Data Privacy in the Public Cloud (Ozel)

Holloway Competition – University of New Hampshire

Alex MacPherson, Bryan Bickford, Julia Irwin, Ken Graf(advisor)

{agk23, bsq4, jmx93, kmh722} @wildcats.unh.edu

This paper describes technical detail of processes to provide private data in the public cloud. The process is based on using multiple cloud vendors in such a way that no single vendor is relied upon. The concept is named Ozel (Turkish for private/confidential). In brief Ozel leverages established math and Forward Error Correction (FEC) to securely scatter fragments of an object, such that any entity possessing any single fragment cannot determine any portion of the original object.

Keywords  Trust, Privacy, Cloud, Distributed Storage, Encryption, Security, Collaboration, Internet enabled

1. What is meant by Data Privacy in the Public Cloud?

This concept of private data storage in the cloud is based on the fundamental belief that ubiquitous high speed internet, powerful mobile/embedded devices, inexpensive reliable cloud storage and the need for security are driving the desire for compute resources to be easier to manage, predictable, trustworthy and private.

*“The Internet should have required that each endpoint on the network be able to authenticate itself”* Dr. Vint Cerf

*“The rapidly increasing number of security breaches and data loss incidents highlights the need to shift the focus from locking down devices and infrastructure to protecting sensitive data itself.”* Enrique Salem, Symantec (CEO)

*“The prevalence and dynamic nature of online applications is creating new challenges for meeting compliance and security need*s*.”* Brian Trustowski, IBM/ISS (GM)

*“The architecture of the future requires research and development that focuses on game-changing technologies that could enhance the security, reliability, resilience and trustworthiness of our digital infrastructure.”* Melissa Hathaway, Director of USA Cyberspace, at RSA ‘09

The title for this concept went through a number of iterations. Let us consider each term in detail.

**Data**: Files are the typical data object. While Ozel can work with files, in Ozel a data object can be any resource that has a URI that can be resolved to a set of bytes. Ozel data objects are independent and have no relationship to other objects.

**Privacy**: Ozel allows you to trust no one, including the authors of Ozel. Cloud providers can be subpoenaed. Communications can be monitored. Access controls compromised. In Ozel there is no key management or the reliance on any one vendor/provider to do the right thing.

**Public Cloud**: Amazon, Google, Dropbox, et.al. Anyone that provides services hosted or outsourced via the Internet. The Internet brings us the ability to locate and use remote resources, standards for exchanging both processes and data. The Public Cloud allows us to reliably coordinate multiple independent vendors and services.

1. What are we trying to achieve?
   1. Allow public clouds to be private

Cloud vendors (e.g. Dropbox, Ubuntu 1, SkyDrive, Amazon EC2) offer cheap, reliable and secure storage. Many vendors go to substantial lengths to provide trustworthy services using the latest authentication, access control, policies and operational controls. Generally the service level exceeds what the customer can provide in their own facility. So what is the concern?

For example: Assume an attorney storing client information in the cloud. The cloud vendor is a 3rd party that can be required by law to disclose information that otherwise might be protected by attorney-client privilege. Doctors, accounts and other professionals face similar challenges.

Additionally, government regulations and industry compliance standards limit the use of public cloud services solely because privacy cannot be maintained.

* 1. No reliance on any single cloud vendor

Ozel fragment is based on an N of M distribution model, in this M fragments are generated but only N fragments are needed to recreate the original object. This means the M-N fragments can be loss, destroyed, discarded or unavailable but the original data can still be attained. The user selects the M and N values. Given we can “lose” a subset of fragments no single cloud vendor is required to present their fragment.

* 1. Eliminate encryption key management

All encryption keys in Ozel are transitory. This means a key is generated during the fragmentation process, and then distributed across the fragments such that no single fragment contains the encryption key. After fragmentation the key is destroyed. The object encryption key is never persisted in any manner, thus there is no key management required.

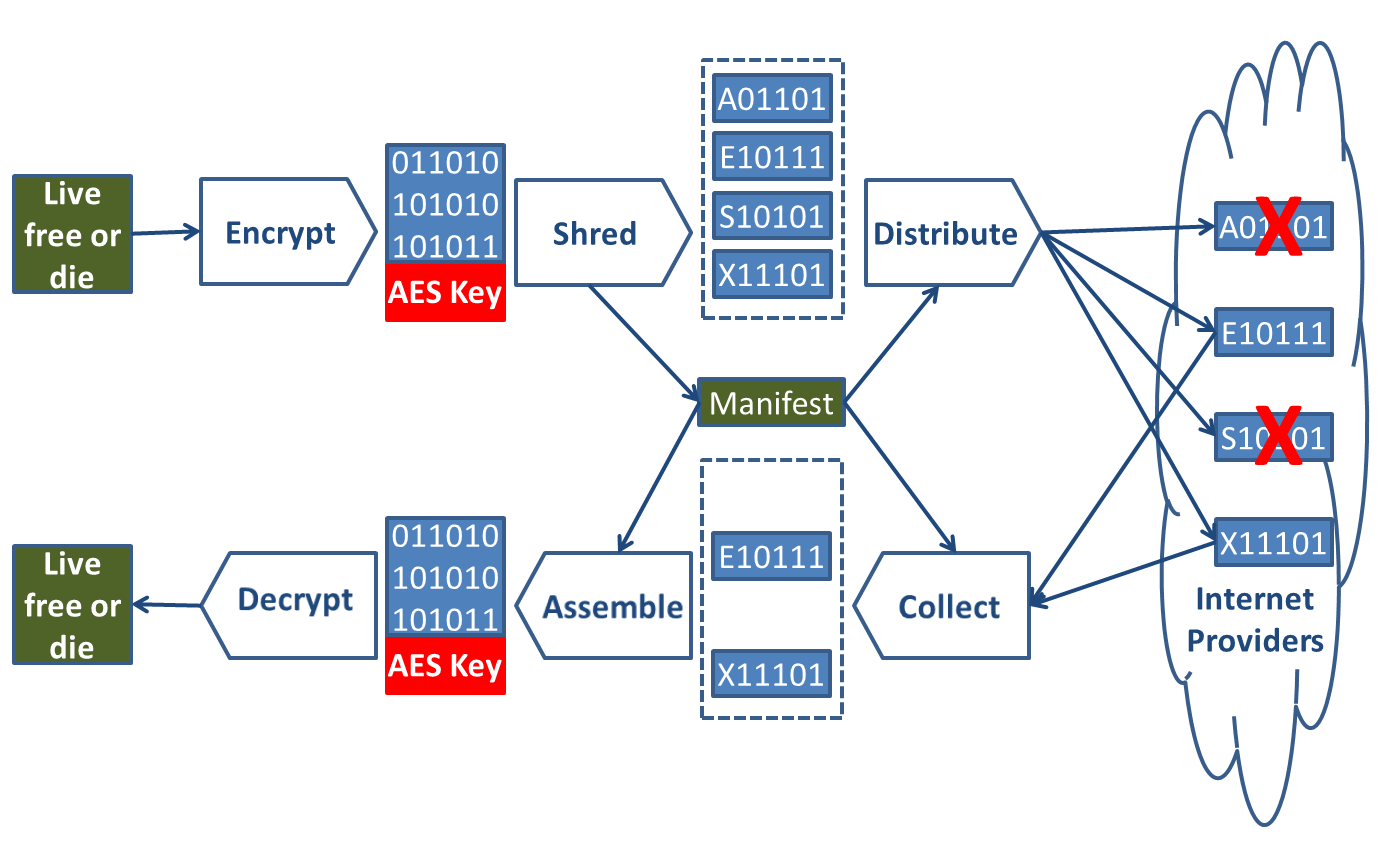
* 1. Access objects from any location

Once the fragments are distributed a manifest is maintained by Ozel to manage how fragments can be reassembled back into the original object. The manifest typically is a few hundred bytes per object. Thus manifests containing millions of objects can easy be transported via the internet or physically maintained on a USB drive.

* 1. Share Access to objects

As an advanced feature of Ozel we plan on allowing access control and sharing of objects between groups of users my adding additional controls to the management of the manifest. This additional feature is targeted to paid professional and business users.

1. The Ozel Process

Figure 1 shows the Ozel process flow. Ozel uses established mathematics—Galois fields, Vandermonde matrices and AES encryption—to shred the object data and keys into a set of fragments such that only a random subset of the original fragments is needed to reconstruct the data. The fragmentation is combined with the abstract nature of the Internet to securely scatter the fragments to the four corners of world. Ozel’s innovation is the reconstruction of the fragments based solely on the user controlled manifest and not the location of the original object or the scattered fragments.

* 1. Encode Phase

The encoding phase takes an existing object encrypts and then distributes fragments to the cloud vendor authorized by the user.

Figure 1: Ozel Encode and Decode process steps

* + 1. Encryption step

Input: Set of bytes that is the Object.

Process: The object bytes are encrypted with a random symmetric key generated by Ozel for this object. The encryption is randomly generated and based on AES256.

Output: Encrypted set of bytes and encryption key

* + 1. Shred step

The output of the encryption step is evenly divided into N number of fragments. The value of N is user specified. The encryption key is destroyed after fragmentation is complete.

* + 1. Distribute step

Using the published cloud vendor storage APIs the fragments are persisted to the cloud and the manifest is updated to allow later retrieval in the decoding phase. Most cloud vendors present a RESTful API using authentication based on OAuth. User credentials are managed by the cloud vendor and are not known to Ozel.

* 1. Decode Phase

The Decode phase is the reverse of the encode phase; collecting and assembling fragments (only M of N are needed) then decrypting to reconstruct the original object.

* + 1. Collection step

Authenticated RESTful retrieval of fragments is attempted until M fragments are located. The hash value of each retrieved fragment is validated.

* + 1. Assemble step

Fragments are ordered and the AES encryption key is retrieved.

* + 1. Decryption step

The ordered fragments are decrypted and the encryption key is destroyed. The hash value of the original object stored in the manifest is validated.

1. Threat Impact

Figure 2 is a STRIDE based threat model for an individual Ozel user. There is only one trust boundary between the user’s machine and the fragments stored in the cloud.

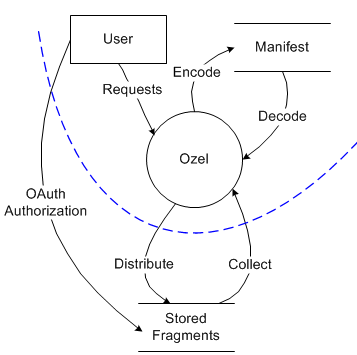


Figure 2: Ozel Single User Threat Model

* 1. Cloud storage vendor compromised

Through Ozel all data given to any vendor is encrypted. Fragments do not contain any information about any other fragment. Ozel allows for the “loss” of some fragments. With loss defined as either temporarily missed SLA or permanent erasure. Thus there is no reliance on any single vendor to return the fragments they were provided nor the need for strict SLAs.

* 1. An attacker could poll all Cloud storage vendors

An attacker could poll all possible storage vendors for data related to a specific user (object URIs, creation/update times, ownership, access control lists, etc.). The cloud vendors provide authenticated RESTful API to limit access to properly authenticated users. Authenticated users and proper legal authorities can based on the presence of the encryption key in the fragments attempt permutations of all fragments to reconstruct the original object.

* 1. An attacker could sniff communications

All communication is based on SSL to prevent eavesdropping by an attacker.

* 1. Poor encryption used

Ozel does not support any symmetric encryption strength less than AES 256. Ozel limits the use of PKI to communications (SSL).

* 1. Disclosure of object encryption key

The object encryption key is never persisted in its entirety or made available to the user, the manifest or the cloud storage vendor. The encryption key is broken up and distributed in the fragments. Thus a portion of the keys is distributed in some fragments. This approach reduces the overall entropy of the key strength but does not compromise the entire key.

* 1. Disclosure of manifest

Disclosing the manifest does allow for retrieval of fragments and assembly of the object, provided the attacker has access to the fragment storage and can authenticate as the object owner. Authentication information is not maintained in the manifest.

* 1. Fragments are altered in flight or by storage vendor

The manifest maintains hashes for all fragments and the original object. The hashes are validated during the decoding process

Professional and Corporate version of Ozel will allow sharing access to objects by sharing access to a common manifest. This requires Ozel to be running as a shared service under the control of the object owners. The STRIDE threat model is shown in Figure 3.

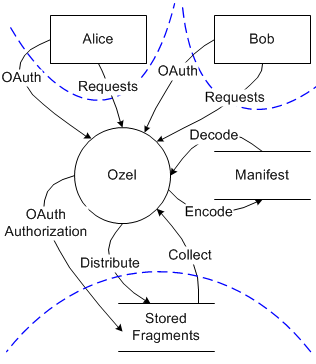


Figure 3: Ozel server based threat model

Server version threats.

* 1. Server admin users have access to the manifest

The manifest is encrypted for authorized users only.

* 1. Access to manifest objects must be controlled

RBAC model for manifest access will be implemented.

* 1. Who has CRUD rights for stored fragments?

A single set of common credentials will be applied for the fragment storage operations.

T.T. Chung, "An Automatic Data Segmentation Method for 3D Measured Data Points," National Taiwan University, 1998.

Daniel Ellard , James Megquier, DISP: Practical, efficient, secure and fault-tolerant distributed data storage, ACM Transactions on Storage (TOS), v.1 n.1, p.71-94, 2005

Indrajit Ray , Eunjong Kim , Ross McConnell , Dan Massey, Reliably, securely and efficiently distributing electronic content using multicasting, Proceedings of the 6th international conference on E-Commerce and Web Technologies, August 23-26, 2005, Copenhagen, Denmark

Giampaolo Bella , Costantino Pistagna , Salvatore Riccobene, Distributed Backup through Information Dispersal, Electronic Notes in Theoretical Computer Science (ENTCS), 142, p.63-77, January, 2006

Microsoft, Real-Time Streaming Protocol (RTSP) Windows Media Extensions, 2009, [http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-RTSP].pdf](http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5bMS-RTSP%5d.pdf) .

Julio Araujo , Frédéric Giroire , Julian Monteiro, Hybrid approaches for distributed storage systems, Proceedings of the 4th international conference on Data management in grid and peer-to-peer systems, September 01-02, 2011, Toulouse, France

Michael Rabin, “Probabilistic Algorithms In Finite Fields”, SIAM J. Computing, vol 9, pp 273-280, 1980

Michael Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance", Journal of the Association for Computing Machinery, vol. 36, No. 2, April pp. 335-348, 1989.

Jason K. Resch , James S. Plank, AONT-RS: blending security and performance in dispersed storage systems, Proceedings of the 9th USENIX conference on File and storage technologies, p.14-14, 2011

Rosenberg, J., and Schulzrinne, H., "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, 1999, <http://www.ietf.org/rfc/rfc2733.txt>

Alexandre Sora , Jérôme Lacan, FNT-based Reed-Solomon erasure codes, Proceedings of the 7th IEEE conference on Consumer communications and networking conference, p.466-470, 2010

Adi Shamir, "How to Share a Secret" Communications of the ACM, vol. 22, No. 11. 1979.

Wicker, B., "Reed-Solomon Codes and Their Applications", Wiley-IEEE Press, 1999, ISBN: 0780353919.

Appendix: Ozel fragmentation process

Text to be protected: “Ozel=>Privacy”

AES 256 bit key: 48 0e 05 9f 39 f7 b4 b9 81 68 4b e8 19 f1 8a d1

Encrypted text (SUN JCE): c1 15 cb 66 33 9a 87 8d c3 2d dd 63 8c 14 30 45

A single predefined finite field is always used: The tables for log and exp Galois field GF(28) are shown in hexadecimal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **exp()** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **0** | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | 1d | 3a | 74 | e8 | Cd | 87 | 13 | 26 |
| **1** | 4c | 98 | 2d | 5a | b4 | 75 | ea | c9 | 8f | 03 | 06 | 0c | 18 | 30 | 60 | c0 |
| **2** | 9d | 27 | 4e | 9c | 25 | 4a | 94 | 35 | 6a | d4 | b5 | 77 | Ee | c1 | 9f | 23 |
| **3** | 46 | 8c | 05 | 0a | 14 | 28 | 50 | a0 | 5d | ba | 69 | d2 | b9 | 6f | de | a1 |
| **4** | 5f | be | 61 | c2 | 99 | 2f | 5e | bc | 65 | ca | 89 | 0f | 1e | 3c | 78 | f0 |
| **5** | fd | e7 | d3 | bb | 6b | d6 | b1 | 7f | fe | e1 | df | a3 | 5b | b6 | 71 | e2 |
| **6** | d9 | af | 43 | 86 | 11 | 22 | 44 | 88 | d | 1a | 34 | 68 | d0 | bd | 67 | ce |
| **7** | 81 | 1f | 3e | 7c | f8 | ed | c7 | 93 | 3b | 76 | ec | c5 | 97 | 33 | 66 | cc |
| **8** | 85 | 17 | 2e | 5c | b8 | 6d | da | a9 | 4f | 9e | 21 | 42 | 84 | 15 | 2a | 54 |
| **9** | a8 | 4d | 9a | 29 | 52 | a4 | 55 | aa | 49 | 92 | 39 | 72 | e4 | d5 | b7 | 73 |
| **A** | e6 | d1 | bf | 63 | c6 | 91 | 3f | 7e | fc | e5 | d7 | b3 | 7b | f6 | f1 | ff |
| **B** | e3 | db | ab | 4b | 96 | 31 | 62 | c4 | 95 | 37 | 6e | dc | a5 | 57 | ae | 41 |
| **C** | 82 | 19 | 32 | 64 | c8 | 8d | 07 | 0e | 1c | 38 | 70 | e0 | Dd | a7 | 53 | a6 |
| **D** | 51 | a2 | 59 | b2 | 79 | f2 | f9 | ef | c3 | 9b | 2b | 56 | Ac | 45 | 8a | 09 |
| **E** | 12 | 24 | 48 | 90 | 3d | 7a | f4 | f5 | f7 | f3 | fb | eb | Cb | 8b | 0b | 16 |
| **F** | 2c | 58 | b0 | 7d | fa | e9 | cf | 83 | 1b | 36 | 6c | d8 | Ad | 47 | 8e | 01 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **log()** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **0** | ff | 00 | 01 | 19 | 02 | 32 | 1a | c6 | 03 | df | 33 | ee | 1b | 68 | c7 | 4b |
| **1** | 04 | 64 | e0 | 0e | 34 | 8d | ef | 81 | 1c | c1 | 69 | f8 | c8 | 08 | 4c | 71 |
| **2** | 05 | 8a | 65 | 2f | e1 | 24 | 0f | 21 | 35 | 93 | 8e | da | f0 | 12 | 82 | 45 |
| **3** | 1d | b5 | c2 | 7d | 6a | 27 | f9 | b9 | c9 | 9a | 09 | 78 | 4d | e4 | 72 | a6 |
| **4** | 06 | bf | 8b | 62 | 66 | dd | 30 | fd | e2 | 98 | 25 | b3 | 10 | 91 | 22 | 88 |
| **5** | 36 | d0 | 94 | ce | 8f | 96 | db | bd | f1 | d2 | 13 | 5c | 83 | 38 | 46 | 40 |
| **6** | 1e | 42 | b6 | a3 | c3 | 48 | 7e | 6e | 6b | 3a | 28 | 54 | Fa | 85 | ba | 3d |
| **7** | ca | 5e | 9b | 9f | 0a | 15 | 79 | 2b | 4e | d4 | e5 | ac | 73 | f3 | a7 | 57 |
| **8** | 07 | 70 | c0 | f7 | 8c | 80 | 63 | 0d | 67 | 4a | de | ed | 31 | c5 | fe | 18 |
| **9** | e3 | a5 | 99 | 77 | 26 | b8 | b4 | 7c | 11 | 44 | 92 | d9 | 23 | 20 | 89 | 2e |
| **A** | 37 | 3f | d1 | 5b | 95 | bc | cf | cd | 90 | 87 | 97 | b2 | Dc | fc | be | 61 |
| **B** | f2 | 56 | d3 | ab | 14 | 2a | 5d | 9e | 84 | 3c | 39 | 53 | 47 | 6d | 41 | a2 |
| **C** | 1f | 2d | 43 | d8 | b7 | 7b | a4 | 76 | c4 | 17 | 49 | ec | 7f | 0c | 6f | f6 |
| **D** | 6c | a1 | 3b | 52 | 29 | 9d | 55 | aa | fb | 60 | 86 | b1 | Bb | cc | 3e | 5a |
| **E** | cb | 59 | 5f | b0 | 9c | a9 | a0 | 51 | 0b | f5 | 16 | eb | 7a | 75 | 2c | d7 |
| **F** | 4f | ae | d5 | e9 | e6 | e7 | ad | e8 | 74 | d6 | f4 | ea | a8 | 50 | 58 | af |

Assuming we want to force a minimum of 4 (M) to be required of a possible (N) 8 fragments the following 4\*8 Vandermonde matrix is generated:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 0 | 0 | 0 |  |
|  | 1 | 1 | 1 | 1 |  |
|  | 1 | 2 | 4 | 8 |  |
|  | 1 | 4 | 10 | 40 |  |
|  | 1 | 8 | 40 | 3a |  |
|  | 1 | 10 | 1d | cd |  |
|  | 1 | 20 | 74 | 26 |  |
|  | 1 | 40 | cd | 2d |  |

Using Forward Error Correction (FEC) the following identity matrix is generated:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 0 | 0 | 0 |  |
|  | 0 | 1 | 0 | 0 |  |
|  | 0 | 0 | 1 | 0 |  |
|  | 0 | 0 | 0 | 1 |  |
|  | 77 | 40 | 38 | 0e |  |
|  | c7 | A7 | 0d | 6c |  |
|  | 53 | 2 | 6f | 3f |  |
|  | f1 | 7b | 83 | 08 |  |

For the example we use a small packet size of 8, in practice a larger block oriented fragment size would be used. The AES key and the encrypted text are then distributed into the following matrix:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 48 | 0e | 05 | 9f | c1 | 15 | cb | 66 |  |
|  | 39 | f7 | b4 | b9 | 33 | 9a | 87 | 8d |  |
|  | 81 | 68 | 4b | e8 | c3 | 2d | dd | 63 |  |
|  | 19 | f1 | 8a | d1 | 8c | 14 | 30 | 45 |  |

Multiplying the FEC identity matrix and the encrypted text matrix results in the following fragmentation matrix, where each row represents a fragment to be distributed. The original data can be reconstructed from any N rows.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 48 | 0e | 05 | 9f | c1 | 15 | cb | 66 |  |
|  | 39 | f7 | b4 | b9 | 33 | 9a | 87 | 8d |  |
|  | 81 | 68 | 4b | e8 | c3 | 2d | dd | 63 |  |
|  | 19 | f1 | 8a | d1 | 8c | 14 | 30 | 45 |  |
|  | 02 | 54 | bc | e3 | 06 | 52 | bd | 29 |  |
|  | 8a | eb | 5b | 2a | 3c | 06 | 0a | dd |  |
|  | 52 | 8c | 36 | c2 | bb | a9 | df | 91 |  |
|  | 34 | 3e | 07 | 95 | a6 | cc | 10 | 9f |  |

Note: Using the FEC identity matrix approach for fragmentation combined with the AES key being distributed in the binary data, does by definition mean all fragments could allow deriving 1/Mth of the AES key. This reduces the effective strength of the AES key used. Selecting larger values for M will reduce this impact.