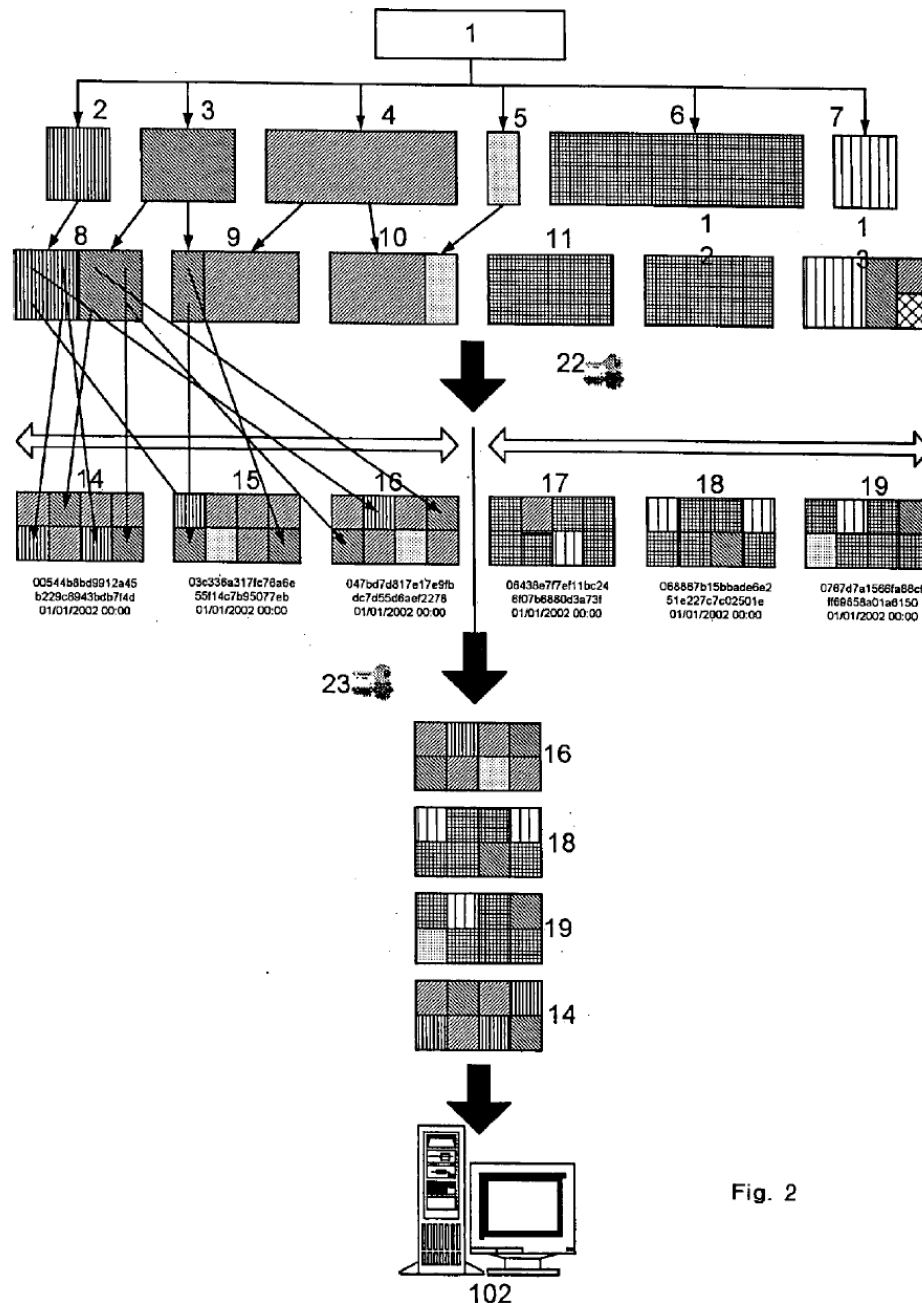


Fig. 2

EX1005 (Aldridge), Figure 2.

343. Aldridge states that the grouping parameter n “is less than or equal to the total number of segments.” EX1005 (Aldridge), [0057]. It further explains that reducing “the number of segments over which the data from a file is distributed” (*i.e.*, the grouping parameter n) can be advantageous, particularly when the scrambled files are stored on a recipient’s device, as doing so “reduces the time required to reconstruct a file from the scrambled data by reducing the amount of data that must be accessed.” EX1005 (Aldridge), [0059]. Thus, a POSITA would have understood that n could be set to 1, or at a minimum, would have found it obvious to choose $n=1$ to achieve the stated benefit of minimizing data distribution across segments. EX1005 (Aldridge), [0059].

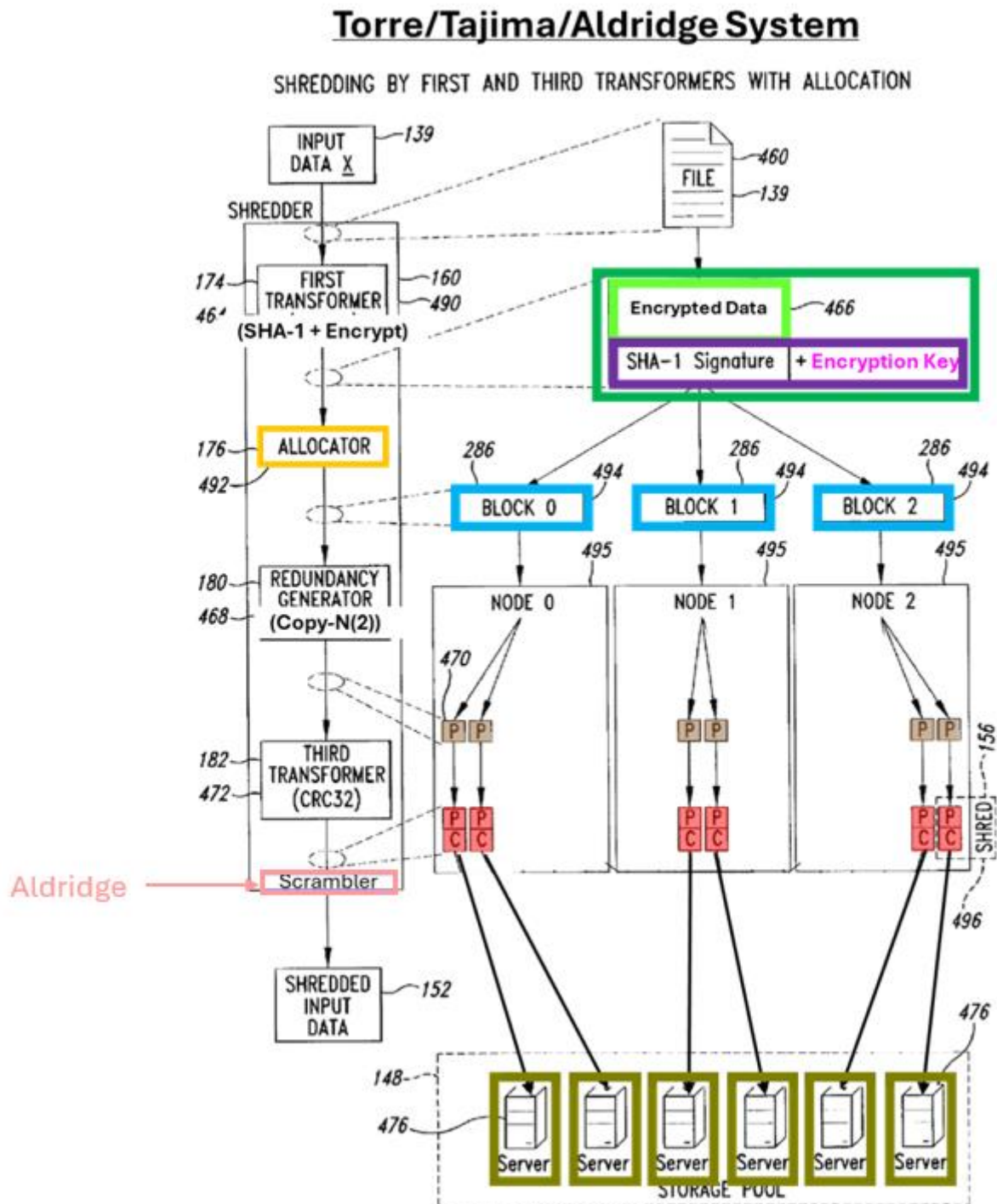
3. Torre/Tajima With Aldridge (“Torre/Tajima/Aldridge”)

344. The Torre/Tajima System is designed to securely store input data by parsing and splitting it into shreds. *See* §IX.A.4 above [Torre/Tajima].

345. It would have been obvious to enhance the Torre/Tajima System by incorporating Aldridge’s data scrambling technique to further strengthen data security. In particular, applying Aldridge’s scrambling approach to scramble individual **shreds** would have been a straightforward adaptation aimed at increasing resistance to unauthorized reconstruction of the original data.

346. The combination of the Torre/Tajima System with Aldridge’s data scrambling technique is depicted in the figure below, illustrating Aldridge’s data

scrambling step using a coral scrambler block. The annotations of this figure use the same color-coding scheme as that described in the table in §IX.B.1 above.



EX1003 (Torre), FIG. 34 (modified and annotated).

a. Reasons To Combine Torre/Tajima With Aldridge

347. A POSITA would have been motivated to incorporate Aldridge's data scrambling technique into the Torre/Tajima System for at least the following two reasons.

348. **First**, Aldridge states that a scrambled data segment "is obscured in a way that makes it very difficult to reconstruct without using the correct protocols" and that this enhances "security." EX1005 (Aldridge), [0032]. *See also* EX1005 (Aldridge), [0005] ("According to a first aspect of the present invention, a method of *preparing a dataset for secure transmission* over a communications network comprises the steps of *scrambling the dataset according to a first key*, splitting the dataset into a plurality of blocks and ordering the blocks according to a second key."), [0013] ("The present invention enables storage and transfer of large datasets with *high security* across existing networks, including the public internet. Security is achieved through two steps. In the first aspect, data to be transferred are *obscured by scrambling the datasets*, and then *splitting them* into virtually unidentifiable file segments of uniform size and characteristics."), [0031] ("The invention provides a highly secure data transfer mechanism that can be used on the public internet without revealing the data.").

349. Aldridge explicitly teaches scrambling data within a portion of the input—such as a **shred** in the Torre/Tajima System—to achieve stronger data protection. EX1005 (Aldridge), [0056]–[0057] (“The data in each group of n adjacent segments may then be ‘scrambled’ resulting in segments 14-19 of equal length, which may be the same length as the original segments. Thus segments 14, 15 and 16 are the result of scrambling segments 8, 9 and 10, and segments 17, 18 and 19 are the result of scrambling segments 11,12, and 13. Each one of the new segments 14, 15 and 16 includes data from all of the segments 8, 9 and 10 in a way that ensures that it is not obvious how to recreate the original data set. ‘Scrambling’ may involve any process which obscures data, including but not limited to encryption, byte shuffling, byte and/or bit rotation.”), [0059] (“The advantage of this method of scrambling segments with their $n-1$ neighbours is that it *limits the number of segments over which the data from a file is distributed*, and this has advantage if the files are to be held in obscured format on the recipient’s device as it reduces the time required to reconstruct a file from the scrambled data by reducing the amount of data that must be accessed. Also if only one file need be transferred across the network from a source directory, this reduces the number of segments that need be transferred to obtain the complete file.”); *see also* §IX.B.2 above [Aldridge].

350. Accordingly, a POSITA would have been motivated to implement a known technique (Aldridge’s scrambling technique) to improve security of a known

device (Torre/Tajima System) in the same way (scrambling data within a **shred** (EX1005 (Aldridge), [0059])) to obtain predictable results of making it “very difficult to reconstruct” the input data.

351. **Second**, the Torre/Tajima system already randomizes the data into multiple locations, as discussed for Claim [3] in §IX.A.7 above. Aldridge merely adds a stronger form of randomization (*i.e.*, scrambling) to improve security even further. A POSITA would have recognized this modification to be simple yet beneficial.

352. Indeed, a POSITA would have recognized that scrambling techniques are common in encryption systems. For example, “CSS” is “[] 2. Acronym for Content Scrambling System. An encryption feature added to DVDs distributed with approval of the MPAA. CSS looks for a matching region code on the DVD and the playback device. If the codes do not match (such as for a DVD purchased in Japan and a DVD player purchased in the United States), CSS will not allow the DVD to play. CSS also will not allow a DVD to be played on playback equipment not approved by the MPAA. *See also* deCSS, region code.” EX1016 (Microsoft Computer Dictionary), 126, 136. *See also* §V.C above.

353. Next, a POSITA would have had a reasonable expectation of success in applying Aldridge’s data scrambling technique within the Torre/Tajima System for at least the following two reasons.

354. **First**, Aldridge states that “[t]here is much prior art which may be described as ‘scrambling’” and that “[t]his will be obvious to a person skilled in the art.” EX1005 (Aldridge), [0055]. Moreover, Aldridge teaches that “[s]crambling” may involve any process which obscures data, including but not limited to encryption, byte shuffling, byte and/or bit rotation.” EX1005 (Aldridge), [0057].

355. **Second**, implementing such a technique in Torre/Tajima would have involved rearranging data units within each shred—a task well within the skill of the art—using either software or hardware-based solutions. As a result, this combination would have predictably achieved the desired outcome of enhancing security by shuffling data within each shred.

4. Claim 4: The method of claim 1, further comprising causing a plurality of data units in each of the **shares to be rearranged relative to one another after the separating step.**

356. Torre/Tajima in combination with Aldridge renders obvious the additional limitation of claim [4]. As discussed in §IX.B.3 above [Torre/Tajima/Aldridge], the Torre/Tajima/Aldridge System results in the rearrangement of bytes or bits (“*data units*”) within each shred (“*share*”), consistent with the scrambling technique illustrated in Aldridge’s Figure 2 (below). EX1005 (Aldridge), [0057] (“‘Scrambling’ may involve any process which obscures data, including but not limited to encryption, *byte* shuffling, *byte and/or bit rotation*.”).

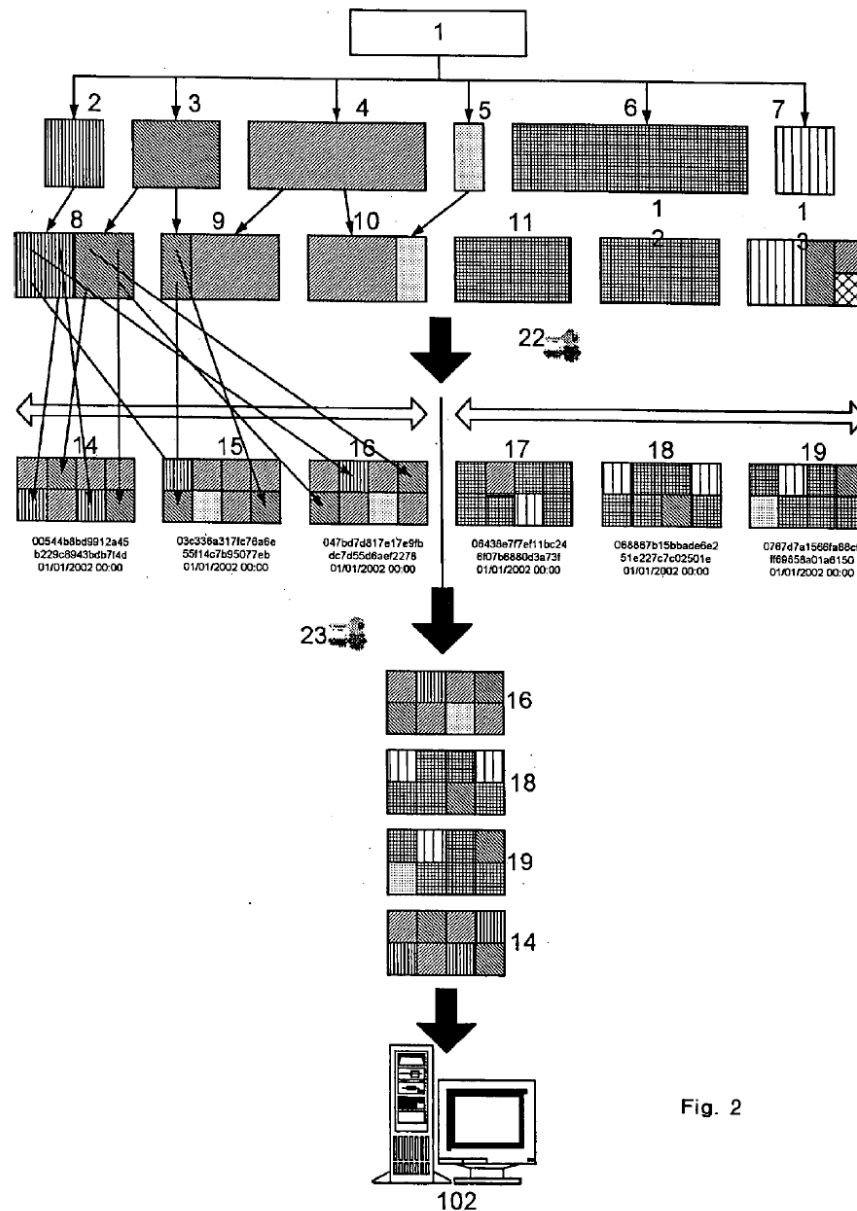
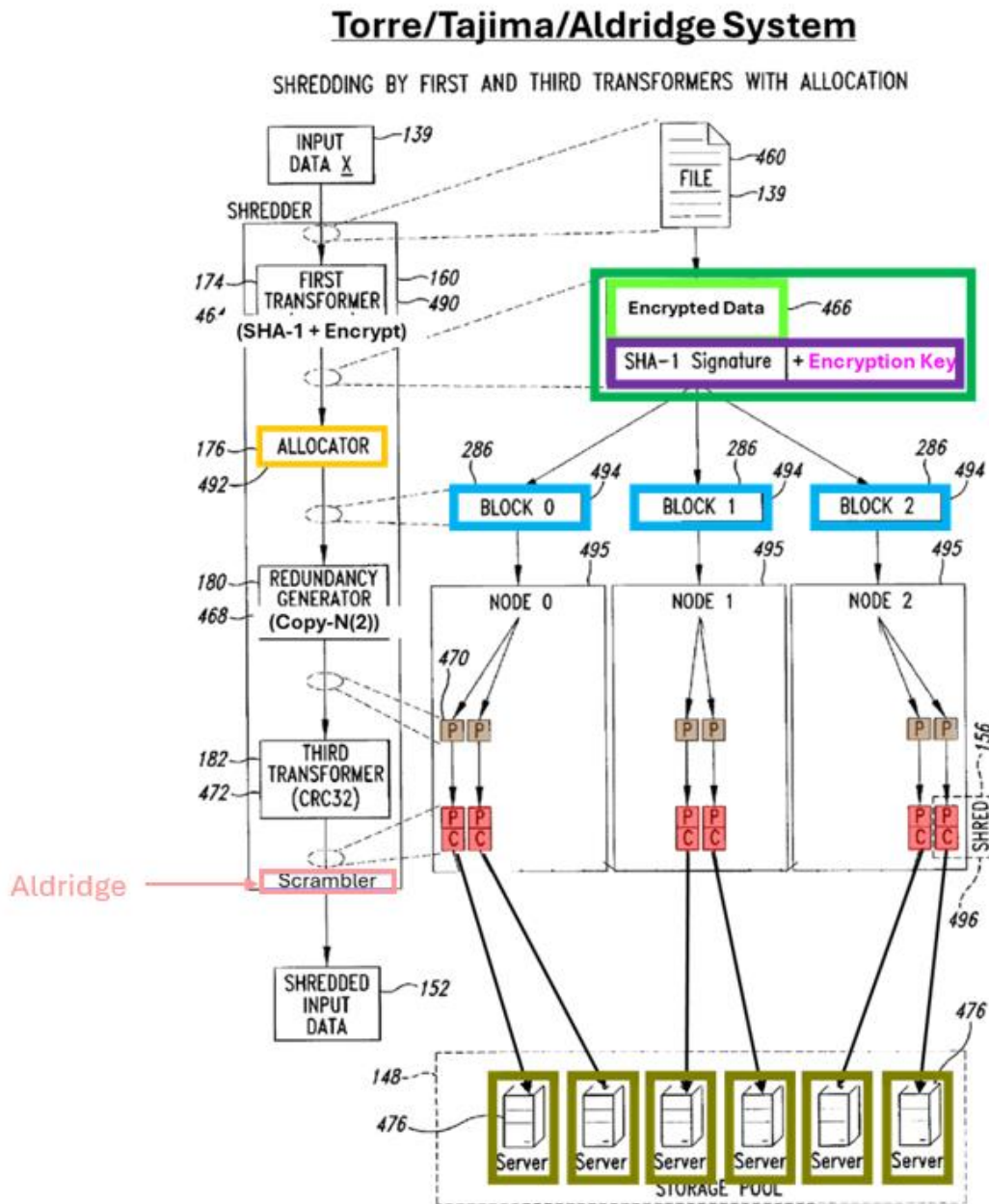


Fig. 2

EX1005 (Aldridge), Figure 2.

357. In this combined system, the scrambling operation is carried out after the separating step that produces the **shreds**, as explained in §IX.A.5.v above for claim limitation [1D]. For example, as shown in the annotated Figure 34 of Torre (introduced in §IX.B.3 above), once the third transformer forms the shreds 156

(“plurality of shares”), the scrambling operation is carried out thereafter, as shown by the coral box. This is shown in the figure below, annotated to use the same color-coding scheme as that described in the table of §IX.B.1 above.



EX1003 (Torre), FIG. 34 (modified and annotated).

358. **In sum**, Torre/Tajima in combination with Aldridge teaches “*causing a plurality of data units in each of the shares* [Torre’s shreds] *to be rearranged relative to one another after the separating step* [using Aldridge’s scrambling technique].”

5. **Claim 13: The non-transitory computer readable medium of claim 10, wherein the method further comprises causing a plurality of data units of the encrypted data set to be rearranged relative to one another after the separating step.**

Claim 22: The system of claim 19, wherein the method further comprises causing a plurality of data units of the encrypted data set to be rearranged relative to one another after the separating step.

359. The additional limitation in claims 13 and 22 is identical to that in claim 4, except that claim 4 requires “causing a plurality of data units in *each of the shares* to be rearranged,” whereas claims 13 and 22 specify “causing a plurality of data units *of the encrypted data set* to be rearranged.”

360. As discussed in §IX.A.5.iv above for claim limitation [1C], each **shred** (“*share*”) contains data units from the **encrypted data set** 466.

361. Therefore, by rearranging data units within each share—as described above in §IX.B.4 above [Claim 4]—the Torre/Tajima/Aldridge combination also teaches rearranging “a plurality of data units *of the encrypted data set*.”

362. **In sum**, Torre/Tajima in combination with Aldridge teaches “*causing a plurality of data units of the encrypted data set* [Torre’s encrypted data set 466] *to*

be rearranged relative to one another after the separating step [using Aldridge's scrambling technique]."

C. Ground III: Torre, Tajima, And Krawczyk Render Obvious Claims 6, 15, and 24.

363. Claims 6, 15, and 24 would have been obvious in view of Torre, Tajima and Krawczyk.

364. In this Ground, I introduce illustrations to show how the Torre/Tajima/Krawczyk combination renders Claims 6, 15, and 24 obvious.

1. Torre/Tajima/Krawczyk Color-Coding Scheme

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

Color	Disclosures in Torre, Tajima or Krawczyk
violet	Torre's SHA-1 digital signature information
pink	Torre's encryption operation
lime green	Torre's encrypted data set
	Tajima's encrypted data
royal blue	Torre: selection information for selecting between allocation modes
	Tajima: arbitrary length information
red	Torre's shreds
light blue	Torre's intermediate data blocks
	Tajima's data fragments
magenta	Tajima's encryption key
olive	Torre's storage servers
brown	Torre's data unit from data blocks
yellow	Torre's allocator
	Tajima's allocation method

Color	Disclosures in Torre, Tajima or Krawczyk
forest green	Collection of the encrypted data, SHA-1 digital signature information, and/or encryption key
gray	Krawczyk's "Shamir's secret sharing algorithm"

2. Overview of Krawczyk (EX1006)

366. Krawczyk is directed to a "secret sharing" technique that integrates "an information dispersal scheme with a secure encryption scheme and a perfect (e.g. Shamir's) secret sharing scheme." EX1006 (Krawczyk), 8, 12. This approach distributes a secret S as n shares, such that the reconstruction of the secret requires any m out of those n shares. EX1006 (Krawczyk), 12–13. The method described in Krawczyk proceeds as follows:

- "Choose a random encryption key K [and] [e]ncrypt the secret S ," which produces an encrypted secret E .
- Use an information dispersal scheme to split the encrypted secret E into n fragments: $E_1, \dots, E_n$.
- Apply "a perfect (n, m) -secret sharing scheme (e.g. Shamir's)" in order to "generate n shares for the key K ": $K_1, \dots, K_n$.
- Distribute to each of the n participants their respective share, defined as $S_i = (E_i, K_i)$.

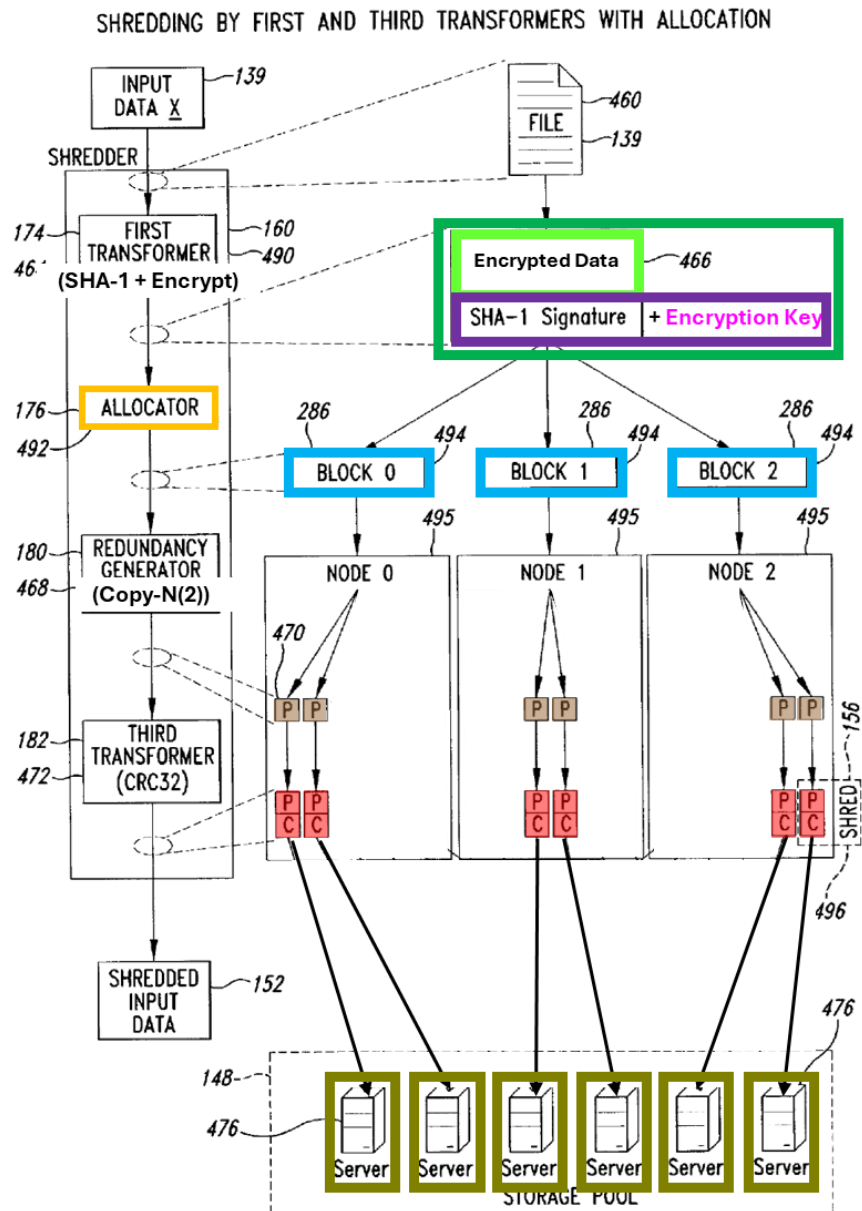
EX1006 (Krawczyk), 13.

367. To reconstruct the original secret S , Krawczyk's method involves collecting m shares S_i . From these m shares, the encrypted secret E and the encryption key K are recovered. The secret S is then obtained by decrypting E using the recovered key K . EX1006 (Krawczyk), 13.

3. Torre/Tajima In Combination With Krawczyk ("Torre/Tajima/Krawczyk")

368. In the Torre/Tajima System, when Torre's first transformer performs encryption, it generates a tag that includes both the **encryption key** and the SHA-1 signature, as illustrated in Figure 34 of Torre as modified, below. *See* §IX.A.4 above [Torre/Tajima].

Torre/Tajima System

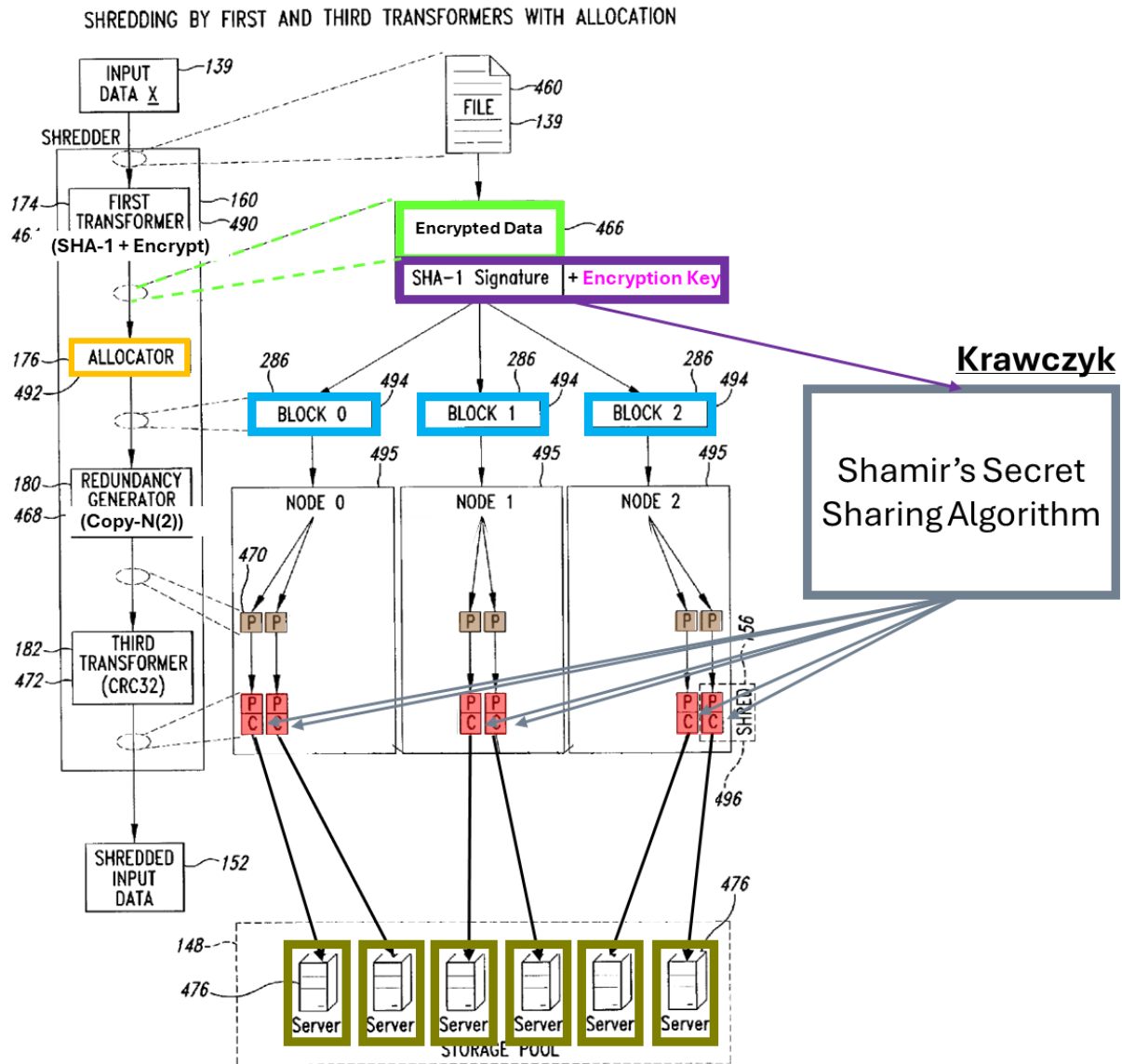


369. The system then parses and splits the encrypted data along with the tag to produce shreds that contain information indicative of both the **encryption key** and the SHA-1 signature, as discussed in §IX.A.5.iv above for claim limitation [1C] and §IX.A.5.vi above for claim limitation [1E].

370. Krawczyk teaches an alternative data splitting method for incorporating the tag (including the encryption key) into the shreds. Specifically, Krawczyk describes using “a perfect (n, m) -secret sharing scheme (e.g. Shamir’s)” to process the tag separately from the **transformed data** and then appending the outputs of the secret-sharing scheme to the **shreds**. EX1006 (Krawczyk), 12. *See* §IX.C.2 above [Krawczyk].

371. A POSITA would have found it obvious to modify the Torre/Tajima System by separately processing the tag (including the **encryption key**) using Shamir’s secret sharing method, as Krawczyk explains, and appending the resulting shares of the tag to the data shreds, as shown in Torre’s annotated FIG. 34 below. In this approach, the **shreds** would include information indicative of the encryption key. This is shown in the figure below, which is annotated using the same color-coding scheme as that described in the table in §IX.C.1 above.

Torre/Tajima/ Krawczyk System



EX1003 (Torre), FIG. 34 (modified and annotated).

372. A POSITA would have been motivated to combine Krawczyk's alternative data splitting method into the Torre/Tajima system for at least the following two reasons.

373. **First**, a POSITA would have been motivated to combine the Torre/Tajima System with Krawczyk in order to enhance security. It would have been understood that maintaining the confidentiality of the tag—especially one containing the **encryption key**—is critical to preventing unauthorized decryption of the encrypted data. *See* §V.C above. Therefore, a POSITA would have been motivated to improve the protection of the tag (or its components) when stored within the **shreds**. And a POSITA would have understood that Krawczyk teaches a solution: “*a perfect (n, m)-secret sharing scheme (e.g. Shamir’s)*” that may be used to generate “n shares for the key K” that are then appended to the n shares of the encrypted data. EX1006 (Krawczyk), 13. *See also* EX1006 (Krawczyk), 13 (“this is essential in Shamir’s scheme to provide *perfect secrecy*”).

374. **Second**, based on this teaching, a POSITA would have been motivated to apply Shamir’s secret-sharing scheme to the tag in the Torre/Tajima System to further secure the encryption key. The combination would have merely amounted to using a known technique (Shamir’s algorithm) to improve a similar device (Torre/Tajima System for securely storing input data) in the same way (applying Shamir’s algorithm to the tag to create n shares and appending the n shares of tag to n shares of encrypted data as taught by Krawczyk) to yield predictable results (*e.g.*, improving the security of the tag).

375. Next, a POSITA would have had a reasonable expectation of success in combining the Torre/Tajima System with Krawczyk for at least the following two reasons.

376. **First**, both Torre/Tajima and Krawczyk are directed to the same underlying technical challenge: securely distributing secret information. They each offer substantially similar solutions that involve splitting encrypted data into shares and distributing those shares along with information about the encryption key. *See* EX1003 (Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and scalability and adaptability to resources.*”) *with* EX1004 (Tajima), [0044] (“It is an object of the present invention to provide a personal information management system, a personal information management method, and a *server that can improve the security of personal information that is registered with areas that are connectible to the Internet.*”), [0048] (“Thus, because *personal information that is registered with areas that are connectible to the Internet is divided into a plurality of data portions and then registered with areas that are each under separate control,...*”) *and with* EX1006 (Krawczyk), 12-13 (“For completeness, we outline the following simple information dispersal scheme based on Reed Solomon erasure codes. ... We proceed to show how to build a space efficient secret sharing scheme.”).

377. Moreover, both systems incorporate redundancy to enable recovery of the original information from fewer than all available shares. *See* EX1003 (Torre), [0051] (“Shredding is done in a manner that adds redundancy to the shreds so that the input data can be recovered from a fewer number of shreds than the total number stored.”); EX1006 (Krawczyk), 12-13 (“For completeness, we outline the following simple information dispersal scheme based on Reed Solomon erasure codes. ... We proceed to show how to build a space efficient secret sharing scheme.”).

378. The combination would have merely amounted to applying a known technique (applying Shamir’s secret-sharing scheme to the tag (including the encryption key) and appending the outcome of the secret-sharing scheme to shares, as taught by Krawczyk) to a known device (Torre/Tajima System) ready for improvement (for improved security of the tag) to yield predictable results (improving the tag’s security). This modification would have been a straightforward enhancement aimed at improving the security of the tag, with a predictable and well-understood outcome.

379. **Second**, a POSITA would have understood that implementing Shamir’s known secret-sharing scheme within the Torre/Tajima System would have simply involved implementing the corresponding functionality in software, by writing program code and following Krawczyk’s algorithmic steps listed in §IX.C.2 above.

That task would have been well within the abilities of a POSITA. *See* EX1003 (Torre), [0141].

380. At a minimum, it would have been obvious to try to use Shamir's secret-sharing scheme in Torre/Tajima System. There was a design need to distribute the tag (including encryption key) securely so that the encrypted data can be decrypted later. And there were several solutions to distributing the tag: (1) provide information indicative of the tag in shreds, as taught by Torre/Tajima System-1 or Torre/Tajima System-2; or (2) append outputs of Shamir's secret-sharing scheme to shreds, as taught by Krawczyk (EX1006 (Krawczyk), 12-13). A POSITA would have had a reasonable expectation of success for each solution: for (1) as explained in §IX.A.4 above [Torre/Tajima]; and for (2) as explained above just above.

4. Claim 6: The method of claim 1, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.

381. Torre/Tajima in view of Krawczyk teaches the additional limitation of Claim 6. In the Torre/Tajima/Krawczyk combination, “a perfect (n, m)-secret sharing scheme (e.g. *Shamir's*)” is used to process the tag (including the encryption key) and append the portions of the processed tag to the shares. EX1006 (Krawczyk), 13; *see* §IX.C.3 above. As a result, the shares include “*data indicative*

of the encryption key” “created using a Shamir algorithm.” EX1006 (Krawczyk),
13.

382. **In sum**, Torre/Tajima in combination with Krawczyk teaches “*wherein the data indicative of the encryption key comprises data created using a Shamir algorithm* [Krawczyk’s teaching of using Shamir’s perfect (n, m)-secret sharing scheme].”

5. Claim 15: The non-transitory computer readable medium of claim 10, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.

Claim 24: The system of claim 19, wherein data indicative of the encryption key comprises data that was created using a Shamir algorithm.

383. The additional limitation in claims 15 and 24 is materially the same as the additional limitation in claim 6. Therefore, for the same reasons discussed with respect to claim 6 in §IX.C.4 above, the Torre/Tajima/Krawczyk combination renders claims 15 and 24 obvious.

D. Ground IV: Torre, Tajima, And Foster Render Obvious Claims 7-8, 16-17, and 25-26.

384. Claims 8, 16-17, 25-26 would have been obvious in view of Torre, Tajima and Foster.

385. In this Ground, I introduce illustrations to show how the Torre/Tajima/Foster combination renders Claims 8, 16-17, and 25-26 obvious.

1. Torre/Tajima/Foster Color-Coding Scheme

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

Color	Disclosures in Torre, Tajima and/or Foster
violet	Torre's SHA-1 digital signature information
pink	Torre's encryption operation
lime green	Torre's encrypted data set
	Tajima's encrypted data
	Foster's encrypted data
royal blue	Torre: selection information for selecting between allocation modes
	Tajima: arbitrary length information
red	Torre's shreds
light blue	Torre's intermediate data blocks
	Tajima's data fragments
magenta	Tajima's encryption key
	Foster's encrypted key
olive	Torre's storage servers
brown	Torre's data unit from data blocks
yellow	Torre's allocator
	Tajima's allocation method
forest green	Collection of the encrypted data, SHA-1 digital signature information, and/or encryption key
cyan	Foster's key encrypting key

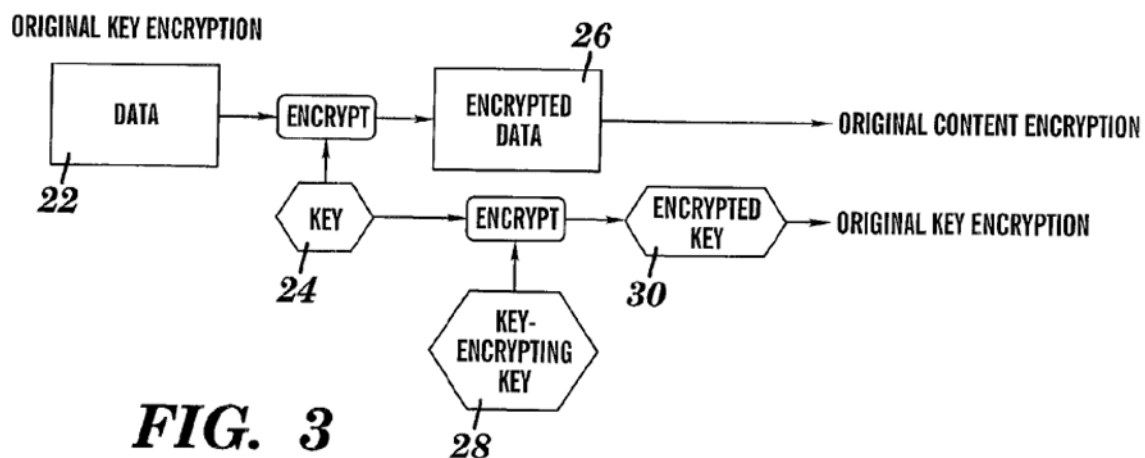
2. Overview of Foster (EX1007)

387. Foster teaches “[a] method and system for attaching a title key [*i.e.*, an encryption key] to encrypted content for synchronized transmission to, or storage by, a recipient.” EX1007 (Foster), Abstract. It further explains that “[s]pecifically, a *title key* used to encrypt content is *itself encrypted*, and *attached to the encrypted*

content so that both can be synchronously transmitted to a recipient.” EX1007 (Foster), [0002]. For example, Figure 3 (below) of Foster describes and illustrates that:

content 22 is encrypted with title key 24 to yield encrypted content 26. Then, title key 24 is itself encrypted with key encrypting key 28. The encrypted title key 30 can then be transmitted along with the encrypted content 26 to a recipient.

EX1007 (Foster), [0053].



EX1007 (Foster), Figure 3.

388. To recover the input data, the recipient decrypts the encrypted data through a two-step process: (1) retrieving the title key by decrypting the encrypted key 30 using the key encrypting key 28, and (2) using the recovered title key to decrypt the encrypted data. EX1007 (Foster), [0055] (“Once recovered, key encrypting key 28 will be used to recover title key 24, which will be used to decrypt

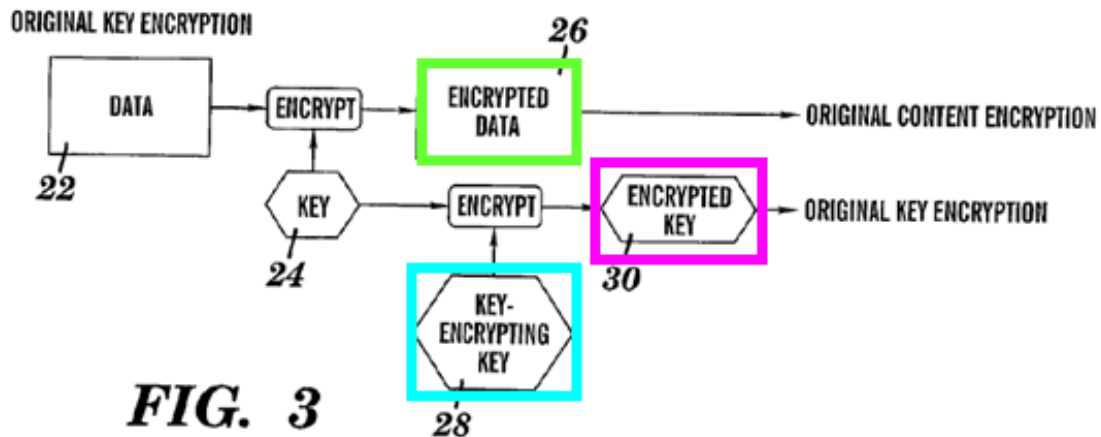
the content.”), [0061] (“Once key encrypting key 28 has been determined, encrypted combination 42 can be decrypted. Then, using the received content usage conditions 36, digest 38 is re-created and title key 24 is recovered.”).

389. Foster explains that “key encrypting key 28 is generally determined based on a key management block (KMB), which is typically delivered to a recipient concurrently with or prior to the delivery of encrypted content 26.” EX1007 (Foster), [0055]. *See also* EX1007 (Foster), [0056] (“As indicated above, because the KMB is necessary to access encrypted content, the KMB should be delivered synchronously with, or prior to, the encrypted content.”). Additionally, “a receiver may cache multiple KMBs (or the corresponding *key encrypting keys*) in memory.” EX1007 (Foster), [0058].

3. Torre/Tajima In Combination With Foster (“Torre/Tajima/Foster”)

390. It would have been obvious to combine the Torre/Tajima System with Foster to enhance the protection of the encryption key by encrypting it with a “key encryption key” prior to storage. EX1007 (Foster), [0053], Figure 3. Specifically, it would have been an obvious design choice to implement the first transformer in part according to Foster’s Figure 3, such that it (1) encrypts the encryption key using a *key encrypting key*, and (2) stores the resulting *encrypted encryption key* rather than the key in its unencrypted form. EX1007 (Foster), [0053]. This is shown in

the figure below, annotated using the same color-coding scheme as that described in the table in §IX.D.1 above.

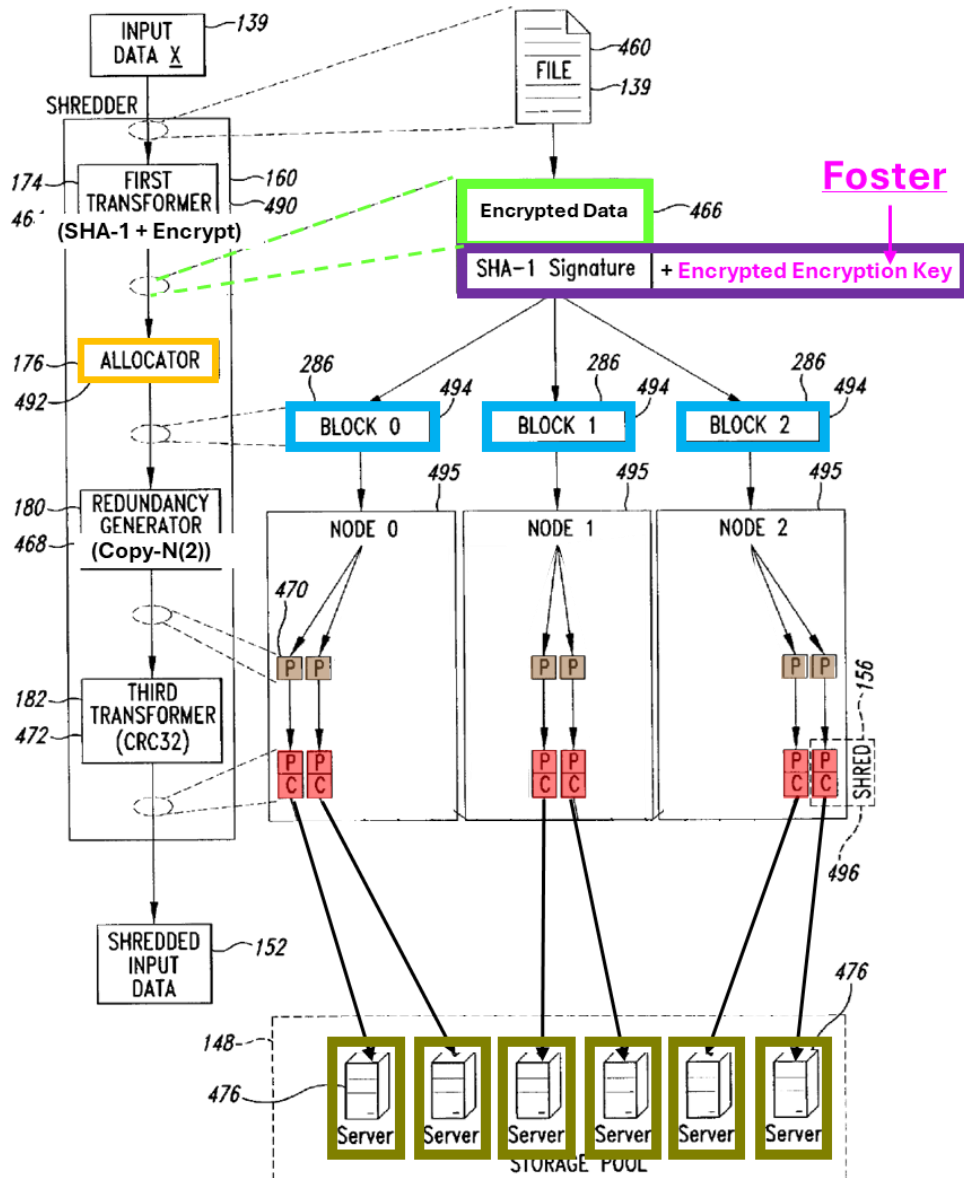


EX1007 (Foster), Figure 3 (annotated).

391. The formed combination is depicted below which replaces the encryption key in the tag per the Torre/Tajima System with the encrypted encryption key as taught by Foster. This is shown in the figure below, annotated to use the same color-coding scheme as that described in the table in §IX.D.1 above.

Torre/Tajima/Foster System

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (modified and annotated).

392. In this combination, the deshredder is capable of recovering the original input data by reversing the operations performed by the shredder, including decrypting the encrypted encryption key using the **key encrypting key** 28. See

EX1003 (Torre), [0078] (“The tags 110 are used to indicate which functions should be performed in various stages of reconstructing the input data 139 from the shredded input data 132.”), [0123] (“For instance, if compression, encryption, or signature generation was done by a particular transformer of the shredder 160, then decompression, decryption, or signature verification would be performed by the inverse transformer of the deshredder 390 corresponding to the particular transformer of the shredder.”); EX1007 (Foster), [0055] (“Once recovered, key encrypting key 28 will be used to recover title key 24, which will be used to decrypt the content.”), [0061] (“By processing a key management block (KMB) (e.g., with valid device keys), key encrypting key 28 is recovered. Once key encrypting key 28 has been determined, encrypted combination 42 can be decrypted.”).

393. The deshredder—functionally similar to the recipient in Foster—can cache the **key encrypting key** 28 in memory, as Foster teaches. *See* EX1007 (Foster), [0058] (“[A] receiver may cache multiple KMBs (or the corresponding **key encrypting keys**) in memory.”).

i. Reasons To Combine Torre/Tajima System with Foster’s Key Encryption Method

394. A POSITA would have been motivated to combine Foster’s key encryption method into the Torre/Tajima system for at least the following two reasons.

395. **First**, a POSITA would have been motivated to combine the Torre/Tajima System with Foster in order to enhance the security of the encryption key. It would have been understood that protecting the encryption key is critical to preventing unauthorized decryption of the encrypted data. *See* §V.C above.

396. Therefore, a POSITA would have been motivated to encrypt the encryption key before storing it, as taught by Foster. EX1007 (Foster), [0053] (“Referring to FIG. 3, the encryption of content 22 using title key 24 is depicted in greater detail. As shown, content 22 is encrypted with title key 24 to yield encrypted content 26. Then, title key 24 is itself encrypted with key encrypting key 28. The encrypted title key 30 can then be transmitted along with the encrypted content 26 to a recipient.”).

397. **Second**, the combination would have merely amounted to applying a known technique (Foster’s key encryption scheme) to a known device (Torre/Tajima System) ready for improvement (for improved security of the stored encryption key) to yield predictable results (improving the stored encryption key’s security).

398. Next, a POSITA would have had a reasonable expectation of success in combining the Torre/Tajima System with Foster for at least the following three reasons.

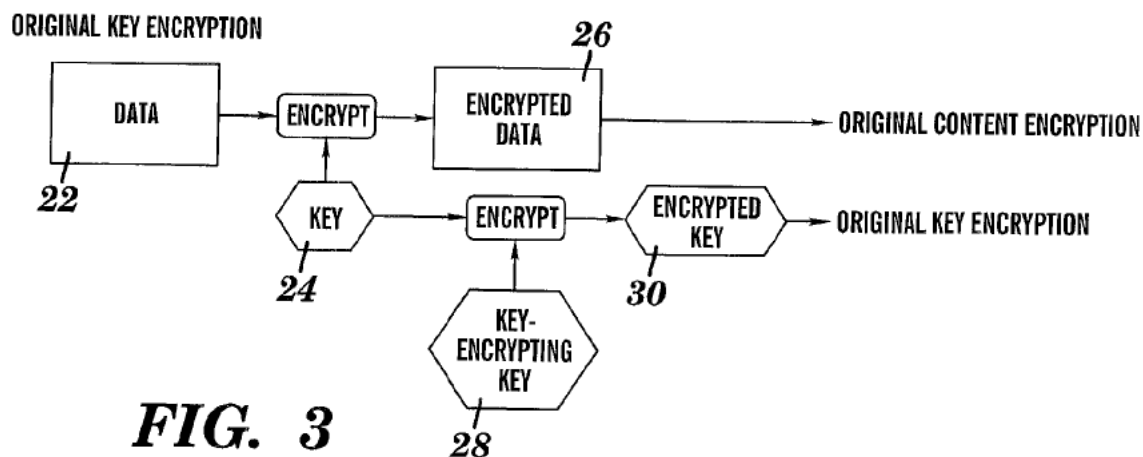
399. **First**, both systems are directed to the same technical objective: securely distributing secret information. Each employs substantially similar

approaches by splitting encrypted data into shares or packets and distributing that data alongside associated encryption key information. *Compare* EX1003 (Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and scalability and adaptability to resources.*”), [0051]–[0052] (“In the process of shredding, input data is generally *transformed into many smaller pieces of data called shreds.* ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of data (step 134).*”) *with* EX1004 (Tajima), [0044] (“It is an object of the present invention to provide a personal information management system, a personal information management method, and a *server that can improve the security of personal information that is registered with areas that are connectible to the*

*Internet.”), [0048] (“Thus, because **personal information that is registered with areas that are connectible to the Internet is divided into a plurality of data portions and then registered with areas that are each under separate control, ...**”), [0109] (“As still another example of a method of dividing personal information, as shown in FIG. 11, personal information that has been submitted to certificate authority 40a by the user of user terminal 10 is first encrypted, and **the encrypted personal information and information regarding the key that is used in the encryption are then divided into a plurality of data portions each of fixed length as shown in FIG. 8, or of arbitrary length as shown in FIG. 9 or FIG. 10.** The plurality of data portions are then arranged into two data fragments, data fragment X and data fragment Y, according to a set method of arrangement, and **data fragment X is then registered with database 41a of certificate authority 40a and data fragment Y is registered with database 41b of certificate authority 40b.**”) and EX1007 (Foster), [0053] (“As shown, content 22 is encrypted with title key 24 to yield encrypted content 26. Then, title key 24 is itself encrypted with key encrypting key 28. The encrypted title key 30 can then be transmitted along with the encrypted content 26 to a recipient.”).*

400. **Second**, implementing the modification would have involved a straightforward adjustment—adding a step to encrypt the encryption key before embedding it in the tag, consistent with the method shown in Foster’s Figure 3

below. The combination would have merely amounted to using a known technique (Foster's key encryption scheme) to improve a similar device (Torre/Tajima System for securely storing input data) in the same way (encrypting the encryption key as taught by Foster) to yield predictable results (*e.g.*, improving the security of the stored encryption key).



EX1007 (Foster), Figure 3.

401. **Third**, a POSITA would have understood that implementing Foster's key encryption scheme within the Torre/Tajima System would have simply involved implementing the corresponding (simple) functionality in software, by writing program code that follows Foster's steps. That task would have been well within the abilities of a POSITA. *See* EX1003 (Torre), [0141]. *See also* §V.C above.

402. At a minimum, it would have been obvious to try to use Foster's key encryption method in Torre/Tajima System. There was a design need to share the

encryption key so that the encrypted data can be decrypted later. And there were multiple predictable solutions to distributing the encryption key: (1) as taught by Torre/Tajima System-1, Torre/Tajima System-2, or Torre/Tajima/Krawczyk (see §§IX.A.4 above and IX.C.3 above); or (2) encrypt the encryption key, as taught by Foster (EX1007 (Foster), [0053]). Each solution would have had a reasonable expectation of success, as explained in §§IX.A.4 above and IX.C.3 above, and in paragraphs just above.

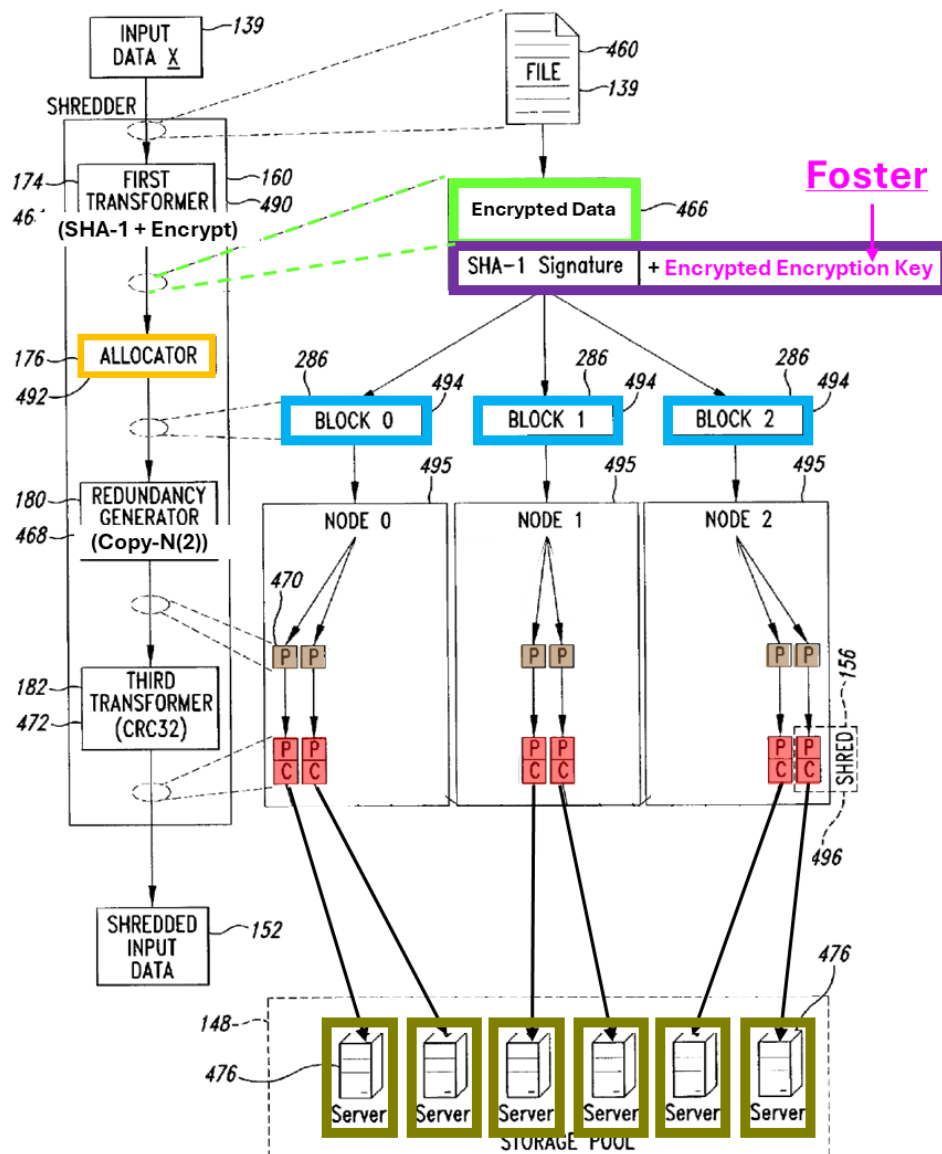
4. **Claim 7: The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.**
 - i. [7A] The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a **second key**; and
 - ii. [7B] including in the **plurality of shares** data indicative of the **encrypted encryption key**.

403. Torre/Tajima/Foster teaches the additional claim limitations of [7A]-[7B]. In the Torre/Tajima/Foster System, the first transformer encrypts the encryption key using a **key encrypting key** and includes the resulting **encrypted encryption key** as part of the tag. EX1007 (Foster), [0053] (“Referring to FIG. 3, the encryption of content 22 using title key 24 is depicted in greater detail. As shown, content 22 is encrypted with title key 24 to yield encrypted content 26. Then,

title key 24 is itself encrypted with key encrypting key 28. The encrypted title key 30 can then be transmitted along with the encrypted content 26 to a recipient.”). See §IX.D.3 above [Torre/Tajima/Foster]. This is shown in the figure below, annotated to use the same color-coding scheme as that described in the table in §IX.D.1 above.

Torre/Tajima/Foster System

SHREDDING BY FIRST AND THIRD TRANSFORMERS WITH ALLOCATION



EX1003 (Torre), FIG. 34 (modified and annotated).

404. Next, the first transformer parses and divides both the **encrypted data** and the tag—including the **encrypted encryption key**—into **shreds**, as detailed in §IX.A.5.vi above for claim limitation [1E]. As a result, the Torre/Tajima/Foster System distributes into the **shreds** (“*plurality of shares*”) fragments of the **encrypted encryption key** (“*data indicative of the encrypted encryption key*”).

405. **In sum**, Torre/Tajima in combination with Foster teaches “*wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key [Torre/Tajima’s encryption key] with a second key [Foster’s key encryption key]; and including in the plurality of shares [Torre’s shreds] data indicative of the encrypted encryption key [Foster’s encrypted encryption key].*”

- 5. Claim 8: The method of claim 7 further comprising the method step of storing the **second key** outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the **second key**.**

406. Torre/Tajima/Foster teaches the additional limitation of claim 8.

407. **First**, the Torre/Tajima/Foster System teaches “*storing the **second key** outside of the **plurality of shares**.*” In The Torre/Tajima/Foster System, the deshredder reconstructs the original input data by reversing the shredding steps. See §IX.D.3 above [Torre/Tajima/Foster]; EX1003 (Torre), [0078] (“The tags 110 are used to indicate which functions should be performed in various stages of

reconstructing the input data 139 from the shredded input data 132.”), [0123] (“For instance, if compression, encryption, or signature generation was done by a particular transformer of the shredder 160, then decompression, decryption, or signature verification would be performed by the inverse transformer of the deshredder 390 corresponding to the particular transformer of the shredder.”). One such reversal operation involves using the **key encrypting key** to decrypt the encrypted encryption key. EX1007 (Foster), [0055] (“Once recovered, key encrypting key 28 will be used to recover title key 24, which will be used to decrypt the content.”), [0061] (“By processing a key management block (KMB) (e.g., with valid device keys), key encrypting key 28 is recovered. Once key encrypting key 28 has been determined, encrypted combination 42 can be decrypted.”). Foster further explains that the **key encrypting key** may be held in memory— “[A] receiver may cache multiple KMBs (or the corresponding key encrypting keys) in memory.” EX1007 (Foster), [0058]. Therefore, the Torre/Tajima/Foster System stores the **key encrypting key** (“*second key*”) separately from the **shreds** (“*plurality of shares*”).

408. **Second**, the Torre/Tajima/Foster System also satisfies the requirement that “*the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares,*” as discussed in §IX.A.5.viii above for claim limitation [1G].

409. **Third**, the Torre/Tajima/Foster combination teaches accessing and utilizing the **key encrypting key** (“**second key**”) in the process of restoring the input data (“**data set**”). In this system, the deshredder reconstructs the input data from the **shreds** by reversing the shredding operations. See §IX.D.3 above [Torre/Tajima/Foster]; EX1003 (Torre), [0078], [0123].

410. For example, the deshredder first retrieves the **encrypted data** and associated tag—which includes the **encrypted encryption key**—from the shreds. See §IX.D.3 above [Torre/Tajima/Foster]; EX1003 (Torre), [0136] (“[T]he deshredder 498 has up to twelve of the third inverse transformers 478 to receive twelve shreds 156, up to three inverse redundancy generators 480 in the second-tier component 394 of the deshredder 498 to receive the three data blocks 494, and an implementation of the deallocator 402 in the first tier component 392 to reassemble the data blocks into the pre-shred 466.”). The deshredder then performs the following steps to restore the data: (1) accesses the **key encrypting key** 28 (“**second key**”) that is stored locally at the deshredder; (2) decrypts the **encrypted encryption key** using the **key encrypting key** 28 (“**second key**”) to recover the original key 24; and (3) decrypts the **encrypted data** using key 24 to obtain the original input. EX1007 (Foster), [0055], [0058], [0061].

411. **In sum**, Torre/Tajima in combination with Foster teaches “*the method step of storing the second key* [Foster’s key encrypting key] *outside of the plurality*

of shares [Torre's shreds], wherein the data set [Torre's input data] is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key [Foster's key encrypting key]."

6. **Claim 16: The non-transitory computer readable medium of claim 10, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.**

Claim 17: The non-transitory computer readable medium of claim 16 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

Claim 25: The system of claim 19, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.

Claim 26: The system of claim 25 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

412. The additional limitation in claims 16 and 25 is materially identical to that in claim 7, and the additional limitation in claims 17 and 26 is materially identical to that in claim 8.

413. Accordingly, for the same reasons discussed with respect to claims 7 (§IX.D.4 above) and 8 (§IX.D.5 above), the Torre/Tajima/Foster combination renders claims 16–17 and 25–26 obvious.

E. Ground V: Orsini And Torre Render Obvious Claims 1-5, 9-14, 18-23, and 27.

414. In this Ground, I introduce illustrations to show how the Orsini/Torre combination renders Claims 1-5, 9-14, 18-23, and 27 obvious.

415. In this Ground, I introduce illustrations to show how the Torre/Tajima/Orsini combination renders Claims 1-5, 9-14, 18-23, and 27 obvious.

1. Torre/Tajima/Orsini Color-Coding Scheme

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

Color	Disclosure in Orsini or Torre
light blue	Orsini's splitting of encrypted data using session master key
dark orange	Orsini's splitting of session master key
light amber	Orsini's encrypted data share
magenta	Orsini's session master key or its shares
lime green	Orsini's encrypted input data
purple	Torre's SHA-1 digital signature
forest green	Collection of encrypted input data and SHA-1 digital signature
red	Orsini's shares

2. Overview of Orsini (EX1008)

417. Orsini contains disclosures that are substantially the same as those found in the '802 Patent. Specifically, Orsini's Figures 1–25, along with their accompanying descriptions, closely mirror the corresponding Figures 1–25 and descriptions in the '802 Patent. *See* EX1012 (showing redline comparisons between Orsini and the specification of the '802 Patent).

418. Because the '802 Patent relies on those figures and descriptions to provide written description support for the majority of its claim limitations, Orsini likewise discloses subject matter that invalidates most of the limitations in the '802 Patent's claims.

419. For instance, similar to Figure 21 of the '802 Patent, Figure 21 in Orsini (reproduced below) depicts a secure data parser that is relevant to the '802 Patent's claims.

420. It “show[s] the steps of the process performed by the parser software suite on data to store the session master key with the parsed data.” EX1008 (Orsini), [0294]. *See* §VI above ['802 Patent]; EX1008 (Orsini), [0295]–[0298] (“1. Generating a session master key and encrypt the data using RS1 stream cipher. 2. Separating the resulting encrypted data into four shares or portions of parsed data according to the pattern of the session master key. 3. In this embodiment of the method, *the session master key will be stored along with the secured data shares*

in a data depository. Separating the session master key according to the pattern of the Parser Master Key and append the key data to the encrypted parsed data.”), [0346]–[0357] (“In one embodiment, it may not be desired to have any one depository contain enough information to decrypt the information held there, so *the key required to decrypt the share is stored in a different data depository:* *Depository 1: SD1, Kn Depository 2: SD2, K1 Depository 3: SD3, K2 Depository n: SDn, K3.*”), [0363]–[0367] (“In the incorporated session key example described herein, the session master key is split into “n” shares according to the contents of the installation dependent Parser Master Key (SK1, SK2, SK3, SKn): Depository 1: SD1, Kn, SK1 Depository 2: SD2, K1, SK2 Depository 3: SD3, K2, SK3 Depository n: SDn, K3, SKn.”).

421. Figure 21 of the '802 Patent and Figure 21 of Orsini are shown and annotated below using the same color-coding scheme as that described in the table in §IX.E.1 above.

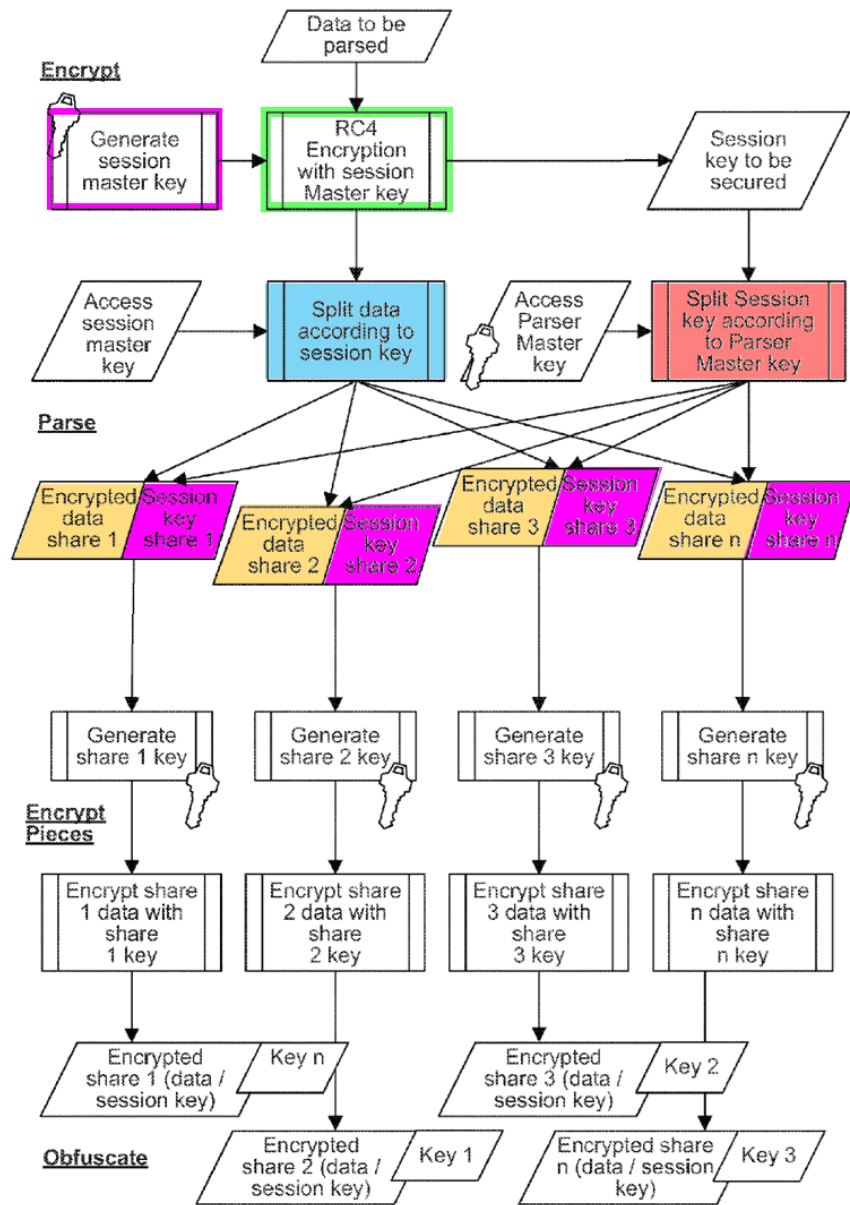
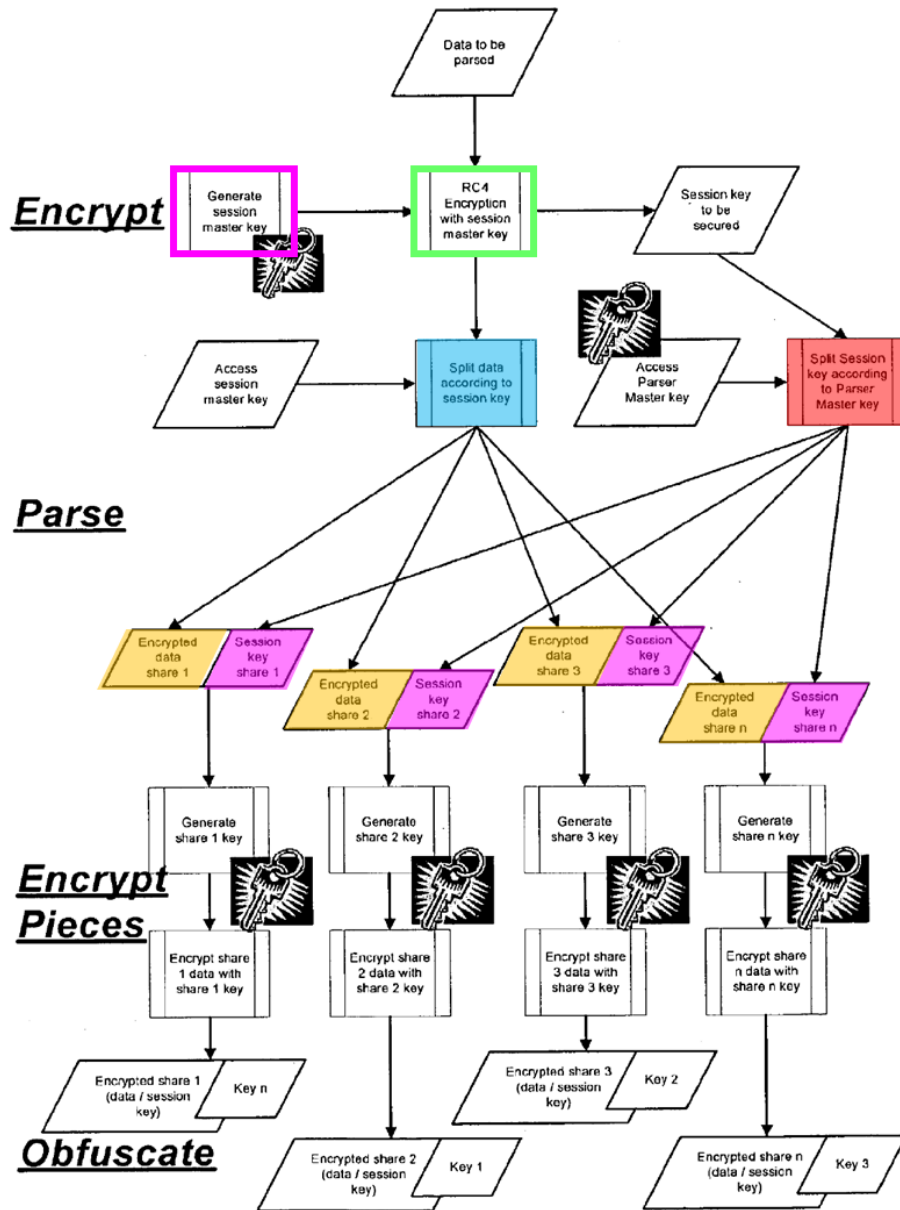


FIG. 21

EX1001 ('802 Patent), Figure 21 (annotated).

Figure 21



EX1008 (Orsini), FIG. 21 (annotated).

422. Orsini further discloses that the act of parsing data into multiple shares can be accomplished through “[a] cryptographic split (cryptosplit).” EX1008 (Orsini), [0281]. This approach “provide[s] sufficient redundancy in the shares such

that only a subset of the shares are needed to reassemble or restore the data to its original or useable form.” EX1008 (Orsini), [0283]. *See also* EX1008 (Orsini), [0284], [0288]–[0292] (“The parser software suite of the present invention, in one embodiment, addresses this problem by performing a cryptographic split or parsing of the encrypted file into two or more portions or shares.”), [0293].

423. Orsini’s Figure 21 illustrates an implementation of the redundant cryptosplit process. Orsini states that “[t]he *parser software suite* of the present invention ... perform[s] a *cryptographic split* or parsing of the encrypted file into two or more portions or shares, and ... preferably four or more shares.” EX1008 (Orsini), [0292]. Orsini further clarifies that “[a]n example of one embodiment of the *parser software suite* of the present invention and an example of how it may be utilized is *shown in FIG. 21*.” EX1008 (Orsini), [0293]. Accordingly, Orsini teaches using the *parser software suite* that performs a *cryptographic split* or parsing of the encrypted file in the context of Figure 21. EX1008 (Orsini), [0293] (“[I]t is readily apparent to those of ordinary skill in the art that the parser software suite of the present invention may be utilized in a wide variety of ways in addition to the non-limiting example below.”) (referring to Figure 21), [0424] (“other combinations, admissions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein.”).

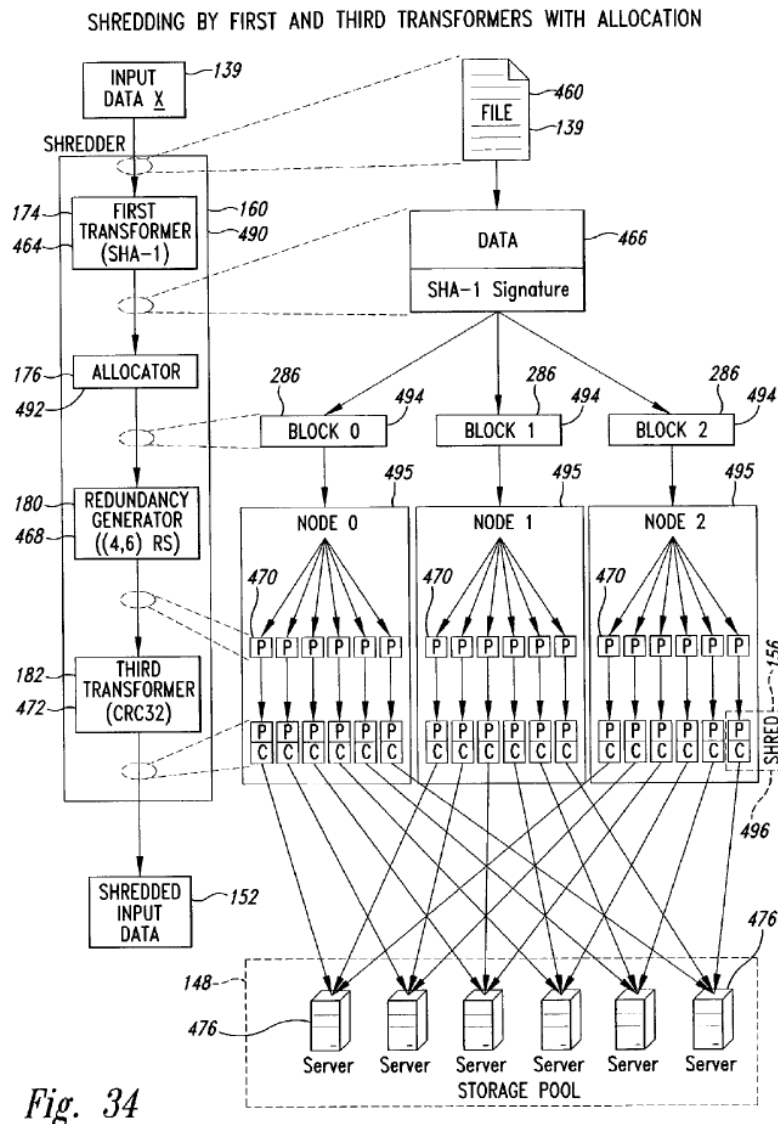
424. At a minimum, it would have been obvious to use a redundant cryptosplit process in Orsini's Figure 21 "such that only a subset of the shares are needed to reassemble or restore the data to its original or useable form." EX1008 (Orsini), [0283].

425. It was well known in the art that adding such redundancy provides resilience in case one of the storage devices fails. EX1003 (Torre), [0111] ("The end result of the redundancy generator is to generate data redundant to the input data 139 so that some of the shredded input data 106 that is stored in storage pool 148 could be lost without preventing the input data from being recovered."). For this reason, a POSITA would have been motivated to incorporate the redundant cryptosplit method into Figure 21 to achieve a fault-tolerant storage solution. A POSITA would have had a reasonable expectation of success, as the combination would have merely amounted to applying a known technique (redundant cryptosplit process) to a known device (storage system of Orsini's Figure 21) ready for improvement (to guard against failure of certain storage devices) to yield predictable results of improving fault tolerance so "that only a subset of the shares are needed to reassemble or restore the data to its original or useable form." EX1008 (Orsini), [0283]. *See also* EX1008 (Orsini), [0424]; §V.D above.

3. Orsini In Combination With Torre (“Orsini/Torre”)

426. Orsini describes recovering the original input data from a redundant set of shares by reversing the steps used to generate those shares. *See* EX1008 (Orsini), [0294]–[0300]. While Orsini does not explicitly disclose a mechanism for verifying that the restored data is identical to the original input, such integrity verification techniques were well known in the art at the time. *See* §V.C.1 above.

427. For example, Torre describes “[a] system and method for shredding and deshredding ... to *store and retrieve data* while emphasizing considerations such as redundancy, *integrity*, and security of data.” EX1003 (Torre), [0049]. *See* §IX.A.2 above [Torre]. Torre’s Figure 34 (below) depicts this shredding system, which begins by applying “a SHA-1 signature generation function to generate a signature” and appending that signature to the input data to produce a pre-shred 466. EX1003 (Torre), [0131]. The system then takes the resulting pre-shred 466 and proceeds to parse and divide it into multiple redundant shreds. EX1003 (Torre), [0131]–[0132].



EX1003 (Torre), Figure 34.

428. As illustrated in Torre's Figure 35 below, during the reconstruction phase, the deshredder processes the retrieved subset of shreds to reassemble the pre-shred 466. EX1003 (Torre), [0133]–[0134]. Once the pre-shred is recovered, the deshredder reads the appended SHA-1 signature and “then performs an SHA-1 signature verification function to verify that the data of the pre-shred 466 is

uncorrupted.” EX1003 (Torre), [0134]. “[I]f so, the data file 460 is outputted by the deshredder 477 as the input data 139.” EX1003 (Torre), [0134]. *See also* EX1003 (Torre), [0088] (referring to “a final check of data integrity”). Thus, Torre’s system “allows for **data integrity checking** to verify the validity of reconstruction of the input data.” EX1003 (Torre), [0086].

DESHREDDING BY FIRST AND THIRD INVERSE TRANSFORMERS WITH DEALLOCATOR

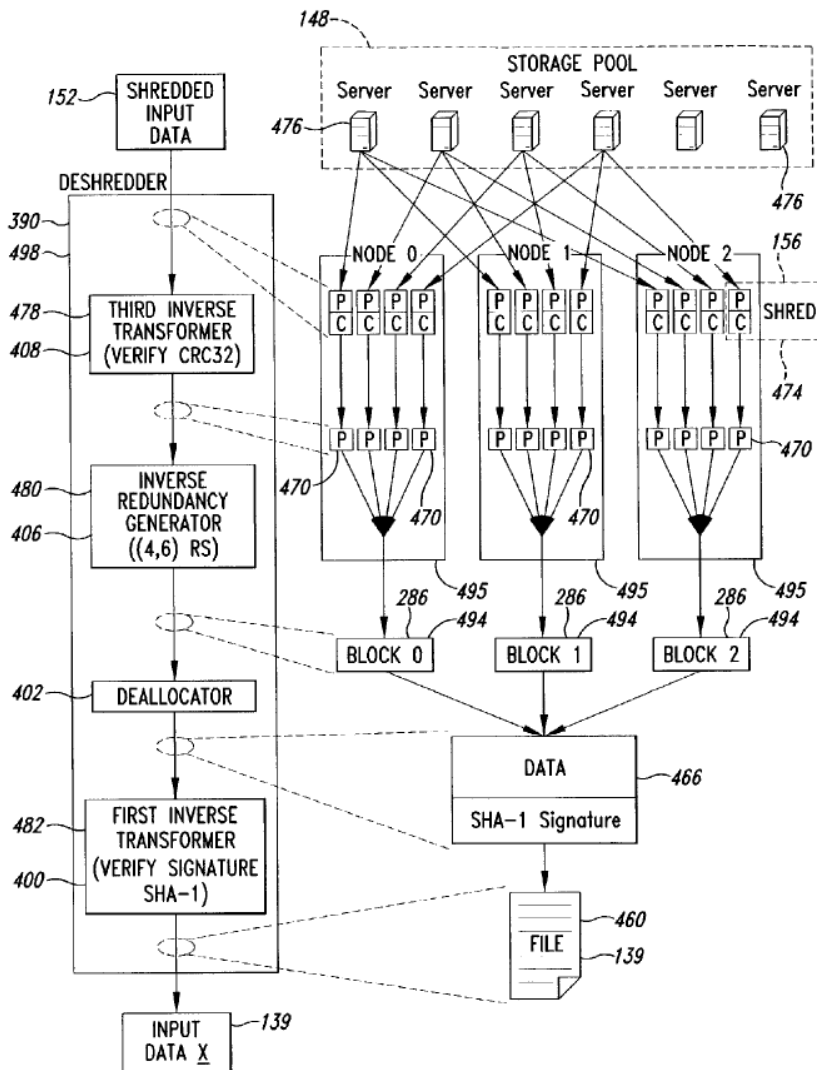


Fig. 35

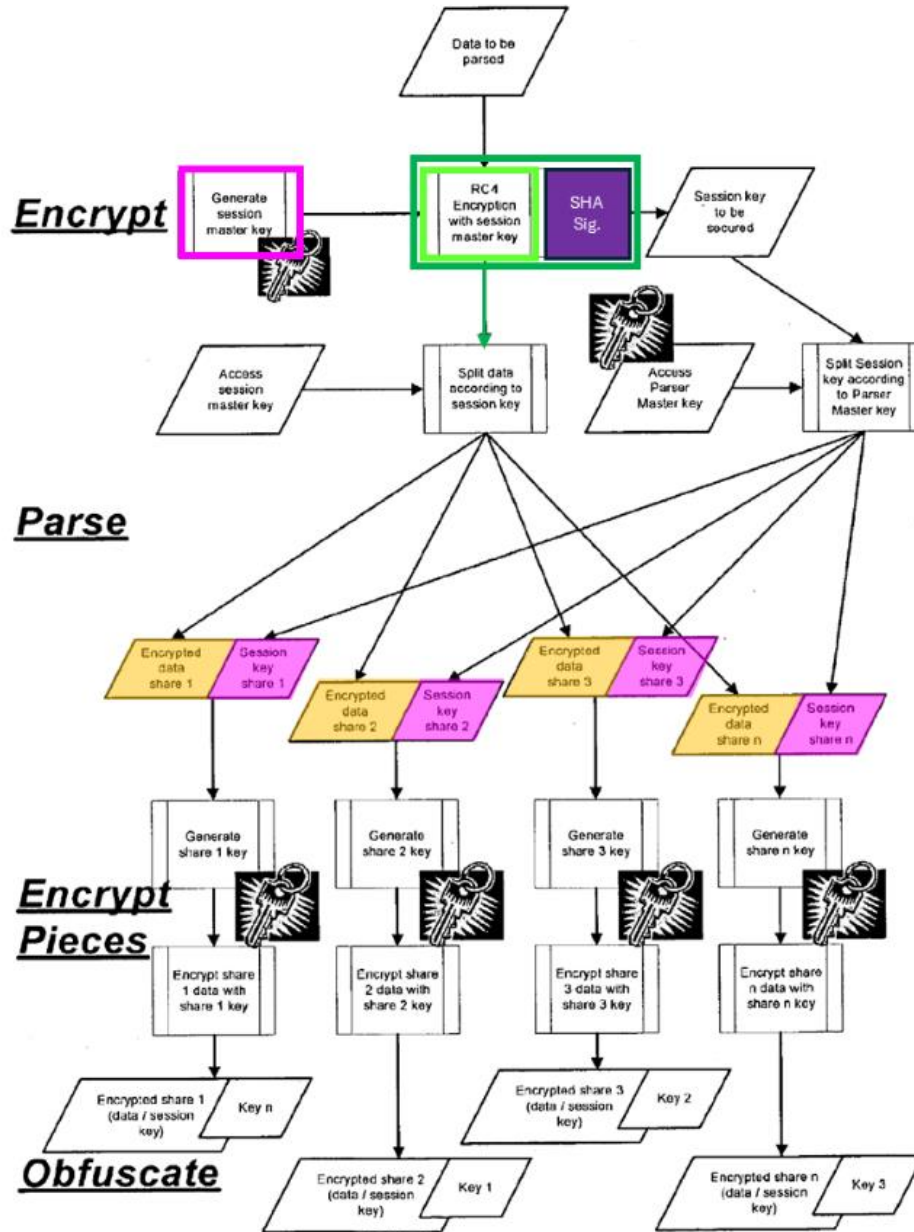
EX1003 (Torre), Figure 35.

429. It would have been obvious to modify Orsini's Figure 21 by incorporating the enhancements described in Torre, specifically to: (1) calculate a **SHA-1 signature** of the input data, (2) encrypt the input data, and (3) append the **SHA-1 signature** to the encrypted data. Torre specifically discloses such modification. For example, in Torre, when a shredder receives input data, the shredder first computes the digital signature, encrypt the input data, and subsequently adds the digital signature and encryption information. *See* §IX.A.2 above [Torre]; EX1003 (Torre), Figure 11 (Rule 9), [0091], Figure 34. And just as in the combination, Torre then teaches generating shreds from the tagged encrypted data. *See* §IX.A.2 above [Torre].

430. A visual depiction of the combination of Orsini and Torre is shown and annotated below using the same color-coding scheme as that described in the table in §IX.E.1 above.

Orsini / Torre System

Figure 21



EX1008 (Orsini), FIG. 21 (annotated).

i. Reasons To Combine Orsini and Torre

431. A POSITA would have been motivated to combine the teachings of Orsini and Torre for at least the following two reasons.

432. **First**, both Orsini and Torre are directed to distributed storage architectures that divide data into portions or shares for storage across multiple devices, and later reconstruct the original input from those shares. *Compare* EX1003 (Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and scalability and adaptability to resources.*”), [0051]–[0052] (“In the process of shredding, input data is generally *transformed into many smaller pieces of data called shreds.* ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of*

data (step 134).”) with EX1008 (Orsini), Abstract (“The present invention provides a method and system for securing sensitive data from unauthorized access or use.”), [0278]–[0280] (“In a separate embodiment, the present invention comprises a complete system for performing the data securing methods of the present invention on any data set. The computer system of this embodiment comprises a data splitting module that comprises the functionality shown in FIG. 8 and described herein.”), [0291] (“The data splitting or parser modules of the present invention may be integrated into a wide variety of infrastructure products or applications as desired.”).

433. Torre highlights that, during reconstruction, it is advantageous to confirm that the reassembled data is the same as the original data, and one method of doing so is through using a digital signature generated from the original data. EX1003 (Torre), [0086] (“Implementations using signature generation allows for *data integrity checking to verify the validity of reconstruction* of the input data 139 from the shredded input data 156. ... In some implementations, the first transformer 174 uses a signature generation function such as SHA-1 to determine later on in a first-tier of the deshredding process described further below whether the original input data 139 as been reproduced through the deshredding process.”), [0088] (“Examples of some signature generation functions include the Secure Hash Algorithms (SHA) (e.g. SHA-1), the UMAC function, the MD5 function, and the HMAC function.”), [0134] (“... the third inverse transformer(s) 478 use a CRC32

signature verification function to verify that data of the received shreds is not corrupted.”). Therefore, a POSITA would have been motivated to enhance Orsini’s system by adopting Torre’s method of computing a digital signature for the input data and appending it to the encrypted data.

434. This would have enabled Orsini’s system to perform data integrity verification during the reconstruction phase. EX1003 (Torre), [0086], [0088] (“These examples would be particularly useful for the first transformer 174, whose generated signature could be used later in the deshredding process as a final check of data integrity.”).

435. **Second**, the combination would have merely amounted to applying a known technique (Torre’s digital signature generation scheme) to a known device or method (Orsini’s system and method) for ready for improvement (improving the integrity of reconstructed data) to yield predictable results (using the digital signature to verify the integrity of the reconstructed data).

436. Next, a POSITA would have reasonably expected success in combining Torre with Orsini for at least the following three reasons.

437. **First**, as noted, both Orsini and Torre are directed to similar distributed storage architectures. *Compare* EX1003 (Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and*

scalability and adaptability to resources.”), (“In the process of shredding, input data is generally *transformed into many smaller pieces of data called shreds*. ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of data (step 134).*”) with EX1008 (Orsini), Abstract (“The present invention provides a *method and system for securing sensitive data* from unauthorized access or use.”), [0278]–[0280] (“In a separate embodiment, the present invention comprises a complete system for performing the *data securing methods* of the present invention on any data set. ... This embodiment may further comprise a data storage facility or multiple data storage facilities, as well. ... The process comprises, in one embodiment, splitting of the data according to the contents of a generated random number, or key and performing the same cryptographic splitting of the key used in the encryption of splitting of the data to be

secured into two or more portions, or shares, of parsed data, and in one embodiment, preferably into four or more portions of parsed data, encrypting all of the portions, then scattering and storing these portions back into the database, or relocating them to any named device, fixed or removable, depending on the requestor's need for privacy and security.”), [0291] (“The data splitting or parser modules of the present invention may be integrated into a wide variety of infrastructure products or applications as desired.”).

438. **Second**, the proposed combination would have involved a straightforward modification to Orsini's system—specifically, incorporating Torre's technique of appending a digital signature to the encrypted input data, while continuing to follow Orsini's original processing flow for the remaining operations. Implementing this digital signature step would have simply required writing a program code for a well-known technique, a task well within the capabilities of a POSITA. *See* EX1003 (Torre), [0141] (“Those skilled in the art will recognize that the embodiments disclosed herein, in whole or in part, can be equivalently implemented in integrated circuits, as one or more computer programs running on one or more computers (e.g., as one or more programs running on one or more data processing systems), as one or more programs running on one or more controllers (e.g., microcontrollers) as one or more programs running on one or more processors e.g., microprocessors, as firmware, or as virtually any combination thereof, and that

designing the circuitry and/or writing the code for the software and or firmware would be well within the skill of one of ordinary skill in the art in light of this disclosure.”). *See also* §V.C above. This modification would predictably result in embedding digital signature information into the shares, enabling integrity verification of the reconstructed data—precisely as Torre teaches.

439. **Third**, accordingly, this combination would have amounted to integrating known prior art components—namely, Orsini’s distributed storage system and Torre’s digital signature generation technique—using established implementation practices, such as writing software to incorporate Torre’s signature functionality into Orsini’s framework. This straightforward integration would have led to predictable outcomes, as discussed here, and therefore would have been obvious to a POSITA.

440. Thus, modifying Orsini with Torre’s digital signature technique 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 embedding digital signatures to verify data integrity is a well-established technique). Lastly, a POSITA would have had a reasonable expectation of success because Torre merely adds implementation detail to functionality already aligned with Orsini’s goals.

4. **Claim 1: A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.**

- i. **[1Pre] A method for securing a data set, the method steps implemented by a programmed computer system, the method steps comprising:**

441. To the extent the preamble is limiting, Orsini discloses claim preamble [1Pre].

442. Orsini teaches that “[t]he present invention provides a method and system for *securing sensitive data* from unauthorized access or use.” EX1008 (Orsini), Abstract, [0059]. *See also* EX1008 (Orsini), [0278] (“the present invention comprises a complete system for performing the *data securing methods* of the present invention on any *data set*.”). These techniques are executed through a “parser *software* suite,” which is software that programs a computer system. *See* EX1008 (Orsini), [0278], [0293]–[0294], [0409]–[0410], [0422]; Figure 21.

443. **In sum**, Orsini discloses “[a] *method for securing a data set* [Orsini’s data set], *the method steps implemented by a programmed computer system* [Orsini’s parser *software* suite that programs a computer system].”

ii. [1A] creating integrity information using the data set

444. Orsini in combination with Torre teaches claim limitation [1A].

445. As explained in §IX.A.5.ii above [Ground I, claim limitation 1A], Torre discloses claim limitation [1A], and it would have been obvious to incorporate Torre’s digital signature functionality into Orsini’s framework to create a **SHA-1 signature** (“*integrity information*”) using Orsini’s “data to be parsed.” See §IX.E.3 above [Orsini/Torre]. See also §V.C.1 above. The input subject to parsing in Orsini—such as the input data in Figure 21—qualifies as a “*data set*.” See EX1008 (Orsini), [0278] (“the present invention comprises a complete system for performing the data securing methods of the present invention on any *data set*.”); Claim 1(a) (“encrypting a *data set* to provide an encrypted data set”); Claim 9.

446. **In sum**, Orsini in combination with Torre teaches “*creating integrity information* [Torre’s SHA-1 digital signature information] *using the data set* [Orsini’s data set].”

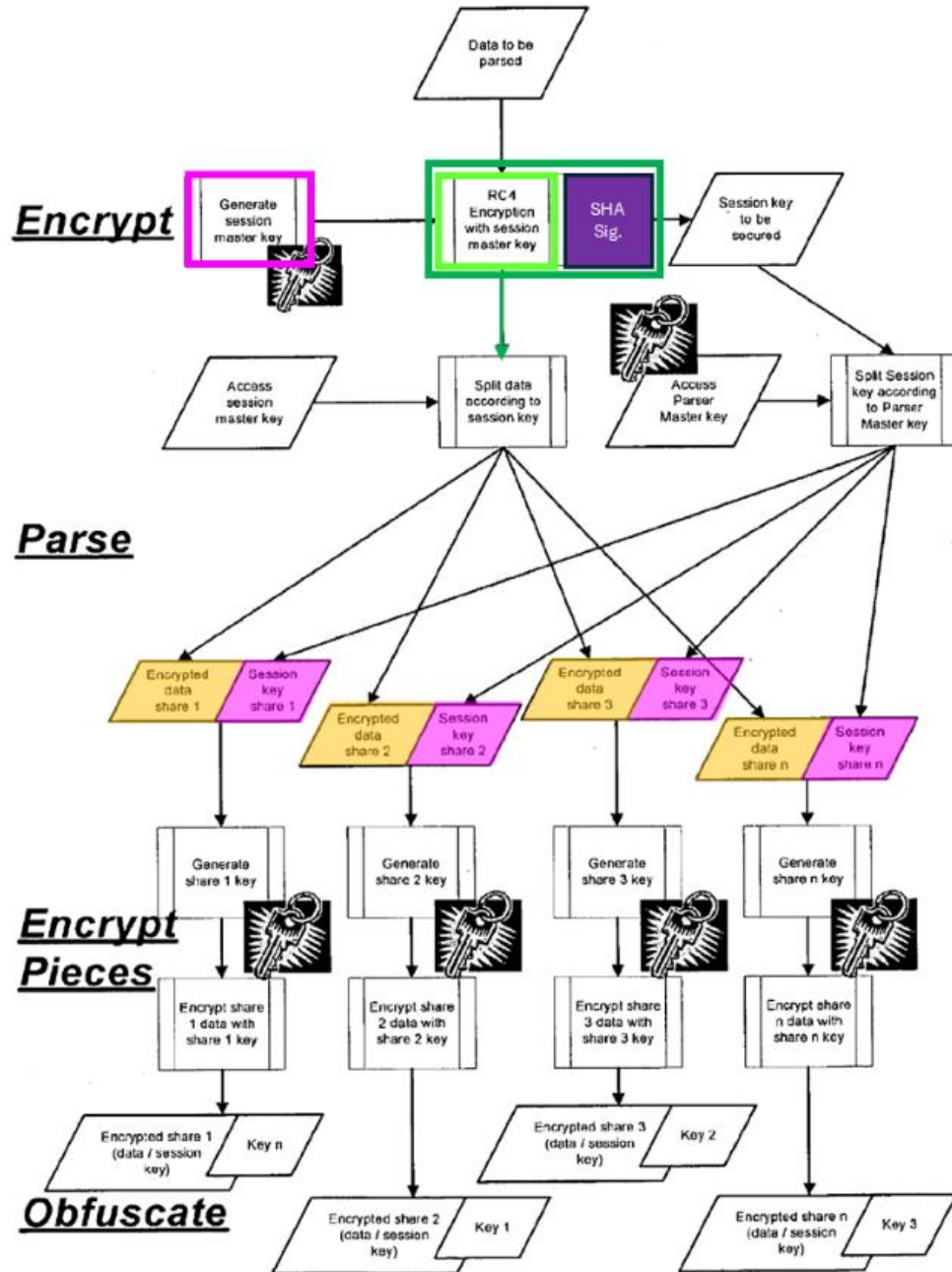
iii. [1B] encrypting the data set based on an encryption key to produce an encrypted data set;

447. Orsini discloses claim limitation [1B].

448. Orsini describes “[g]enerating a session master key and encrypt[ing] the data using RS1 stream cipher” to produce an encrypted data set, as illustrated in Figure 21 annotated below. EX1008 (Orsini), [0295], Figure 21. This Figure is shown and annotated below using the same color-coding scheme as that described in the table in §IX.E.1 above. *See also* EX1008 (Orsini), [0296] (referencing the “resulting encrypted data”); Claim 1(a) (“encrypting a data set to provide an encrypted data set”).

Orsini / Torre System

Figure 21



EX1008 (Orsini), FIG. 21 (annotated).

449. **In sum**, Orsini discloses “*encrypting the data set* [Orsini’s data set] *based on an encryption key* [Orsini’s session master key] *to produce an encrypted data set* [Orsini’s encrypted data set].”

- iv. [1C] generating **data splitting information**, wherein the **data splitting information** is usable to determine into which of a plurality of shares of data a unit of data of the **encrypted data set** will be placed;

450. Orsini discloses or renders obvious claim limitation [1C].

451. Orsini describes “[g]enerating a **session master key**” (“*data splitting information*”) and then “[s]eparating the resulting **encrypted data** into **four shares or portions of parsed data according to the pattern of the session master key**.” EX1008 (Orsini), [0295]–[0296], Figure 21.

452. Orsini further states, “[t]he resulting **four shares of data** will contain *encrypted portions of the original data* and portions of the session master key.” EX1008 (Orsini), [0298].

453. As one example, in distributing data “into one of the 4 shares,” Orsini uses a cryptosplit method that produces a sequence of random numbers (*i.e.*, the session master key), where each number falls between 1 and 4, each corresponding to one of the four shares. A given data unit is assigned to a share based on its associated random number. EX1008 (Orsini), [0282]. *See also* EX1008 (Orsini), [0284]–[0290].

454. Because this “pattern of the *session master key*” governs how the *encrypted data* is split into *four shares* that “contain *encrypted portions of the original data*,” Orsini satisfies the requirements of claim limitation [1C]. EX1008 (Orsini), [0295], [0298].

455. If Patent Owner argues that Figure 21 of Orsini does not disclose “*data splitting information*” because the *session master key* is used to represent both the “*encryption key*” and the “*data splitting information*,” I disagree. I understand that a POSITA would have understood that a claim of a patent (such as claim 1 of the ’802 Patent) to cover embodiments described in the patent itself. Moreover, the ’802 Patent provides an embodiment in which the session master key—like the one disclosed in Orsini—is “used to encrypt the data” and also functions as a “splitting key.” EX1001 (’802 Patent), 60:57–58 (“*The Session Master Key is used to encrypt the data* prior to the parsing and splitting operations. The Session Master Key is used to encrypt the data prior to the parsing and splitting operations.”), 74:17–21 (“As previously discussed, the secure data parser may use keys in the parsing and splitting of data. For purposes of clarity and brevity, these keys shall be referred to herein as “splitting keys.” For example, *the Session Master Key, previously introduced, is one type of splitting key*.”). Therefore, a POSITA would have understood that a single key, such as the session master key, can meet both the “*encryption key*” and the “*data splitting information*” limitations.

456. At the very least, it would have been obvious to a POSITA to generate a separate sequence of random numbers—distinct from the session master key—for the purpose of splitting the encrypted data. Doing so would have simply involved using two independent keys to carry out the same sequence of operations already described in Figure 21 of Orsini. EX1008 (Orsini), [0282]; *see also* EX1008 (Orsini), [0284]–[0290]. A POSITA would have recognized that employing two keys instead of one enhances overall security, because an attacker would need to compromise both keys to reconstruct the original data. *See* §V.C above. *See also* EX1008 (Orsini), [0424] (“other combinations, admissions, substitutions and modifications will be apparent to the skilled artisan in view of the disclosure herein.”).

457. **In sum**, Orsini discloses or renders obvious “*generating data splitting information* [Orsini’s session master key or separate sequence of random numbers], *wherein the data splitting information is usable to determine into which of a plurality of shares of data* [Orsini’s shares] *a unit of data of the encrypted data set* [Orsini’s encrypted data set] *will be placed.*”

- v. [1D] separating the encrypted data set into the plurality of shares based on the data splitting information;

458. Orsini discloses or renders obvious claim limitation [1D].

459. It describes “[s]eparating the resulting **encrypted data** into **four shares** or portions of parsed data *according to the pattern of the session master key.*”

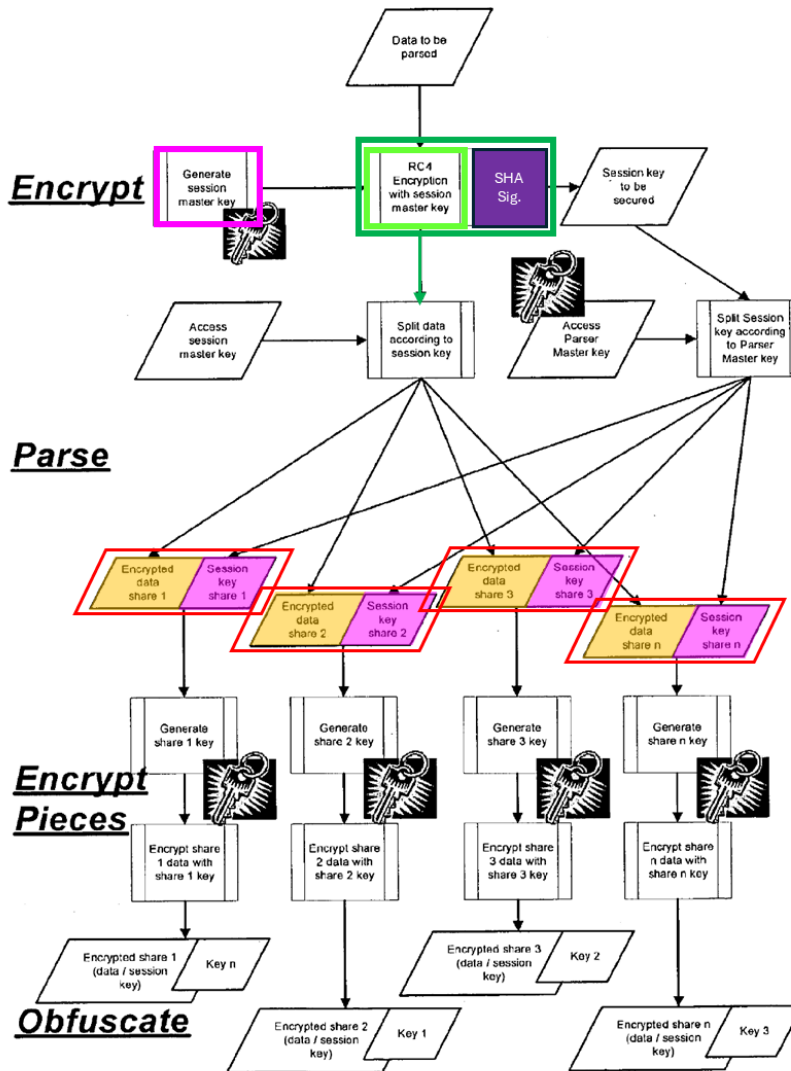
EX1008 (Orsini), [0282], [0284]–[0290], [0296], Figure 21 (annotated below).

460. To the extent that the “*data splitting information*” is interpreted to be distinct from the “*encryption key*,” it would have been obvious to a POSITA to use a distinct key for that purpose, as explained in §IX.E.4.vi below [Ground V, claim limitation 1C].

461. A visual depiction of the combination of Orsini and Torre is shown and annotated below using the same color-coding scheme as that described in the table in §IX.E.1 above.

Orsini / Torre System

Figure 21



EX1008 (Orsini), FIG. 21 (annotated).

462. **In sum**, Orsini discloses or renders obvious “*separating the encrypted data set [Orsini’s encrypted data set] into the plurality of shares [Orsini’s shares] based on the data splitting information [Orsini’s session master key or separate sequence of random numbers].*”

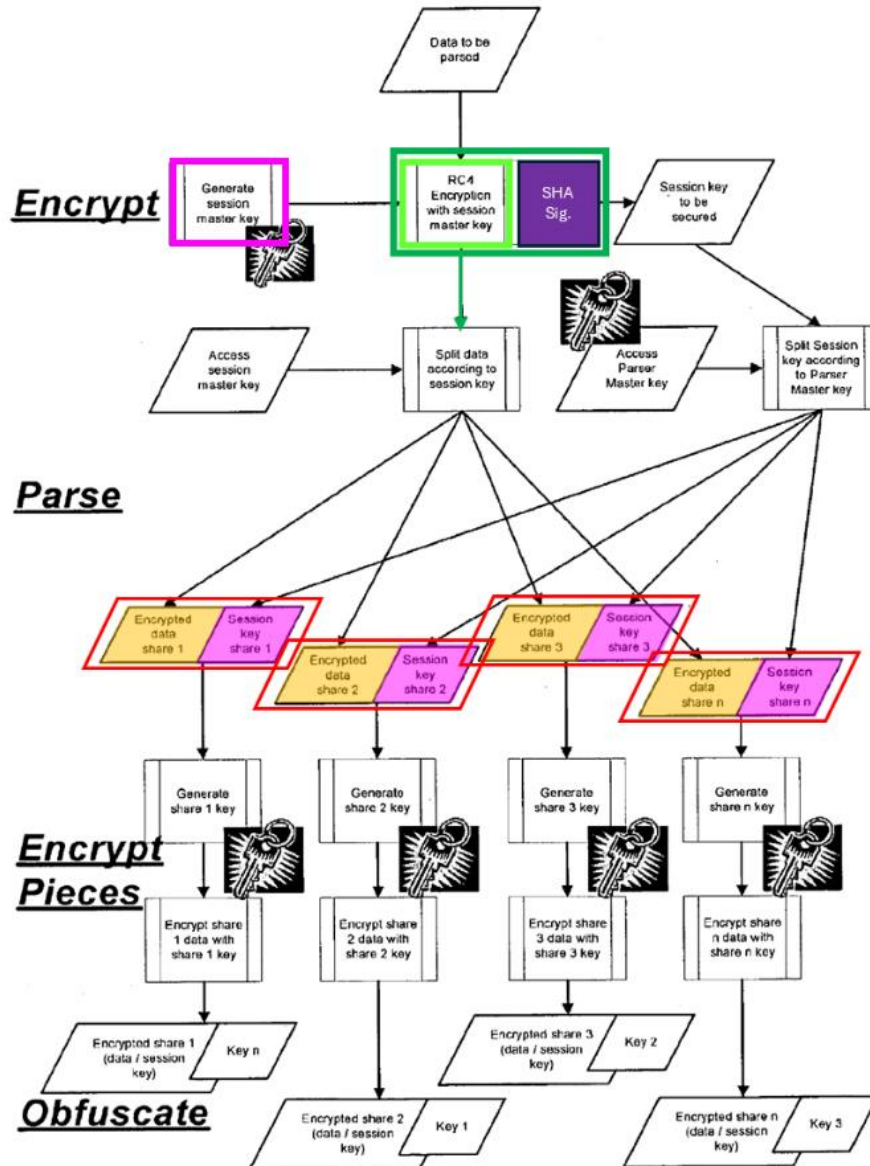
vi. [1E] including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and

463. Orsini in view of Torre teaches claim limitation [1E].

464. As illustrated in Orsini's annotated Figure 21 below, the Orsini/Torre combination discloses separating a collection that includes the encrypted data, the encryption key, and a SHA-1 signature into distinct shares. EX1008 (Orsini), [0296]–[0298] (“The resulting four shares of data will contain encrypted portions of the original data and portions of the session master key.”); EX1003 (Torre), [0131] (“... generate a pre-shred 466 containing the received data and a tag of the SHA-1 signature.”); *see also* §IX.E.4.v above [Ground V, claim limitation 1D]. Therefore, the resulting shares include information indicative of both (a) the encryption key and (b) the SHA-1 signature, which serves as the “integrity information.” A visual depiction of the combination of Orsini and Torre is shown and annotated below using the same color-coding scheme as that described in the table in §IX.E.1 above.

Orsini / Torre System

Figure 21



EX1008 (Orsini), FIG. 21 (annotated).

465. **In sum**, Orsini in combination with Torre teaches “including in the plurality of shares [Orsini’s shares] data indicative of (a) the encryption key

[Orsini's session master key] *and (b) the integrity information* [Torre's SHA-1 digital signature information].”

vii. [1F] causing the **plurality of shares** to be stored in respective separate storage locations;

466. Orsini discloses or renders obvious claim limitation [1F].

467. It explains that, in Figure 21, “the **secured data shares** [are stored] in a *data depository*.” EX1008 (Orsini), [0297]. As one example, Orsini describes “[e]ncrypting each **share**” and storing those encrypted shares across different “locations.” EX1008 (Orsini), [0299]. It further teaches that these shares may be saved in “different physical and/or logical locations.” EX1008 (Orsini), [0292] (“storing the shares in different physical and/or logical locations”).

468. That is consistent with Orsini's other teachings that “the depository system 700 comprises multiple *geographically separated independent data storage systems*.” EX1008 (Orsini), [0111]. *See also* EX1008 (Orsini), [0111] (“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.”), [0114] (“In addition, the pairings may all be stored on one depository, and may be stored in separate locations on said depository.”), [0175] (“According to one embodiment of the invention, each of the trust engines 1305, 1310, 1315 and

1320 are geographically separated...”), [0379] (“One advantage is the security gained from moving shares of the data to different locations on one or more data depositories or storage devices, that may be in different logical, physical or geographical locations.”), Claim 1(d) (§IX.E.4.v above). A POSITA would have known that most systems that distribute their storage nodes, like most distributed systems, are geographically separated. *See* §V.B.1 above.

469. **In sum**, Orsini discloses “*causing the plurality of shares* [Orsini’s shares] *to be stored in respective separate storage locations* [Orsini’s geographically separated independent data storage systems].”

viii. [1G] wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.

470. Orsini discloses or renders obvious claim limitation [1G].

471. It describes performing a data-splitting operation using “[a] cryptographic split (cryptosplit)” (EX1008 (Orsini), [0281]), which is designed to “provide sufficient *redundancy* in the shares such that *only a subset of the shares are needed* to reassemble or restore the data to its original or useable form.” EX1008 (Orsini), [0283]. A POSITA would have recognized that “only a subset of the shares are needed” implies having a lower bound (*i.e.*, a threshold) on the minimum number of shares needed to recover the data. *See also* §V.C.2 above. *See also* EX1008 (Orsini), [0284], [0288]–[0292] (“The parser software suite of the present invention,

in one embodiment, addresses this problem by performing a cryptographic split or parsing of the encrypted file into two or more portions or shares.”), [0293].

472. As discussed in §IX.E.2 above [Orsini], Figure 21 reflects the execution of this redundant cryptosplit technique—or, at a minimum, it would have been obvious to apply such a redundancy based approach in that context.

473. **In sum**, Orsini discloses or renders obvious “*wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares* [using Orsini’s cryptographic split (cryptosplit) technique].”

5. Claim 2: The method of claim 1, wherein the step of separating the encrypted data set comprises using a deterministic technique.

Claim 3: The method of claim 1, wherein the step of separating the encrypted data set comprises using a substantially random technique.

Claim 5: The method of claim 1, wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of shares to have a substantially randomly distribution of the encrypted data set.

474. Orsini discloses the additional limitations of Claims 2, 3, and 5.

475. Orsini states that “[a] cryptographic split (cryptosplit) partitions the data into N number of shares.” EX1008 (Orsini), [0281]. The “data” in the context of ¶[0281] and Figure 21 of Orsini is an “*encrypted data set*.” See EX1008 (Orsini), Figure 21, [0296]. “The *partitioning* can be on any size unit of data ... as well as

any pattern or combination of data unit sizes whether *predetermined* [‘deterministic’] or *randomly generated* [‘*substantially random*’].” EX1008 (Orsini), [0281]. Further, “[t]he units are then *distributed* (either *randomly* [‘*substantially randomly* [sic] *distribution*’] or by a predetermined set of values) into the *N shares*.” EX1008 (Orsini), [0281]. Orsini also explains that “the distribution of the data units into the shares may be performed according to ... size-fixed, predetermined sizes, or one or more combination, pattern or sequence of data unit sizes that are *predetermined* or *randomly generated*.” EX1008 (Orsini), [0281]. *See also* EX1008 (Orsini), [0116]. By randomly distributing the data units of the encrypted data set—in the context of Figure 21—into shares, the “*shares [] have a substantially randomly* [sic] *distribution of the encrypted data set*.”

476. **In sum**, for Claim 2, Orsini discloses “*wherein the step of separating the encrypted data set comprises using a deterministic technique* [Orsini’s predetermined partitioning and/or distribution technique].”

477. **In sum**, for Claim 3, Orsini discloses “*wherein the step of separating the encrypted data set comprises using a substantially random technique* [Orsini’s random partitioning and/or distribution technique].”

478. **In sum**, for Claim 5, Orsini discloses “*wherein the step of separating the encrypted data into the plurality of shares comprises causing the plurality of*

shares to have a substantially random distribution of the encrypted data set [Orsini's random distribution technique]."

6. Claim 4: The method of claim 1, further comprising causing a plurality of data units in each of the shares to be rearranged relative to one another after the separating step.

479. Orsini discloses the additional limitation of Claim 4.

480. Orsini explains that "*the data units in each share could be shuffled* utilizing a different algorithm. This data unit shuffling may be performed as the original data is split into the data units, or after the data units are placed into the shares, or after the share is full, for example." EX1008 (Orsini), [0286]. *See also* EX1008 (Orsini), [0282]–[0285].

481. Notably, during the prosecution of U.S. App. No. 13/371,363 in the same patent family as the '802 Patent, applicant stated that the limitation "*causing a plurality of data units in each of the **shares** to be rearranged relative to one another after the separating step*" is supported by paragraphs 290 to 294 of the specification of '363 App., which correspond to paragraphs 282-286 of Orsini. *See* EX1011 ('363 App. File History), 561-562.

482. **In sum**, Orsini discloses "*causing a plurality of data units in each of the shares to be rearranged relative to one another after the separating step* [Orsini's data unit shuffling scheme]."

7. Claim 9: The method of claim 1, wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices.

483. Orsini discloses the additional limitation of Claim 9.

484. As discussed in §IX.E.4.vii above [Ground V, claim limitation 1F], Orsini describes storing shares across “*geographically separated independent data storage systems.*” EX1008 (Orsini), [0111]; [0379] (“one or more data depositories or storage devices”). Orsini also states that the different storage locations may include “one or *more* data depositories or *storage devices*,” such as “distinct and independent storage facilities *D1 through D4.*” EX1008 (Orsini), [0379] (“One advantage is the security gained from moving shares of the data to different locations on one or more data depositories or *storage devices*, that may be in different logical, *physical or geographical locations.*”). A POSITA would have known that most systems that distribute their storage nodes, like most distributed systems, are geographically separated. *See* §V.B.1 above.

485. **In sum**, Orsini discloses “*wherein the step of storing the shares in respective separate storage locations comprises storing the shares on at least two separate storage devices* [Orsini’s geographically separated independent data storage systems].”

8. **Claim 10: A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.**

486. Claim 10 is the same as claim 1 in all respects except for claim preamble [10Pre].

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

Claim 1	Claim 10
[1A]	[10A]
[1B]	[10B]
[1C]	[10C]
[1D]	[10D]
[1E]	[10E]
[1F]	[10F]
[1G]	[10G]

488. Therefore, for the same reasons discussed in §§IX.E.4.ii above—IX.E.4.viii above regarding claim limitations [1A]–[1G], the Orsini/Torre System also satisfies claim limitations [10A] through [10G], respectively.

- i. **[10Pre] A non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out a method for securing a data set, the method comprising the steps of**

489. To the extent the preamble is limiting, Orsini discloses or renders obvious claim preamble [10Pre].

490. As explained in §IX.E.4.i above [Ground V, Claim preamble 1Pre], Orsini describes “*a method for securing a data set.*”

491. Orsini also discloses that its method for securing a data set is implemented using “[*a non-transitory computer readable medium storing computer executable instructions that, when executed by at least one processor, cause a computer system to carry out*]” the described method. In particular, Figure 21 of Orsini details a “parser **software** suite” responsible for executing the steps outlined in that figure. EX1008 (Orsini), [0292]–[0293]. Orsini also explains that this method “may be incorporated into the operations of **any computer**, server, engine or the like, that is capable of executing the desired steps of the method.” EX1008 (Orsini), [0013]. *See also* EX1008 (Orsini), [0119], [0379] (referencing “data security methods and computer systems of the present invention”).

492. A POSITA would have understood—or at a minimum, would have found it obvious—that a computer includes at least one processor, and that the parser **software** suite shown in Figure 21 would be implemented using a non-transitory computer-readable medium containing executable code, run by that processor. *See* EX1003 (Torre), [0141], [0142], [0064]. *See also* §V.A above.

493. **In sum**, Orsini discloses or renders obvious “[a] *non-transitory computer readable medium storing computer executable instructions* [parser **software** suite] *that, when executed by at least one processor cause a computer system* [Orsini’s computer] *to carry out a method for securing a data set* [Orsini’s method for securing a data set].”

9. **Claim 19: A computer system for securing a data set, the system comprising: at least one processor; a non-transitory computer readable medium storing computer executable instructions that, when executed by the at least one processor, cause the computer system to carry out a method for securing a data set, the method comprising the steps of: creating integrity information using the data set; encrypting the data set based on an encryption key to produce an encrypted data set; generating data splitting information, wherein the data splitting information is usable to determine into which of a plurality of shares of data a unit of data of the encrypted data set will be placed; separating the encrypted data set into the plurality of shares based on the data splitting information; including in the plurality of shares data indicative of (a) the encryption key and (b) the integrity information; and causing the plurality of shares to be stored in respective separate storage locations; wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares.**

494. Claim 19 is substantively the same as claim 1, except for claim preamble [19Pre] and claim limitations [19A]–[19B].

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

Claim 1	Claim 19
[1A]	[19C]
[1B]	[19D]
[1C]	[19E]
[1D]	[19F]
[1E]	[19G]
[1F]	[19H]
[1G]	[19I]

496. Therefore, for the same reasons discussed §§IX.E.4.ii above—IX.E.4.viii above regarding claim limitations [1A]–[1G], the Orsini/Torre System also satisfies claim limitations [19C]–[19I], respectively.

497. Additionally, Orsini discloses—or at least renders obvious—claim preamble [19Pre] and claim limitations [19A]–[19B] for the same reasons previously provided for claim preamble [10Pre] (§IX.E.8.i above).

10. Claims 11-14, 18, 20-23, and 27

498. The Orsini/Torre combination renders obvious Claims 11-14, 18, 20-23, and 27.

- The Orsini/Torre combination teaches the additional limitations of claims 11 and 20, for the same reasons discussed in §IX.E.5 above [Ground V, Claims 2, 3, and 5] with respect to claims 2.
- The Orsini/Torre combination teaches the additional limitations of claims 12 and 21 for the same reasons discussed in §IX.E.5 above [Ground V, Claims 2, 3, and 5] with respect to claim 3.
- The Orsini/Torre combination teaches the additional limitations of claims 14 and 23 for the same reasons discussed in §IX.E.5 above [Ground V, Claims 2, 3, and 5] with respect to claim 5.
- The Orsini/Torre combination also teaches the additional limitations of claims 13 and 22, as explained in §IX.E.6 above [Ground V, Claim 4].
- The Orsini/Torre combination also teaches the additional limitations of claims 18 and 27, as explained in §IX.E.7 above [Ground V, Claim 9].

F. Ground VI: Orsini, Torre, And Krawczyk Render Obvious Claims 6, 15, and 24.

499. Claims 6, 15, and 24 would have been obvious in view of Orsini, Torre, and Krawczyk.

500. In this Ground, I introduce illustrations to show how the Orsini/Torre/Krawczyk combination renders obvious claims 6, 15, and 24.

1. Orsini/Torre/Krawczyk Color-Coding Scheme

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

Color	Disclosure in Orsini, Torre or Krawczyk
light blue	Orsini's splitting of encrypted data using session master key
dark orange	Orsini's splitting of session master key
light amber	Orsini's encrypted data share
magenta	Orsini's session master key or its shares
lime green	Orsini's encrypted input data
purple	Torre's SHA-1 digital signature
forest green	Collection of encrypted input data and SHA-1 digital signature
royal blue	Krawczyk's "Shamir's secret sharing algorithm"
red	Orsini's shares

2. Claim 6: The method of claim 1, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.

502. Orsini/Torre in combination with Krawczyk teaches the additional limitation of claim [6].

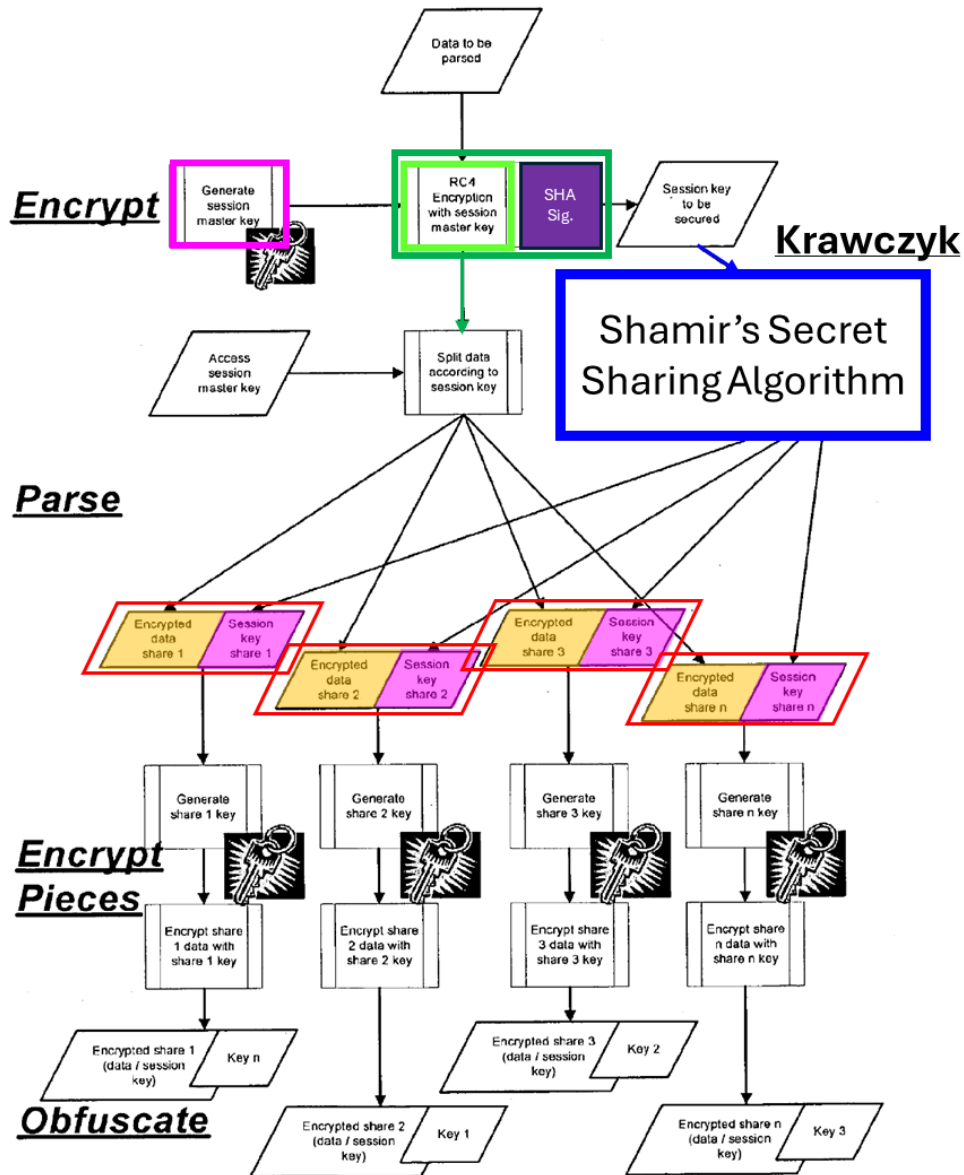
503. Orsini describes embedding information related to the encryption key into the shares by splitting the key using a parser master key. EX1008 (Orsini), [0297]. However, a POSITA would have recognized that this form of key splitting is less secure, as it could result in direct storage of identifiable fragments of the encryption key.

504. Krawczyk, by contrast, offers a different approach for distributing the encryption key among the shares—specifically, through the use of “a perfect (n, m) -secret sharing scheme (e.g. **Shamir’s**).” EX1006 (Krawczyk), 13. *See* §IX.C.2 above [Krawczyk].

505. It would have been obvious to apply Shamir’s scheme to divide the encryption key into shares and to attach those resulting portions to the data shares. This is depicted in Orsini’s Figure 21, shown and annotated below using the same color-coding scheme as that described in the table in §IX.F.1 above. The processed portions of the encryption key appended to the shares form “*the data indicative of the encryption key ... created using a Shamir algorithm.*”

Orsini / Torre/Krawczyk System

Figure 21



EX1008 (Orsini), FIG. 21 (annotated).

3. Reasons to Combine Orsini/Torre with Krawczyk

506. A POSITA would have been motivated to combine Orsini/Torre and Krawczyk as discussed in §IX.F.2 above for at least the following two reasons.

507. **First**, a POSITA would have been motivated to combine the teachings of Orsini/Torre with those of Krawczyk to enhance encryption key security. It would have been well understood that safeguarding the encryption key is critical to preventing unauthorized access to the encrypted data. *See* §V.C above. Thus, a POSITA would have been motivated to strengthen the security of the encryption key embedded within the shares. And Krawczyk specifically discloses such a technique: “*a perfect (n, m)-secret sharing scheme (e.g. Shamir’s)*” that can be used to generate “n shares for the key K,” which are then appended to the corresponding n shares of encrypted content. EX1006 (Krawczyk), 13. *See also* EX1006 (Krawczyk), 13 (“this is essential in Shamir’s scheme to provide *perfect secrecy*”).

508. **Second**, the combination would have merely amounted to using a known technique (Shamir’s algorithm) to improve a similar device (Orsini/Torre System) in the same way (applying Shamir’s algorithm to the encryption key to create n shares and appending the n shares of tag to n shares of encrypted data as taught by Krawczyk) to yield predictable results (*e.g.*, improving the security of the encryption key).

509. Next, a POSITA would have had a reasonable expectation of success in combining the Orsini/Torre System with Krawczyk for at least the following two reasons.

510. **First**, both systems are directed to the same underlying challenge: securely distributing secret information. Each discloses a substantially similar approach, involving the splitting of encrypted data into shares and distributing those shares alongside information relating to the encryption key. *Compare* EX1003 (Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and scalability and adaptability to resources*.”), [0051]–[0052] (“In the process of shredding, input data is generally *transformed into many smaller pieces of data called shreds*. ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of data (step 134)*.”) and EX1008 (Orsini), Abstract (“The present invention provides a method and system for securing sensitive data from unauthorized access or use.”),

[0278]–[0280] (“In a separate embodiment, the present invention comprises a complete system for performing the data securing methods of the present invention on any data set. The computer system of this embodiment comprises a data splitting module that comprises the functionality shown in FIG. 8 and described herein.”), [0291] (“The data splitting or parser modules of the present invention may be integrated into a wide variety of infrastructure products or applications as desired.”) *with* EX1006 (Krawczyk), 12-13 (“For completeness, we outline the following simple information dispersal scheme based on Reed Solomon erasure codes. ... We proceed to show how to build a space efficient secret sharing scheme.”).

511. Additionally, both the Orsini/Torre System and Krawczyk’s method incorporate redundancy to enable reconstruction of the original data even if not all shares are available. *Compare* EX1003 (Torre), [0051] (“Shredding is done in a manner that adds redundancy to the shreds so that the input data can be recovered from a fewer number of shreds than the total number stored.”) *and* EX1008 (Orsini), [0283]–[0290] (“In another embodiment of the cryptosplit process of the present invention, an option for the cryptosplitting process is to provide sufficient redundancy in the shares such that only a subset of the shares are needed to reassemble or restore the data to its original or useable form.”) *with* EX1006 (Krawczyk), 12-13 (“For completeness, we outline the following simple information dispersal scheme based on Reed Solomon erasure codes. ... We proceed to show

how to build a space efficient secret sharing scheme.”). A POSITA would have recognized that Krawczyk’s erasure codes are a form of error-correcting codes that add some redundancy to data, to permit recovery from certain errors. *See* §V.D above.

512. As a result, the combination would have merely amounted to applying a known technique (applying Shamir’s secret-sharing scheme to the encryption key and appending the outcome of the secret-sharing scheme to shares, as taught by Krawczyk) to a known device (Orsini/Torre System) ready for improvement (for improved security of the tag) to yield predictable results (improving the key’s security).

513. ***Second***, a POSITA would have recognized that incorporating Shamir’s secret-sharing scheme into the Orsini/Torre framework would have simply involved implementing the corresponding functionality in software, by writing program code. That task would have been well within the abilities of a POSITA. *See* EX1003 (Torre), [0141].

514. Thus, modifying the Orsini/Torre system with Krawczyk’s secret-sharing scheme would have been simple and would not have required extensive experimentation or imposed undue burden, because all three references address secure data distribution and redundancy. For the same reasons, the combination would have yielded predictable results (because applying Shamir’s secret sharing to

encryption keys is a well-known way to enhance key security). Lastly, a POSITA would have had a reasonable expectation of success because Krawczyk merely discloses a familiar cryptographic technique that could be implemented in software using standard programming practices, as Torre expressly contemplates.

515. Accordingly, the combination of Orsini/Torre with Krawczyk makes the additional limitation of claim [6] obvious.

516. **In sum**, Orsini/Torre in combination with Krawczyk teaches “*wherein the data indicative of the encryption key comprises data created using a Shamir algorithm* [Krawczyk’s teaching of using Shamir’s perfect (n, m)-secret sharing scheme].”

4. **Claim 15: The non-transitory computer readable medium of claim 10, wherein the data indicative of the encryption key comprises data created using a Shamir algorithm.**

Claim 24: The system of claim 19, wherein data indicative of the encryption key comprises data that was created using a Shamir algorithm.

517. The additional limitation in claims 15 and 24 is materially the same as that of claim 6. Therefore, for the same reasons discussed in §IX.F.2 above [Ground V, Claim 6], the combination of Orsini, Torre, and Krawczyk renders claims 15 and 24 obvious.

G. Ground VII: Orsini, Torre, And Foster Render Obvious Claims 7-8, 16-17, and 25-26.

518. Claims 7-8, 16-17, and 25-26 would have been obvious in view of Orsini, Torre, and Foster.

519. In this Ground, I introduce illustrations to show how the Orsini/Torre/Foster combination renders Claims 7-8, 16-17, and 25-26 obvious.

1. Orsini/Torre/Foster Color-Coding Scheme

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

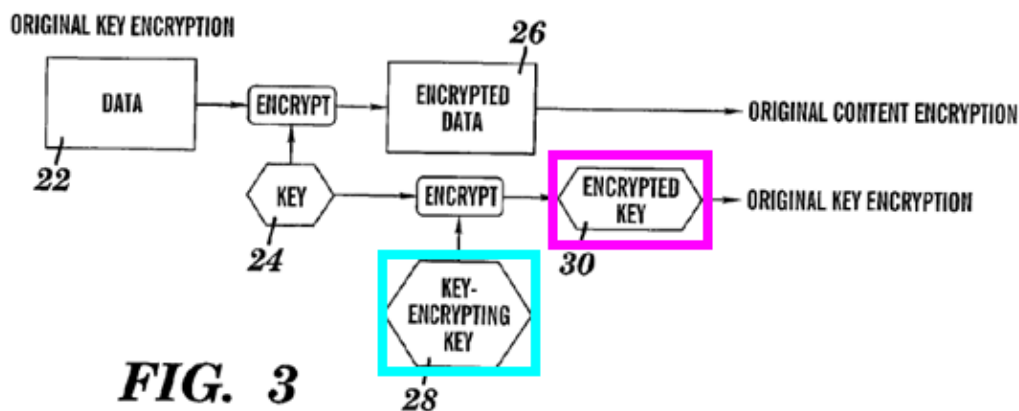
Color	Disclosure in Orsini, Torre or Foster
light blue	Orsini's splitting of encrypted data using session master key
dark orange	Orsini's splitting of session master key
light amber	Orsini's encrypted data share
magenta	Orsini's session master key or its shares
lime green	Orsini's encrypted input data
purple	Torre's SHA-1 digital signature
forest green	Collection of encrypted input data and SHA-1 digital signature
red	Orsini's shares
cyan	Foster's key encrypting key
royal blue	Foster's key encrypting step

2. Orsini/Torre In Combination With Foster ("Orsini/Torre/Foster")

521. As discussed in §IX.F.2 above [Ground V, Claim 6], a POSITA would have recognized that the key-splitting method employed in Orsini may be vulnerable

to security breaches because portions of the encryption key are stored directly, without additional protective measures.

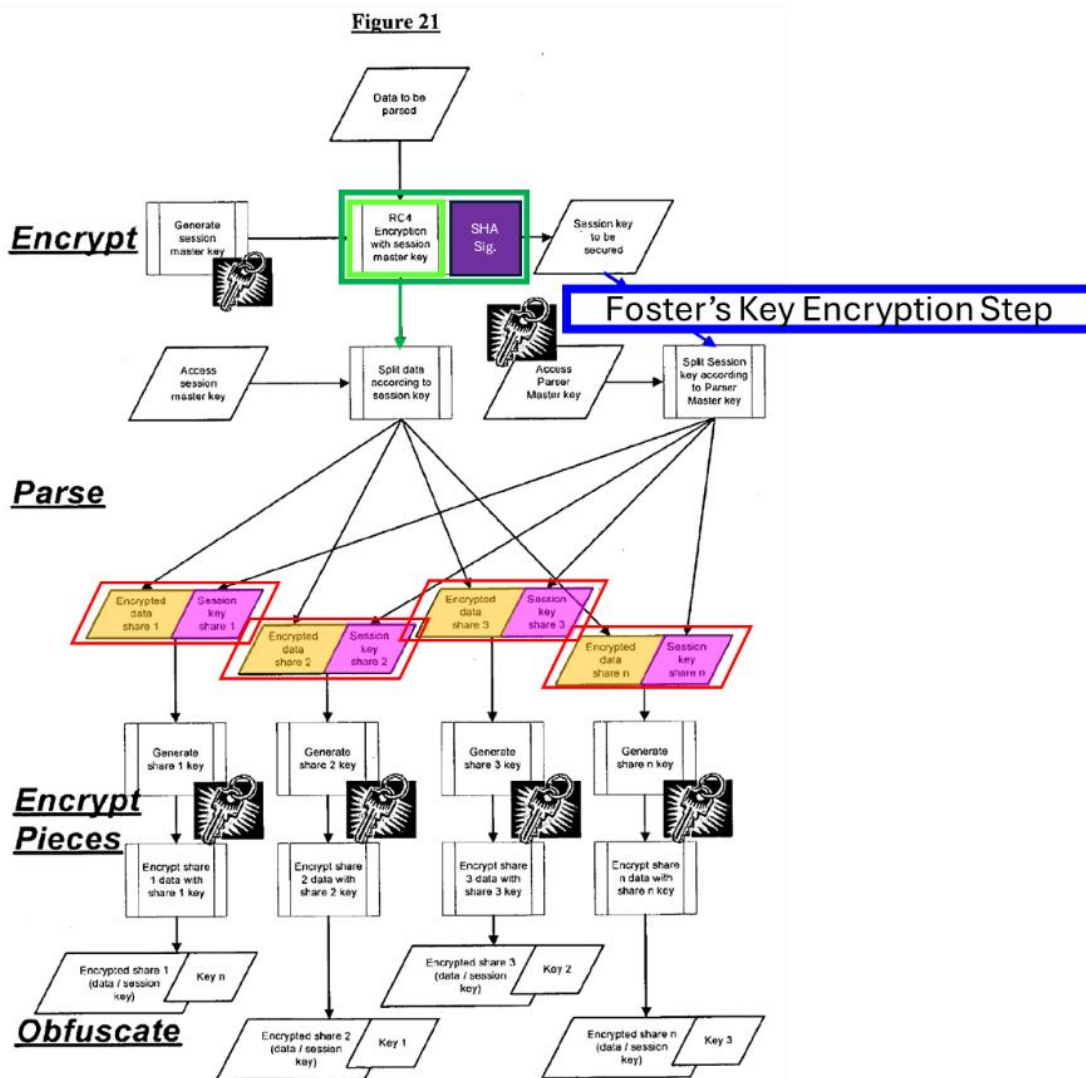
522. It would have been obvious to a POSITA to combine Orsini/Torre with Foster in order to enhance the security of the encryption key by encrypting it with a “key encryption key” prior to storage. EX1007 (Foster), [0053], Figure 3 (annotated below). More specifically, it would have been apparent to integrate Foster’s encryption approach—depicted in Figure 3—into the Orsini/Torre system to: (1) apply a **key encrypting key** to encrypt the encryption key, and (2) store the resulting **encrypted encryption key** instead of saving the key in its original form. EX1007 (Foster), [0053]. This is shown in Foster’s Figure 3 below, annotated using the same color-coding scheme as that described in the table in §IX.G.1 above.



EX1007 (Foster), Figure 3 (annotated).

523. A visual depiction of the combination of Orsini, Torre and Foster (“Orsini/Torre/Foster”) is shown and annotated below using the same color-coding scheme as that described in the table in §IX.G.1 above.

Orsini / Torre/Foster System



EX1008 (Orsini), FIG. 21 (annotated).

i. Reasons To Combine Orsini/Torre With Foster

524. A POSITA would have been motivated to combine Orsini/Torre with Foster to enhance the security of the encryption key for at least the following two reasons.

525. **First**, it would have been well understood that safeguarding the encryption key is essential to prevent unauthorized access to the encrypted data. *See* §V.C above. Therefore, a POSITA would have been inclined to adopt Foster's approach of encrypting the encryption key prior to storage. EX1007 (Foster), [0053].

526. **Second**, the combination would have merely amounted to applying a known technique (Foster's key encryption scheme) to a known device (the Orsini/Torre System) ready for improvement (for improved security of the stored encryption key) to yield predictable results (improving the stored encryption key's security).

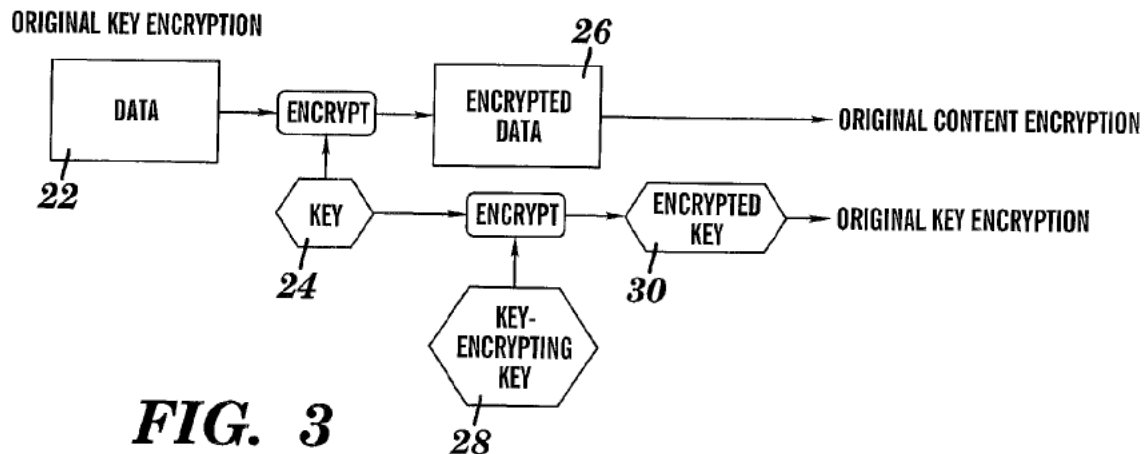
527. Next, a POSITA would have had a reasonable expectation of success in combining Orsini/Torre system with Foster for at least the following two reasons.

528. **First**, both Orsini/Torre and Foster address the same technical challenge—secure distribution of secret information—and propose substantially similar solutions involving splitting encrypted data into shares or packets, and distributing these along with information on the encryption key. *Compare* EX1003

(Torre), [0049] (“A system and method for shredding and deshredding is described herein to store and retrieve data while emphasizing considerations such as redundancy, integrity, and *security of data and scalability and adaptability to resources.*”), [0051]–[0052] (“In the process of shredding, input data is generally *transformed into many smaller pieces of data called shreds.* ... Storage of the shredded input data ... may involve sending it ... to multiple storage units such as multiple memories on a single computer or multiple computers, multiple hard disks on a single server or multiple servers, or multiple storage devices, other than servers, directly attached to a network.”), [0061]–[0062] (“The step of shredding 124 of method 120 and shown in FIG. 3 is further elaborated in a method 130 as shown in FIG. 4. After receiving the input data (step 122), the method goes on to perform the shredding step 124. *The shredding step 124 performs first transformations (step 132), allocates data transformed by the first transformations into various different blocks of data (step 133), and performs second transformations on these blocks of data (step 134).*”) and EX1008 (Orsini), Abstract (“The present invention provides a method and system for securing sensitive data from unauthorized access or use.”), [0278]–[0280] (“In a separate embodiment, the present invention comprises a complete system for performing the data securing methods of the present invention on any data set. The computer system of this embodiment comprises a data splitting module that comprises the functionality shown in FIG. 8 and described herein.”),

[0291] (“The data splitting or parser modules of the present invention may be integrated into a wide variety of infrastructure products or applications as desired.”) *with* EX1007 (Foster), [0053] (“Referring to FIG. 3, the encryption of content 22 using title key 24 is depicted in greater detail. As shown, content 22 is encrypted with title key 24 to yield encrypted content 26. Then, title key 24 is itself encrypted with key encrypting key 28. The encrypted title key 30 can then be transmitted along with the encrypted content 26 to a recipient.”).

529. Incorporating Foster’s approach would simply require adding an extra encryption step to encrypt the encryption key before embedding it in the tag, as illustrated in Foster’s Figure 3 below. Therefore, the combination would have merely amounted to using a known technique (Foster’s key encryption scheme) to improve a similar device (Orsini/Torre System for securely storing input data) in the same way (encrypting the encryption key as taught by Foster) to yield predictable results (*e.g.*, improving the security of the stored encryption key).



EX1007 (Foster), Figure 3.

530. *Second*, a POSITA would have recognized that incorporating Foster's key encryption method into the Orsini/Torre system would have simply involved implementing the corresponding functionality in software, by writing program code and following Foster's steps. That task would have been well within the abilities of a POSITA. *See* EX1003 (Torre), [0141].

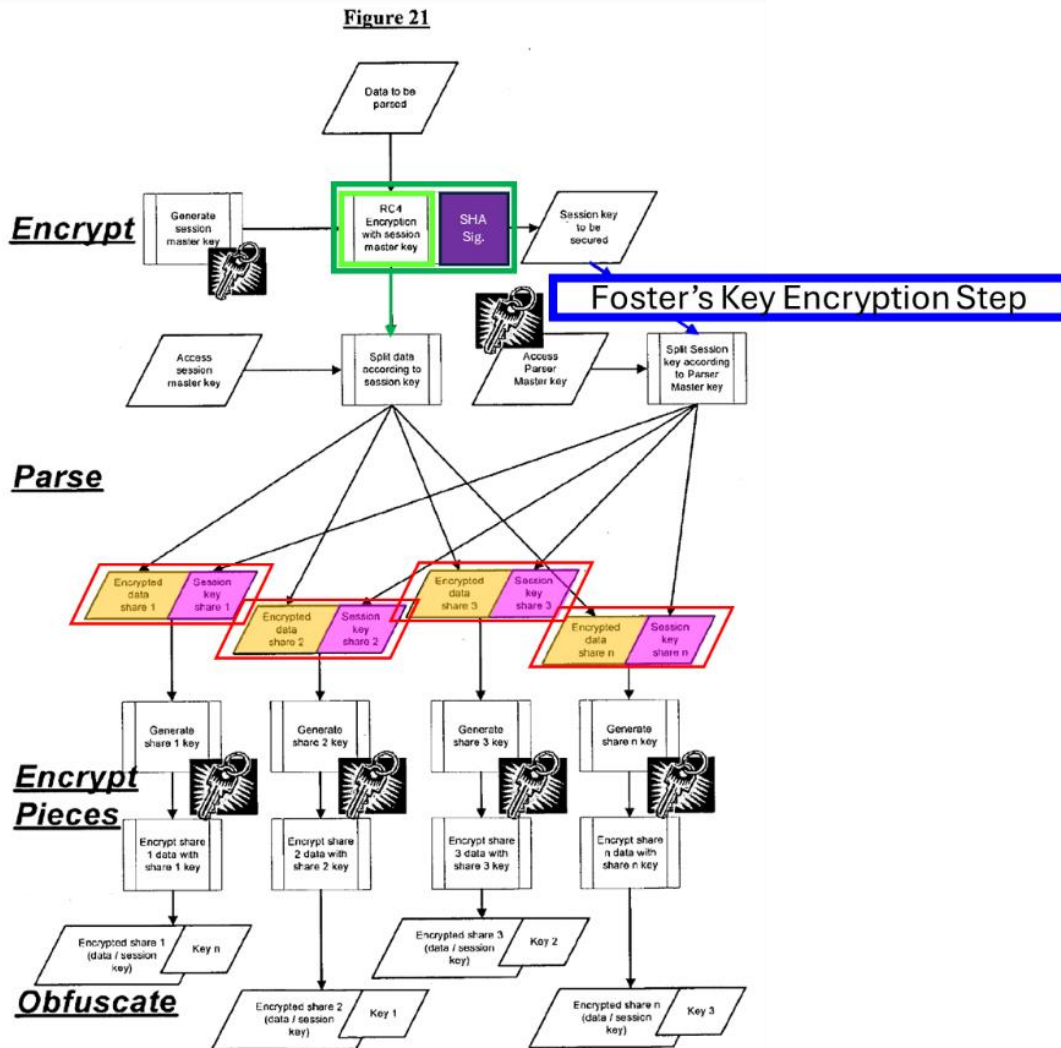
531. Thus, modifying the Orsini/Torre system with Foster's key encryption method would have been simple and would not have required extensive experimentation or imposed undue burden, because all three references address securing sensitive information and disclose encrypting and distributing encryption keys. For the same reasons, the combination would have yielded predictable results (because encrypting the encryption key before storage is a known and reliable way to improve key security). Lastly, a POSITA would have had a reasonable expectation of success because Foster's approach simply adds an extra encryption

step—an easily implemented software modification well within the ordinary skill in the art.

3. **Claim 7: The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.**
 - i. **[7A] The method of claim 1, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a **second key**; and**
 - ii. **[7B] including in the **plurality of shares** data indicative of the **encrypted encryption key**.**

532. Orsini/Torre in view of Foster teaches the additional claim limitations of [7A]-[7B]. The Orsini/Torre/Foster System *encrypts the encryption key using a **key encrypting key*** and includes the resulting **encrypted encryption key** in the tag. EX1007 (Foster), [0053]. See §IX.C.2 above [Orsini/Torre/Foster], Orsini's annotated Figure 21 below, which is shown and annotated using the same color-coding scheme as that described in the table in §IX.G.1 above.

Orsini / Torre/Foster System



EX1008 (Orsini), FIG. 21 (annotated).

533. Next, the system processes the **encrypted data** along with the **encrypted encryption key** by parsing and dividing them into **shreds**, as described in §IX.E.4.vi above [Ground V, claim limitation 1E]. As a result, the Orsini/Torre/Foster combination includes in the **shreds** (“*plurality of shares*”) portions of the **encrypted encryption key** (“*data indicative of the encrypted encryption key*”).

534. **In sum**, Orsini/Torre in combination with Foster teaches “*wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key [Orsini’s session master key] with a second key [Foster’s key encryption key]; and including in the plurality of shares [Orsini’s shares] data indicative of the encrypted encryption key [encrypted encryption key].*”

4. **Claim 8: The method of claim 7 further comprising the method step of storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.**

535. The Orsini/Torre/Foster System teaches the additional limitation of claim 8.

536. **First**, Orsini/Torre/Foster teaches “*storing the second key outside of the plurality of shares.*” In the Orsini/Torre/Foster System, the data recovery process relies on a key encrypting key to decrypt the encrypted encryption key. EX1007 (Foster), [0055], [0061]. That key encrypting key is stored separately and is cached in memory. EX1007 (Foster), [0058] (“[A] receiver may cache multiple KMBs (or the corresponding key encrypting keys) in memory”). Therefore, the Orsini/Torre/Foster System satisfies this limitation by storing the key encrypting key (“second key”) outside the shreds (“plurality of shares”).

537. **Second**, the Orsini/Torre/Foster combination discloses that “*the data set is restorable by accessing less than all, but at least a threshold number of, the*

plurality of shares,” as detailed in §IX.E.4.viii above [Ground V, claim limitation 1G].

538. **Third**, the Orsini/Torre/Foster combination teaches accessing and utilizing the **key encrypting key** (“*second key*”) to reconstruct the input data (“*data set*”). Orsini states that “[t]o restore the original data format, the steps are reversed.” EX1008 (Orsini), [0300]. Accordingly, in this combined system, the **key encrypting key** is accessed and used to decrypt the **encrypted encryption key**, which is subsequently used to decrypt the **encrypted data**—thereby recovering the original data set. EX1007 (Foster), [0055], [0058], [0061]. As a result, “*the data set is restorable by accessing ... the second key.*”

539. **In sum**, Torre/Tajima in combination with Foster teaches “*the method step of storing the second key [Foster’s key encrypting key] outside of the plurality of shares [Orsini’s shares], wherein the data set [Orsini’s data set] is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key [Foster’s key encrypting key].*”

5. **Claim 16: The non-transitory computer readable medium of claim 10, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.**

Claim 17: The non-transitory computer readable medium of claim 16 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

Claim 25: The system of claim 19, wherein the step of including data indicative of the encryption key comprises: encrypting the encryption key with a second key; and including in the plurality of shares data indicative of the encrypted encryption key.

Claim 26: The system of claim 25 wherein the method further comprises storing the second key outside of the plurality of shares, wherein the data set is restorable by accessing less than all, but at least a threshold number of, the plurality of shares, and the second key.

540. The additional limitation in claims 16 and 25 is materially the same as that of claim 7, and the additional limitation in claims 17 and 26 is materially the same as that of claim 8. Therefore, for the same reasons discussed with respect to claims 7 (§IX.G.3 above) and 8 (§IX.G.4 above), the Orsini/Torre/Foster combination renders claims 16–17 and 25–26 obvious, respectively.

X. Conclusion

541. In my opinion, the challenged claims 1-27 of the '802 Patent are unpatentable.

XI. Availability for Cross-Examination

542. 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

543. 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

544. 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.

Declaration of Dr. Erez Zadok
U.S. Patent No. 8,271,802

A handwritten signature in black ink, appearing to read "Erez Zadok", written in a cursive style.

July 9, 2025

Dr. Erez Zadok