Achieving Big Data Privacy via Hybrid Cloud

Xueli Huang and Xiaojiang Du
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA, 19122 USA
Email: {xueli.huang, dux}@temple.edu

***Abstract*—Nowadays the amount of data is being produced exponentially with the rapid development of electronic technology and communication, which makes it hard to cost-effectively store and manage these big data. Cloud computing, a new business model, is considered as one of most attractive solutions for big data, and provides the advantage of reduced cost through sharing of computing and storage resources. However, the growing concerns in term of the privacy of data stored in public cloud have slowed down the adoption of cloud computing for big data because sensitive information may be contained among the big data or the data owner themselves do not want any other people to scan their data. Since the data volume is huge and mobile devices are widely used, the traditional cryptographic approach are not suitable for big data. In this paper, we propose an efficient scheme for image data, which has much more volume than text data. We evaluate our scheme in real networks (including Amazon EC2), and our experimental results on image show that: (1) our scheme achieves privacy but only use 1/585.8∼1/398.6 the time of the AES algorithm; (2) the delay of our hybrid-cloud-based scheme is only 3%∼5% more than that of the traditional public-cloud-only approach.**

***Index Terms*—big data; cloud computing; hybrid cloud; data privacy; security; image**

I. Introduction

With the rapid development of electronic and communication technology, the amount of data produced by medical systems, surveillance systems or social networks has been grown exponentially, which makes it hard for many organizations to cost-effectively store and manage these big data. Cloud computing, a new business model, is considered as one of most attractive and cost-effective solutions for big data, and provides the advantage of reduced cost through sharing of computing and storage resources [1]. It utilizes an on demand provisioning mechanism and a pay-per-use model, and has drawn a lot of attentions in recent years.

However, the growing concerns in term of the privacy of data stored in public cloud have delayed the adoption of cloud computing for big data [2]. On one hand, a large amount of image, such as medical systems or social networks, may contain sensitive information. On the other hand, Cloud Service Providers (CSPs), who own the infrastructures on which clients' data are stored, have full control of the stored data. Therefore, the data stored in public cloud may be scanned by CSPs for advertisement or other purposes. Furthermore, attackers may be able to access data stored in cloud if there is not sufficient secure mechanism provided by CSPs.

Most existing solutions (e.g., [3]–[5]) employ traditional cryptographic algorithms, such as AES, to encrypt data and then store encrypted data in public cloud. However, for image data, which have much larger size than text data, heavy computation overhead will be introduced by this approach. Meanwhile, for the mobile devices, which have been widely used, much battery energy will be consumed, and it will increase delay because of the limited computation resources. Therefore, the traditional cryptographic approaches are not suitable for big data privacy.

In recent years, various image encryption algorithms have been proposed to speed up the process, among which the chaos-based approach with a substitution-diffusion layout appears to be a promising direction [6]–[9]. In the substitution stage, the positions of pixels of the image are shifted via some chaotic map, and then the pixel values of the shuffled-image are changed by chaotic sequences in the diffusion stage. However, the chaos system itself causes large computation overhead.

Another approach is to take advantage of hybrid cloud by separating sensitive data from non-sensitive data and storing them in trusted private cloud and un-trusted public cloud respectively [10]. However, if we adopt this approach directly, all images containing sensitive data or the ones that would not like to be seen by others have to be stored in private cloud, which would require a lot of storage in private cloud. Most users want to minimize the storage and computation in private cloud, and let public cloud do most of the storage and computation. To address the above challenge, we need to answer an important problem: How to efficiently achieve big data privacy by using hybrid cloud? Compared to using public cloud only, using hybrid cloud would have communication overhead between private and public cloud. Besides achieving data privacy, we want to reduce storage and computation in private cloud, as well as communication overhead between private and public cloud. In addition, the delay introduced by communications between private and public cloud should be small.

In this paper, we present a scheme that can efficiently achieve image data privacy in hybrid cloud. A novel random one-to-one mapping function is proposed for image encryption, which makes the pairwise affinity among jigsaws unreliable and at the same time significantly speeds up the process of substitution and diffusion. Only the random parameters of the mapping function are stored in private cloud.

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

II. Related Work

A. Data Privacy and Security in Cloud Computing

[3] utilizes and uniquely combines techniques of attributed-based encryption (ABE), proxy re-encryption and lazy re-encryption to provide fine-grained access control on encrypt files. In [3], each data file is associated with a set of attributes and each user is assigned an expressive access structure defined over the attributes of files.

In [4], J. Li et al. present a framework for data outsourcing and sharing in hybrid cloud. In the framework, the access control mechanism is provided by trusted private cloud to realize users' authorization and revocation. [5] presents an anonymous privilege control scheme to achieve anonymous cloud data access and fine grained privilege control by using multiple authorities in cloud computing system.

Even though the methods above could protect data privacy, data owner has to encrypt data via traditional cryptographic method before sending them to public cloud.

B. Image Encryption Methods

In [6], an image is cut into pieces, each pixel value is modified, pieces are permutated via the spatiotemporal chaos system, also called nearest-neighboring coupled-map lattices (NCML). A multi-scroll chaos system is designed in [7] to generate random sequences that XORs each electrocardiography (ECG) signal. Coupled map lattice (CML) is adopted in [8] as the chaos system to permute pieces and change the pixel values within each pieces. However, heavy computation is brought in by the chaos system because it has to iterate thousands of time first to get the smooth phase and much of float calculation is employed at the same, which makes this method hardly used for big data.

[10] proposes an efficient image encryption algorithm via hybrid cloud to secure image data stored in public cloud. In [10], the image is cut into pieces, and the pixels of each pieces are modified by a random noise before they are shuffled. However, since pixels within one piece is modified by a common noise, it is vulnerable to attacks based on gradient analysis.

III. Achieving Big Data Privacy via Hybrid Cloud

A. System and Threat Model

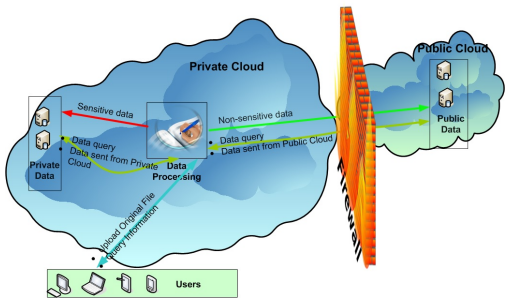

Fig. 1. The architecture of hybrid cloud

The architecture of a hybrid cloud is illustrated in Fig. 1. The original data come from private cloud, and are processed on servers within private cloud. If there are no sensitive data, the original data may be sent to public cloud directly. Otherwise, the original data will be processed to make no sensitive data leaked out. After being processed, most data are sent to public cloud, and a small amount of sensitive data are kept in private cloud. When a user queries the data, both private cloud and public cloud will be contacted to provide the complete query result.

We consider an un-trusted public cloud who are curious and may intend to browse users' data. The public cloud has full control of its hardware, software, and network.

B. Design Goals

We want to protect image data privacy stored in public cloud via hybrid cloud. Specifically, we want to remove sensitive data and store them in trusted private cloud, and store the processed data (without sensitive information) in un-trusted public cloud. It would require too much storage in private cloud if we simply store the entire image with sensitive information in private cloud. Therefore, our design goal is to achieve image data privacy via hybrid cloud and at the same time reduce the following overheads: (1) the amount of data stored in private cloud, (2) the communication overhead between private and public cloud, and (3) the delay introduced by communications between private and public cloud.

IV. Privacy of Image Data

In this Section, we discuss how to achieve image data privacy via hybrid cloud. The notations are given in Table I.

TABLE I
Notations

n	the maximum of valid value
m	the random parameter, with $0 < m < n$
g	greatest common divider of n and m
p, q	the original value
S	the valid data set, $S = \{0, 1, 2, ..., n-1\}$
p', q'	$p's$ and $q's$ mapped value via Equation 1
p'', q''	$p's$ and $q's$ mapped value via Equation 3
r	$r = floor(\frac{n}{m})$
$index_{piece}$	index of duplication group via Equation 3
num_{piece}	number of elements before the p'' group
$startpoint\_index$	the start point value of the p'' line
$line\_start[i]$	start point value of ith line with $0 \leq i < \frac{m}{g}$
$line\_num[i]$	number of points before ith line with $0 \leq i < \frac{m}{g}$
$point\_index$	location of p'' within the line

A. Dividing Image into Blocks

We divide a large image (size of $N \times N$) into n number of blocks, where each block has the same size $k \times k$. Take the Lena image for example, which has a size of 256×256, the size of each block ($k \times k$) is set to 32×32. The image is divided into $n = (256 \div 32) \times (256 \div 32) = 64$ pieces.

B. The Proposed Mapping Function

In this work, we propose a one-to-one mapping function, which maps an original pixel value to a different (unpredictable) value. The original pixel value of p is mapped into p' according to Equation 1.

$$p' = module(m \times p, n) + \lfloor \frac{p}{\frac{n}{g}} \rfloor \quad (1)$$

The proof that our mapping function is one-to-one is given in propositions 1 to 4 in Appendix.

The original image and the encrypted image obtained from our shuffling and mapping function are shown in Fig. 2 and Fig. 3 respectively.

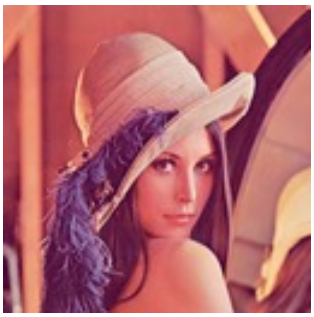

Fig. 2. Original image

Fig. 3. Encrypted image

C. Reverse of the One-to-One Mapping Function

Determine which pieces p' ***came from***:

Given p', m and n, we can get its greatest common divisor g. Since we add a different value for each pieces without bring in any duplication, p' satisfies $index_{piece} = module(p', g)$.

According to ***Proposition 2*** for Equation 3, we know that the module operation cut the hash function into g equal pieces, and the size of each size is $\frac{n}{g}$. Therefore, the number of elements before this group is $num_{piece} = \frac{n}{g} \times index_{piece}$.

After deducting the different value $index_{piece}$ from p', that is $p'' = p' - index_{piece}$, we can obtain its value by Equation 3.

Determine which line the point locates.

We have proved the ***Proposition 5*** in Section Appendix that: **If p is the start point of the** ith **line**, **q is the start point of the** $(i+1)th$ **line with** $0 \leq i < \frac{m}{g}$**, then** $q'' = module(p'' + (m - r), m)$.

Hence, the start point of the line where p'' locates is $module(p'', m)$.

Determine the location of the line.

It is obvious that $line\_start[0] = 0$, and as mensioned above, the start point of each line is $line\_start[i] = module(line\_start[i-1] + m - r, m)$ with $with\ i < \frac{m}{g}$. At the same time, the number of points starts with p is $k+1$ if $p'' < r$, otherwise the number of points is k.

For each line, the start index is the addition of its previous line's start index and the number of points of its previous line with $line\_index[0] = 0$, that is $line\_index[i] = line\_index[i-1] + line\_num[i-1]$.

Determine the location within the line.

The start point of the line with p'' is $startpoint\_index = module(p'', m)$, and we can get the index value of line contains p'' by searching $line\_start$ array with $line\_start[i] = startpoint\_index$. So the location within the line is $point\_index = floor(\frac{p''}{m})$.

Now, we can get the reversed value of p', that is $p = point\_index + line\_index[i] + num_{piece}$

D. Random Shuffle of Blocks

To make the modified image un-recognizable, we shuffle the blocks of the modified image by clustering the n blocks into a number of groups with randomly chosen strides and each cluster has the same size m. The details are given below:

1) For each cluster of blocks of the modified image.
   a) Randomly choose $stride$ value from $1, 2, ..., n$.
   b) Randomly choose $start$ value from $0, 1, ..., n-1$.
2) Increasing the start position, $start = start + stride$. If the current block has been chose before, move to the next block ($start = start + 1$), until it moves to a block that has not been chosen before. Then copy the block to the shuffled image.
3) Goto step 2 until m number of blocks have been chosen.
4) Goto step 1 until all of the blocks of the modified image have been chosen.

E. Recovering Images

When an image is queried, the request is sent to both the private cloud and public cloud at the same time. As we mentioned above, the information used to recover the image is obtained from the private cloud. First, we obtain the shuffle order $permut$ via Algorithm 1. Second, we re-order the blocks of shuffled image from the public cloud, which gives us the modified image. Third, we obtain the random values from the private cloud, and use them to recover the original image from the modified image.

V. Performance Evaluations

In this Section, we evaluate the performance of the image privacy scheme.

A. Security Analysis of Image Data

We divide an image into pieces, and, if we shuffle these pieces directly, we convert the problem to the "jigsaw puzzle problem", which has been proven to be NP-Complete if the pairwise affinity among jigsaw pieces is unreliable [11]. However, [12] finds that the dissimilarity-based compatibility, which is exploited to measure the color difference along the adjacent boundary, is more discriminative.

In our scheme, each pixel of every piece's color dimension is mapped into another value via our random one-to-one mapping function, disturbing the statistic features that could be used to get the anchor block of a puzzle. This is illustrated in Fig. 4 and Fig. 5.

We conduct experiments on several color-standard-test images of size 512×512, and each experiment is run 100

Algorithm 1 Obtaining Shuffle Order

Input: n: the number of blocks; $shuffle$: a string composed of $start$ value, $stride$ value and split signal $'$,$'$.
Output: Shuffle order - $perm$.
//get start, stride and cluster size information;
$shuff\_string\_array \leftarrow$ split $shuffle$ with $'$,$'$;
$start \leftarrow$ the start value of $shuff\_string\_array$;
$stride \leftarrow$ the stride value of $shuff\_string\_array$;
$cluster\_n \leftarrow$ the number of start;
$m \leftarrow \lceil \frac{n}{cluser\_n} \rceil$;$left \leftarrow n$;$cluster\_id \leftarrow 0$;$i1 \leftarrow 0$;
for $i = 0 \rightarrow n-1$ **do**
  $orig[i] \leftarrow i$;
while $left > 0$ **do**
  $i \leftarrow start[cluster\_id]$;
  if $m < left$ **then**
    $m \leftarrow left$;
  for $j = 0 \rightarrow m-1$ **do**
    while $orig[i]$ has been chosen **do**
      $i \leftarrow i+1$;
      if $i \geq n$ **then**
        $i \leftarrow i-n$;
    $permut[i1] \leftarrow orig[i]$;$i1 \leftarrow i1+1$;$left \leftarrow left-1$;
  $i \leftarrow i+stride[cluser\_id]$;$cluster\_id \leftarrow cluster\_id+1$;

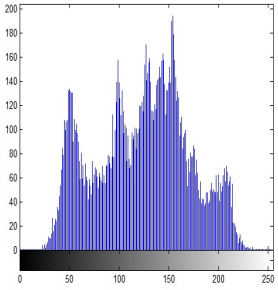

Fig. 4. Original image histogram

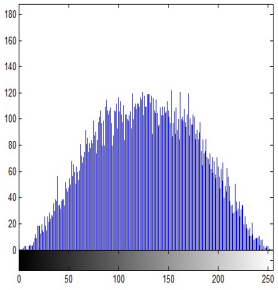

Fig. 5. Modified image histogram

times for each setting. The pairwise affinity is judged by the dissimilarity-based compatibility measurement of the sum of block color difference along adjacent boundaries. Take two blocks blk_i and blk_j for example, the Left-Right dissimilarity between them is calculated via Equation 2:

$$D(blk_i, blk_j) = \sum_{k=1}^{K} \sum_{l=1}^{d3} (blk_i(k,u,l) - (blk_j(k,v,l))^2 \quad (2)$$

where $d3$ is the number of image color dimensions and each block is a $K \times K \times d3$ matrix, u indexes the last column of blk_i, v indexes the first column of blk_j. The number of blocks n is $\lceil \frac{N \times N}{K \times K} \rceil$. The color difference square D is assumed conform to an exponential distribution.

Fig. 6 and Fig. 7 present the accuracy probability of identifying adjacent block correctly from original image and modified image, respectively. Fig. 6 and Fig. 7 show that the features of adjacent block edges are removed at the same time,

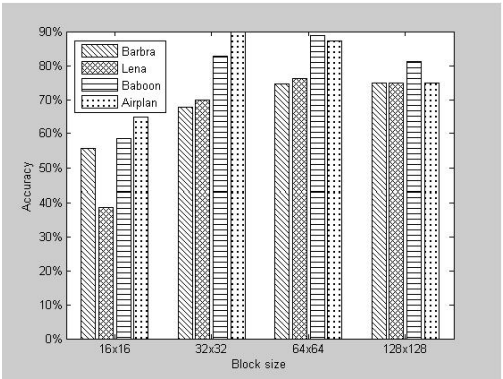

Fig. 6. Accuracy without image modification

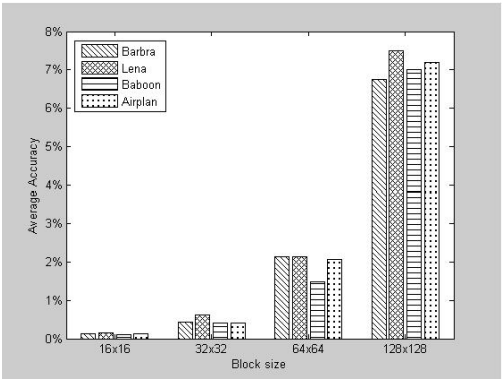

Fig. 7. Accuracy after image modification

making the jigsaw puzzle problem NP-Complete, which means that it cannot be resolved in polynomial time.

B. Evaluation

1) Efficiency of Image Encryption: To evaluate the efficiency of our privacy preserving method, we compare our algorithm with AES algorithm. We use four different sizes of color Lena image, namely 128×128, 256×256, 512×512 and 1024×1024. Both our algorithm and AES (128-bit key) are run on the Matlab platform in the same computer. For each size of image, we run 100 times modification and recovery operations on the image and get the average time. We run AES once to get the time of AES encryption and decryption. The results are given in Table II, where the time unit is second.

TABLE II
RUNNING TIME OF OUR ALGORITHM AND AES

Block Size	Our algorithm	AES	Time ratio
128×128	0.3076	122.6	398.6
256×256	0.9394	490.1	521.7
512×512	3.4415	1957	568.6
1024×1024	13.356	7824	585.8

From Table II, we can see that the time for processing image increases when the image size becomes larger. For all the block sizes, our algorithm is $398.6 \sim 585.8$ times faster than

the AES algorithm. The reason is that our algorithm has no iteration, while AES consists of four stages with many rounds. To sum up, our algorithm provides image data privacy and it is much more efficient than AES.

C. Experiments using Amazon EC2

Our private cloud is set up in a server located in Computer and Information Science department of Temple University, and public cloud is built on Amazon EC2 Cloud. The Microsoft SQL server 2005 is installed in the local server, which is used to store private and sensitive data. Amazon Relational Database Service (Amazon RDS) SQL server 2008 is installed in Amazon EC2 and is used to store non-sensitive data. The Microsoft Visual Studio 2010 software is utilized to create web-sites that provide services through webpages, which are developed using ASP.NET and C♯. Internet Information Services (IIS) is chosen as the web server, which supports both Data Processing Applications and Data Accessing Applications. Four different sizes of Lena image are chosen to evaluate the security, efficiency and overhead of our scheme. We record the delay between the time (t1) when a user sends the request and the time (t2) when the web server has the data ready, and the delay when our scheme is not used. The average of the 100 runs are reported in Table III, where the time unit is millisecond.

TABLE III
COMPARISON OF DELAY

Block Size	Using our scheme	Without our scheme	increase
128×128	23.3123	22.5023	3.60%
256×256	76.1976	73.4673	3.72%
512×512	242.4657	230.6346	5.13%
1024×1024	976.4206	927.3578	5.29%

Table III shows that the delay increases as the image size becomes larger, which is easy to understand. Table III also exhibits that our scheme only increases the delay a little bit, between $3.60\% - 5.29\%$. This demonstrates the efficiency of our scheme. Meantime, if we compare the data of Table II with that of Table III, we discover that the execution time of our scheme implemented via C♯ (Table III) is much less than that implemented in Matlab (Table II, where the time unit is second). The reason is that we adopt the LockBitmap class that converts bitmaps to byte-arrays in the C♯ implementation, which greatly accelerates the image processing.

TABLE IV
OVERHEAD ON PRIVATE CLOUD

Cluster size	Number of cluster	Overhead (byte)
1	64	571
2	32	389
4	16	298
8	8	245

Communications between the private cloud and the public cloud cause some communications overhead.We also run experiments to measure the communication overhead introduced by our scheme. Table IV shows that the communication overhead on private cloud becomes larger with the increase of cluster size when image is divided into 64 blocks. The reason is that the number of random $start$ value and $stride$ value increases even though the image size is same. Table IV shows that our scheme introduces little overhead.

VI. CONCLUSION

To promote the cloud computing as a solution for big data, we proposed an efficient scheme to address the increasing concern of data privacy in cloud for image data. Our scheme divides an image into blocks and shuffles the blocks with random start position and random stride. Our scheme operates at the block level instead of the pixel level, which greatly speeds up the computation. We converted the image privacy problem into the jigsaw puzzle problem. To make the jigsaw puzzle problem NP-complete, we modified each pixel of the image data by using our random one-to-one mapping function. These operations make the pairwise affinity un-reliable and make the shuffled image un-recognizable. We implemented our scheme in real networks (including the Amazon EC2) and tested the security and efficiency. Both our analysis and experimental results showed that our scheme is secure, efficient and has very small overhead.

ACKNOWLEDGMENT

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-0963578, CNS-1022552, and CNS-1065444.

REFERENCES

[1] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 2013.

[2] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, 2012.

[3] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, 2010.

[4] J. Li, C. Jia, J. Li, and Z. Liu, "Novel framework for outsourcing and sharing searchable encrypted data on hybrid cloud," in *Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on*. Springer, 2012.

[5] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *IEEE INFOCOM*, 2013.

[6] Y. Wang, K.-W. Wong, X. Liao, and G. Chen, "A new chaos-based fast image encryption algorithm," *Applied soft computing*, 2011.

[7] F. Sufi, F. Han, I. Khalil, and J. Hu, "A chaos-based encryption technique to protect ecg packets for time critical telecardiology applications," *Security and Communication Networks*, 2011.

[8] X. Wang and L. Teng, "An image blocks encryption algorithm based on spatiotemporal chaos," *Nonlinear Dynamics*, 2012.

[9] G. Zhang and Q. Liu, "A novel image encryption method based on total shuffling scheme," *Optics Communications*, 2011.

[10] X. Huang and X. Du, "Ensuring data privacy by hybrid cloud," in *IEEE ICC*, 2013.

[11] E. Demaine and M. Demaine, "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity," *Graphs and Combinatorics*, vol. 23, 2007.

[12] T. Cho, S. Avidan, and W. Freeman, "A probabilistic image jigsaw puzzle solver," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010.

APPENDIX

In the following, we will prove that the above mapping function is one-to-one.

$$p'' = module(m \times p, n) \tag{3}$$

Proposition 1: **For Equation 3, if g is equal to 1, and $\forall\ p \in S,\ q \in S,\ and\ p \neq q$, then $p'' \neq q''$.**

Proof: Assume $p > q$, vice versa, and $m_p = floor((m \times p)/n)$ and $m_q = floor((m \times q)/n)$. The difference of p'' and q'' is:$p''-q'' = m\times(p-q)-n\times(m_p-m_q)$. Assume $p'' = q''$, we will get $m \times (p-q) - n \times (m_p - m_q) = 0$. Since $p \neq q$ and $m > 0$, $m \times (p-q) > 0$ and $n \times (m_p - m_q) \geq 0$.

Because p, q, m_p and m_q are all integers and the greatest common divisor of m and n is equal to 1, to make $p''-q'' = 0$, $p - q = k \times n$ and $m_p - m_q = k \times m$, where k is an efficient to make $m \times (p-q) - n \times (m_p - m_q) = 0$ and $k \geq 1$.

However, $(n-1) \geq p > q \geq 0$, which makes $p - q < (n-1)$. There is a contradiction between $p-q < (n-1)$ and $p - q = k \times n\ with\ g \geq k \geq 1$, hence the assumption is not correct. ■

Proposition 2: **For Equation** 3, **if** $g \geq 2$, **and** $\forall\ p \in S,\ q \in S,$ **and** $\mid p - q \mid = k \times \frac{n}{g}$ **with** $k \geq 1$, **then** $p'' = q''$.

Proof: Assume $p > q$, the vice versa, then $p'' = module(q \times m + k \times m \times \frac{n}{g}, n)$ Since g is the greatest common divider of m and n, $\frac{m}{g}$ is integer and no less than 1. Therefore $p'' = module(q \times m + k \times \frac{m}{g} \times m, n) = module(q \times m, n) = q''$.■

According to ***Proposition 1*** and ***Proposition 2***, for $\forall\ p, q \in [0, 1, 2, ..., \frac{n}{g} - 1],\ if\ p \neq q\ ,$ then $p'' \neq q''$.

Proposition 3:**for** $\forall\ p, q \in\ [0, 1, 2, ..., \frac{\mathbf{n}}{\mathbf{g}} - 1]$ **and** $p \neq q$, **the minimum distance between** p'' **and** q'' **is** **g**, **that is**, **minimum**$(\mid\ p'' - q''\ \mid) =$ **g**.

Proof: As we know, if there is no module operation, the graph of $p'' = m \times p$ is a straight line, and distance between adjacent points is same. However, after the module operation is joined, the lines is cut into $\frac{m}{g}$ parallel lines.

Assume q is the start point of one of line, and $q + k$ is the last point of line with $k \geq 0$. Then $q'' = m \times q - m_q \times n$, and $(q+k)'' = m \times (q+k) - m_q \times n$, with $0 \leq q'' < m$ and $n < (q+k)'' < n-m$. It is clear that $q+k+1$ is the start point of another line, and $(q+k+1)'' = m\times(q+k+1)-m_p\times n-n$ with $n < (p+1) \times m - m_p \times n < n + m$.

Assume $n = k1 \times g$ and $m = k2 \times g$, where the greatest common divider of $k1$ and $k2$ is 1. Therefore, the distance between the two lines with same index is: $\mid (q+k+1)''-q'' \mid = \mid (k+1) \times m - n) \mid = g\times \mid (k+1) \times k2 - k1 \mid$.

As discussed above, the greatest common divider of $k1$ and $k2$ is 1, and $k \geq 0$ and $k2 < k1$. It is clear that $(k+1)k2 \neq k1$, otherwise g is not the greatest common divider of m and n.

Therefore $\mid\ (k+1) \times k2 - k1\ \mid \geq 1$, and then distance between each line with same index is multiple of g.

We assume the first line is the one whose initial variable is $q = 0$, and then we get $q'' = 0$. Since the distance between each line with same index is multiple of g, the other $\frac{m}{g} - 1$ lines' first mapped value should be $k_i \times g$, where $i = \{1, 2, ..., \frac{m}{g} - 1\}$. Then we sort these $\frac{m}{g}$ mapped value, and k'_i is the coefficient of the sorted ith line's distance from the first line:

$$0 < k'_1 \times g < k'_2 \times g < k'_3 \times g < \ ...\ < k'_{\frac{m}{g}-1} \times g < m$$
$$where\ \frac{m}{g} > k'_i \geq 1\ and\ k'_i\ integer.$$
$$\Leftrightarrow 0 < k'_1 < k'_2 < k'_3 < \ ...\ < k'_{\frac{m}{g}-1} < \frac{m}{g}$$

Since there are $\frac{m}{g}$ lines and the initial mapped value is 0 and the maximum mapped value should not beyond $\frac{m}{n}$, and each mapped value is integer.

$$0 < k'_1 < k'_2 < k'_3 < \ ...\ < k'_{\frac{m}{g}-1} < \frac{m}{g}$$
$$\Rightarrow k'_i = i$$

Therefore, the distance with same adjacent index points is $\mid k'_i - k'_{i+1} \mid = g$. Since $m > g$, the minimum distance between any different point is g .■

Proposition 4: **For Equation 1,** $\forall\ p \in S,\ q \in S,$ **and** $p \neq q$, **then** $p' \neq q'$.

Proof: If the g=1, $\frac{n}{g} = n$. Since $q \in \{0, 1, 2, \ ...\ , n-1\}$, $0 \leq \frac{q}{\frac{n}{g}} < 1$. Therefore $\lfloor \frac{p}{\frac{n}{g}} \rfloor = 0$, and the Equation 1 becomes Equation 3. According to ***Proposition 1***, $\forall\ p \in S,\ q \in S,\ and\ p \neq q$, then $p' \neq q'$.

When $g > 1$, without the second part of Equation 1, according to ***Proposition 2*** we know that the module operation cut the hash function into g pieces, and hashed value with same index in each pieces are equal to each other, and at the same time according to ***Proposition 3***, the minimum distance between each lines is g. For g repeat part mentioned in Equation 1, we add a different value according to $\lfloor \frac{p}{\frac{n}{g}} \rfloor$, which means adding increasing value from 0 to $g-1$. Since the minimum distance between any pair of points within any repeat part is g, the adding operation will not introduce any repeat, and since each of part add a different value, all of the points are not equal to each other. With the module operation, none mapped value is beyond the set S. ■

Proposition 5:**If** p **is the start point of** ith **the line**, q **is the start point of** $(i+1)th$ **line with** $0 \leq i < \frac{m}{g}$**, then** $q'' = module(p'' + (m - r), m)$.

Proof: Assume $n = k \times m + r$ with $k > 0$. Since p is the start point of ith line, $n - r \leq p'' + k \times m < n + m - r$.

If $p'' < r$, which makes $n - r \leq p'' + km < n$, then $p + k$ is the last point of the line, and q'' should be: $q'' = p'' + k \times m + m - n = p'' + m - r$. Then number of point in the line which starts with p is $k + 1$.

If $p'' \geq r$, which makes $n \leq p'' + km < n + m - r$, then $p + k - 1$ is the last point of the line, and q'' should be $p + k$, that is $q'' = p'' + k \times m - n = p'' + n - r - n = p'' - r$ Then number of point in the line which starts with p is k.

Therefore, $q'' = module(p'' + m - r, m)$.■