Design and implementation of Negative Authentication System

Noname manuscript No.
(will be inserted by the editor)

Design and Implementation of Negative Authentication System

Dipankar Dasgupta · Abhijit Kumar Nag · Denise Ferebee · Sanjib Kumar Saha · Kul Prasad Subedi · Arunava Roy · Alvaro Madero · Abel Sanchez · John R. Williams

the date of receipt and acceptance should be inserted later

Abstract Modern society is mostly dependent on online activities like official or social communications, fund transfers and so on. Unauthorized system access is one of the utmost concerns than ever before in cyber systems. For any cyber system, robust authentication is an absolute necessity for ensuring security and reliable access to all type of transactions. However, more than 80% of the current authentication systems are password based, and surprisingly, they are prone to direct and indirect cracking via guessing or side channel attacks. The inspiration of Negative Authentication System (NAS) is based on the negative selection algorithm. In NAS, the password based authentication data for valid users is termed as password profile or self-region (positive profile); any element other than the self-region is defined as non-self-region in the same representative space. The anti-password detectors are generated which covers most of the non-self-region. There are also some uncovered regions left in the non-self-region for inducing uncertainty to the attackers. In this work, we describe the design and implementation of three approaches of NAS and its efficacy over the other authentication methods. These three approaches represent three different ways to achieve obfuscation of password points with non-password space. The experiments are conducted with both real and simulated password profiles to justify the efficiency of different implementations of NAS.

Keywords Cyber-security · levels of abstraction · security event · passwords · authentication · negative authentication · hashing · salting

D. Dasgupta
The University of Memphis E-mail: dasgupta@memphis.edu

A. K. Nag (Corresponding Author)
Texas A&M-Central Texas E-mail: aknag@tamuct.edu

D. Ferebee
The University of Memphis E-mail: dferebee@memphis.edu

S. K. Saha
The University of Memphis E-mail: sksaha@memphis.edu

K. P. Subedi
The University of Memphis E-mail: kpsubedi@memphis.edu

A. Roy
The University of Memphis E-mail: royism.arunava@gmail.com

A. Madero
Massachusetts Institute of Technology E-mail: amaderog@mit.edu

A. Sanchez
Massachusetts Institute of Technology E-mail: doval@mit.edu

J. Williams
Massachusetts Institute of Technology E-mail: jrw@mit.edu

1 Introduction

Online activities, sensitive data transfers, official communication, activities containing delicate personal information are increasing every day. Authenticity is the highest significant issue than ever before because of inherent insecurity in online communication and transactions. For any computing systems, servers or online storage, proper authentication is an absolute necessity for ensuring legitimate access, privacy, security, and reliability.

Although password-based authentication systems are the oldest and most popular among all other methods of authentication, it is very vulnerable to a wide variety of attacks. The majority of the password-based authentication systems store the positive authentication data (password profile) in the database server, and every access request to that server is authenticated by comparing that request with the existing password data [33].

One of the basic problems of positive authentication is the risk of malicious access to the password profile. Any person who has stolen files of hashed passwords can often use brute-force methods to find out a password p whose hash value $H(p)$ is equal to the hash value stored for a given user's password, thus allowing him/her to impersonate the legitimate user. Various password cracking methods have been reported to be successful in cracking hashed passwords. Also, a side-channel attack of stealing password profiles is a severe threat to secure authentication and access control. A recent report from SANS Institute [27] illustrates the statistics of cracking different hashed passwords using different approaches. Password cracking was also instrumental, in a recent cyber espionage campaign against the New York Times [30]. A 2008 study [19] of online black markets found a vibrant economy trading in stolen passwords. Due to the widespread re-use of passwords across multiple websites [4], an emerging attack model is to compromise accounts by a guessing attack against a low-security website and attempt to re-use the credentials at critical internet sites [2]. In addition to that, other possible scenarios can trigger the attacks on passwords. A large number of naive users choose passwords very poorly. There are many cases that some anonymous person successfully impersonates at least some users of a system by attempting logins with common passwords [3]. This situation can be avoided by requiring users to use uncommon passwords [31].

This paper focuses on designing and implementing a negative authentication system that can easily be integrated with the current password based authentication system by providing an additional layer before the positive password database. The details of different versions of NAS and their implementations are illustrated in this paper. The experiments are conducted with real-world password data, and the results are described to justify the three different designs of NAS. The paper is separated into seven parts. Section 2 covers the overall concept of Negative Authentication System. Section 3 highlights the three different approaches of NAS at the high level. Section 4 demonstrates the detail insights of these three approaches. Section 5 demonstrates the architectural design to implement the NAS algorithms. Section 6 covers the experimental design and discussion of the results. Section 7 shows some significant difference of the NAS approach with Positive Authentication. Section 8 illustrates the reliability and validity of NAS approaches and comparison of other contemporary password based approaches. Section 9 provides concluding remarks regarding NAS approach.

2 Negative Authentication System (NAS)

The idea of Negative Authentication System (NAS) is based on the negative selection algorithm [18, 13, 22, 11]. This idea of the negative selection algorithm is inspired from the T-cell maturation procedure in the immune system. A T-cell is eliminated before deploying functionality if a T-cell in thymus recognizes any self-cell. The fundamental principle of the Negative Selection Algorithm is as follows [10]:

1. Define the set S ($Self$) as the regular pattern of activity of a system which needs to be monitored.
2. Generate a set D ($Detector$); those must not match any element from S.
3. The set S is continually monitored for changes. The detector set D is adjusted if any change occurs in S.

Likewise, the negative selection algorithm generates a set of 'detectors' where no detector has any similarity with the 'self 'set. Here, the 'self'set consists of the authentication data denoted as 'password points'and hence 'detector'set consists of other elements excluding the proper authentication data. Now, the generated detectors are used to detect any other element than the authentication data. These detectors consequently recognize malicious access request by using the same matching rule used for authentication purpose. So, this algorithm requires only available training data, and after that, it can be used as an anomaly detection algorithm [10]. Valid credentials or 'self'set are termed as 'positive information', and 'detector'set are the credentials that are non-valid are termed as 'negative information'. Briefly, user identities are confirmed not by using valid or positive information rather by using of negative information.

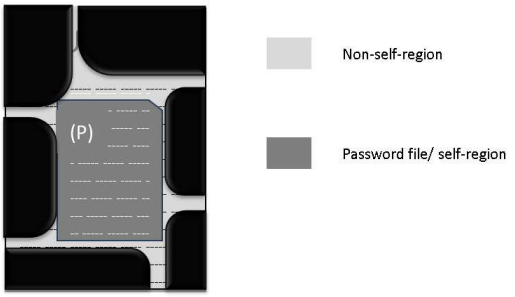

Fig. 1: **The password space (Self region) and anti-password space (Non-self region) shown in the total username-password space**

In NAS, 'self'set or data that contains the password based authentication data for valid users is termed as either password file, self-space or self-region. All other elements that do not belong in the self-region creates the non-self-space or non-self-region. Some elements are created by using NAS to cover those non-self-regions. These elements are either called detectors, anti-passwords or anti-password region. Some areas in the non-self-region are left uncovered to induce ambiguity to the attackers. It is desired to generate detectors to cover most of the non-self-region and thereby increase the detection rate of invalid access requests (Fig. 1).

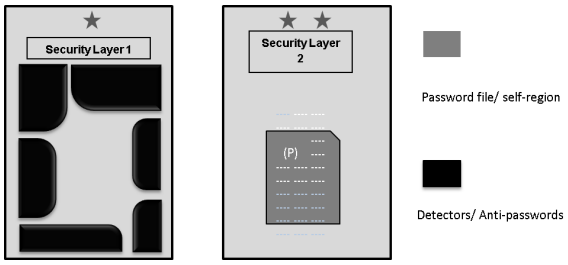

Fig. 2: **Different security layers of NAS separating the detector region and self-region**

NAS uses two servers for the validation of the access request. The self-region and anti-password region are kept in separate servers for enhancement of security of the authentication system (Fig. 2). The negative detectors that check for guessing attacks are stored on the first server that handles all incoming requests. The first server contains only the detectors and no other information (Fig. 2). Every access request arrives at the first server and is checked with the detectors. If any access request information is matched with any detector, then that access request is marked as an invalid request. Otherwise, that request is sent to the next server that contains the password based authentication data. This server checks the request, and if it finds a match, then it marks it as a valid request (Fig. 3). In this way, there is no direct communication between the requests and the positive authentication server (second server). But, the communication between the first server and the second server is transparent to the users. So, the user experience remains the same as the previous method of positive authentication.

This profile represents the negative abstraction of valid credentials and is the problem specific realization of the negative database described by [14]. This negative authentication for a system login application has been tested successfully workable [8]. This negative approach has some benefits over the traditional positive authentication approach. The negative authentication module is supposed to detect and filter out most of the invalid requests, therefore, such guessing requests have a low probability of accessing the positive authentication module. Furthermore, negative detectors are placed in the front security perimeter. So, the negative detectors become vulnerable to malicious access and hence more vulnerable to offline guessing attacks. But,getting a copy of the negative detectors does not reveal any information to the positive password file due to the ambiguity introduced intentionally in their (detectors) generation. Therefore, exposing the negative detector upfront reduces the overall password cracking risk. Hence NAS can significantly benefit from side channel attacks of passwords. Also, the password file that contains the authentication data resides in the second server and is never exposed to the outside of NAS which ensures that the positive profile is safer than being compromised.

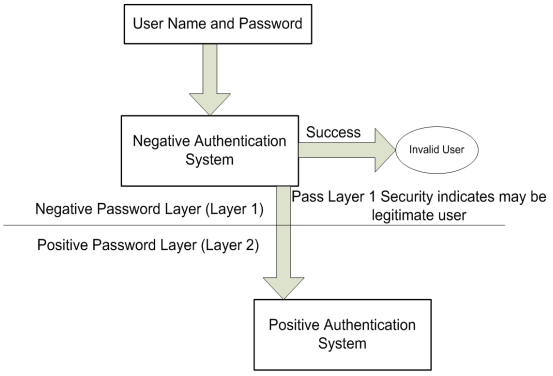

Fig. 3: **Logical view of negative (anti-password space) and positive (password space) authentication NAS layers**

3 Different Approaches of NAS

There are different approaches of the Negative Authentication System (NAS)based on representation of password space. The main focus of these approaches is to generate negative detectors to cover most of the non-self-region and increase the detection rate of invalid access requests. Each approach maps the information in different representation space. In this current research, we consider three representational space and develop three approaches for NAS: Real-valued, Binary-valued, and Grid-based. In this paper, the first two approaches are considered non-deterministic in terms of detector

creation while the third approach is a deterministic approach to create the set of detectors. Various approaches create various sized detectors and provide a different layer of obfuscation against compromise of password data.The details of each of these approaches are discussed in the following subsections.

3.1 Real-valued NAS (R-NAS)

The real valued space model is a hypersphere based non-deterministic model of NAS. This model uses n dimensional real space model to represent self-region and detectors. Both detectors and self-regions are some hypercube of $n \in \mathbb{Z}^+$ dimensions [23, 24]. Non-self-region, not covering any detectors is also of n dimensions. Here, dimension means the encrypted and segmented login information which is clearly shown in Fig. 4a. From Fig. 4a, it is clear that the initial login information are the username and the password. Then using the technique described in Fig. 8, the hashed (encrypted) login information can be generated. The encrypted login information are then divided into n segmented, which can be considered as the dimension. For example, if the length of the encrypted login information is 128 bits and it is segmented into four 32 bit segments, then the dimension of the space is 4. The normalized value of each segment in the real valued space model is spread over [0,1]. To create the set of detectors, points are chosen from the non-self-region that is not covered by any existing detectors. The radius of the detector is chosen in such a way so that the hypersphere will not overlap with the self-region. The Overlap may be allowed among the detectors. Fig. 4b also shows a 2-dimensional projection of a real space model implementation with self-points (self-region) and detectors. However, the remaining portion of Fig. 4b demonstrates the 2-dimensional non-self-region. Here, the real valued space implementation uses Euclidean distance for calculation of the distance between any two points in the space.

3.2 Binary-valued NAS (B-NAS)

The binary valued space model is hypercube based non-deterministic model of NAS. This model is inspired from the original V-Detector algorithm [24]. Binary valued space model is a discrete space model and uses only two values for each dimension; either 0 or 1. This model uses n-dimensional binary space, where n is the length of the representation string (encrypted user information). Therefore, different username-password information will not be mapped to a same point in the represented binary space. A Total number of elements in the binary valued space model is 2^n, where n is the dimension of the space. A sample illustration of binary space is shown in (Fig. 5), where n=3. To create the detectors, points are chosen from the non-self-region of the binary space that is not covered by any existing set of detectors. The radius of a detector is chosen in such a way that the hypercube representing that detector will not overlap with the existing self-region. These generated detectors cover some portion of the non-self-region. Overlaps among the detectors may be allowed. This model uses Hamming distance as a distance measure between any two points in the space. The Hamming distance between two entities of equal length is the number of positions at which the corresponding bits are different. An example of Hamming distance calculation is shown in (Fig. 6).In Fig. 6, the two eight binary digit numbers have four positions where the bits differ to each other. Hence, the hamming distance between these two numbers are 4.The differed positions are shown in bold in the figure.

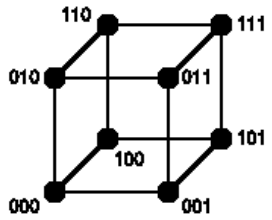

Fig. 5: **Binary Space in three dimensional space.**

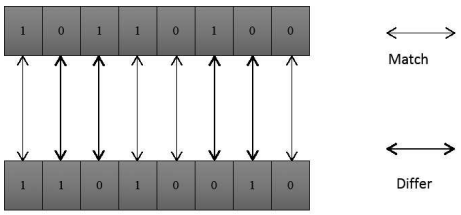

Fig. 6: **Hamming Distance calculation as a distance measure.**

3.3 Grid-based NAS (G-NAS)

Grid based implementation is a deterministic model of NAS [9]. The idea of Grid based approach for NAS is

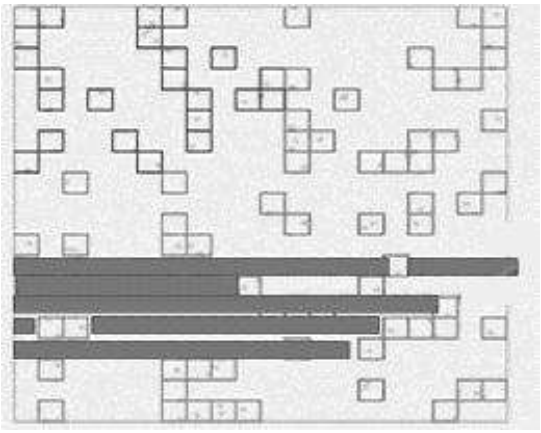

Fig. 7: **Sample display of self-region and detector region in Grid Space. Here gray boxes are detectors and white boxes are self-points.**

inspired from work by Williams et *al.* 2004. This Grid-based NAS model is designed in a way so that, there is no overlap among the detectors. It represents the password space in a square grid ($N \times N$ matrix, where $N \in \mathbb{Z}^+$). The Grid is stored as a 2-dimensional integer array with a specified dimension. We can choose the optimum grid size to divide the space according to the obfuscation level to be achieved. For example, the size of the grid can be considered as 512×512 or 256×256 etc. Self-points that are mapped to particular cells are identified by row number and column number), increases the value of that position by 1. Hence, any cell as value $M >= 1$, means M self-points are mapped to that particular grid. The cells for non-self-points are easily identified with the value that they contain (here the value is 0).

We store the (x, y) coordinates of the cell for each username-password for the positive space, for N username-passwords we store N coordinates and the grid size is stored once for the entire space. The complexity of the algorithm to generate the negative detectors is $O(N)$ or less. We start at a cell occupied by a positive cell (where $M > 0$) and define a run connecting series of cells (say along the x-axis) until we hit another positive cell. We encode the run-length of cells using only the beginning and end cell ids. Thus, the storage we require to define the complete negative space will be no greater than $2 \times N +$ some small constant brought in by runs that end on boundaries of the space. If we combine x and y co-ordinate runs, we will get even larger compression. Thus, even though the negative space is much bigger than the positive space we can define it for our purposes in terms of the N cells of the positive space.

Detectors are created by scanning the Grid once and keeping track of the cells that have their corresponding M equal to zero. Consecutive cells of corresponding "M equal to zero" in a row are considered as one detector and its run-length is calculated with the number of consecutive cells. In the Grid based implementation, it is possible to increase or decrease the detectors' coverage by changing the grid dimension.

Detectors generated from this implementation are stored with their run-length and the starting index. In our implementation, the detectors are saved in row-wise format and each row, the detector is stored with the starting column index and run length. The benefit of using row-wise format is, to find a particular point in the space, the row of that point can be determined easily from the hashing of the password and the search space is reduced to that specific row to check whether it falls in a detector cell or self-point grid. This approach also compresses the space required to store the total detector region, and the region can be reconstructed easily from the saved information of the detector space. A sample grid based representation is shown in Fig. 7. Fig. 7 shows the self-region and detector region in Grid space. The gray boxes are detectors (negative space) and the white boxes are self-points (positive space) in the sample representation of the grid space. The details of the grid space and generation of positive grids and detector grids are discussed in later section.

4 Implementation of Negative Authentication System (NAS)

All of the approaches discussed in the previous sections were implemented and tested in the lab environment. The self-regions in these implementations consist of username-password points termed as self-points. The Implementation details of the approaches are discussed in the following sections.

Hashed string generation

The username and password pairs are converted to corresponding self-point of the representing space model. Password is hashed with the specified hashing algorithm. Unique (dynamic) salts are added with hashed passwords and hashed again. The Corresponding username is added with last hashed output and hashed for the last time after adding a fixed (static) salt. The process is shown in Fig. 8. Self-points are created from these hashed string output using the implementation specific approaches.

Confusion parameter

The derived self-points are usually a single point in the model space. To induce some level of obfuscation, each self-point is surrounded by some of its neighboring regions. For real and binary valued space models, every self-point forms a hypersphere having a radius specified in implementations and center as the self-point itself. The higher confusion parameter shrinks the non-self-region but the total number of detectors does not decrease significantly. The number of detectors depends on the interleaving gaps between hyperspheres, rather than by the total area of non-self-region, as long as this total area does not become too small. As the randomness of hashed self-points are uniformly distributed [35] within the required n-dimensional space (because of the one-way hash function property), the confusion parameter appears to have little effect on the detector set size [12]. For grid space model, every self-point is mapped to a grid cell and the whole cell is defined as self-cell or self-grid, the size of the self-grids (depends on the grid size) is considered as the confusion parameter for the grid model.

Detector Coverage

Detector coverage can be defined as the volume of the non-self-space covered by the detectors. In the present work, the detector coverage has been normalized between 0 and 1. In this work, the detector coverage has been calculated using the Monte-Carlo simulation technique [28].

4.1 R-NAS Implementation

In real space implementation of NAS, a 4-dimensional real space model is implemented to map the total space. Though the real space model is a continuous space model, in the implementation, only predefined places after the decimal point are considered (typically 3 to 5 places). Self-region is defined by self-points that are hashed output of username and password. Every hashed string is divided into four segments to be mapped into the 4-dimensional real space. Coordinates of the self-point are calculated by normalizing each part of the hashed strings from 0 to 1 (Fig. 9). Each self-point forms a hypersphere with a small radius. This radius is defined as confusion parameter.

The non-self-region is created by the implemented R-NAS algorithm. This algorithm generates the detectors to cover the non-self-region. Each detector consists of a center, and a radius and so defines a hypersphere of 4-dimension. Both self-points and detectors form hyperspheres. The radius of self-points is defined as confusion parameter. The center and radius of detectors are generated in such a way so that detectors can overlap with each other, but they cannot overlap with the region of any self-points. The pseudo-code for the real space implementation algorithm is given in appendix (Fig. 26).The input and output of the detector generation algorithm are mentioned in the pseudo-code. Code block in line 8-12 checks whether candidate detector, 'x' falls in the existing list of detectors. If that falls, the failed count will be increased to keep track of total failed attempts. Code block in line 13-18 computes the minimum distance among all the self-points and the candidate detector, x. Code block in line 20-28 checks the overlap with the candidate detector, x with other existing detectors and decides whether it can be included in the existing detector set. Code block in line 29-32 adds the candidate detector, x to the detector set and checks the termination criteria to exit or not. The final set of detectors is then returned.

4.1.1 Confusion Parameter for R-NAS

This term defines the radius of the self-spheres created using the self-points in the real space. Accordingly, the value of the confusion parameter controls the volume of self-point. Usually, a small value is chosen for the confusion parameter so that the self-region remains negligible compared to the total space [12]. In the real space implementation, confusion parameter is set as the minimum value possible (e.g. if four places after a decimal point is considered, then confusion parameter is set as 0.0001).

4.1.2 Expected Coverage of Non-self space

Expected coverage is the term that controls the portion of non-self-region covered by the detectors and it can be defined as the following ratio:

$$\frac{\text{Sum of detectors' regions coverage excluding overlap}}{\text{1 - sum of all self regions' coverage}}$$

In NAS, some region from the non-self-region is not covered by the detectors to induce some obfuscation for the attackers. Generally 98% coverage is targeted while creating the detectors. As the detectors are created with allowing overlap, the actual coverage is smaller than 98%.

4.2 B-NAS Implementation

In binary valued space implementation of NAS, no two points will map to a single point in binary space. Based

on the password points (self-points), which are hashed output of username and password pair, the implemented algorithm calculates the non-self-detectors. Each detector consists of a center $a(\in$ non-self-region) and radius r, where all the points from a with r hamming distance are the members of that detector. These generated detectors cover some portion of the non-self-space (negative space). In this implementation, it is assumed that both the self-points and detectors are hyper-cube. Detectors can overlap with each other but they cannot overlap with the volume of self-points. Because of this radius, a self-point or a detector is not a single point in the space; in fact, it is composed of many points in the space. A sample scenario of self-point and detectors in reference to the NAS are given below (Fig. 10). In Fig. 10, two self-points are mentioned with their radius 1. If we set the detectors' coverage to 60 percent, the count of detectors required to covered the anti-password region is 13, which is shown in the figure along with their individual detector radius.

The pseudo-code for the binary space implementation algorithm is given in Appendix (Fig. 27).

The input and output of the detector generation algorithm are mentioned in the pseudo-code. Code block in line 8-12 checks whether candidate detector, 'x' falls in the existing list of detectors. If that falls, the failed count will be increased to keep track of total failed attempts. Code block in line 13-18 computes the minimum distance among all the self-points and the candidate detector, x point. Code block in line 20-26 checks the overlap with the candidate detector, x with other existing detectors and decides whether it can be included in the existing detector set. Code block in line 28-31 adds the candidate detector, x to the detector set and checks the termination criteria to exit or not. The final set of detectors is then returned. Some relevant terms of B-NAS is discussed in the following subsections:

4.2.1 Confusion Parameter for B-NAS

This term defines the radius of self-points in the binary space. In the binary space implementation, confusion parameter is set to 5 (subtle value in comparison with the total dimension of the space).

4.2.2 Expected Coverage of Non-self space

In binary space, detector coverage is approximated as a normal distribution [26]. For this distribution, mean is $N/2$ and Standard Deviation is $\sqrt{N/2}$, where N is the dimension of the space. The coverage of a detector of radius 'r' ($r << N$) is shown pictorially in the Fig. 11. The gray area denotes the coverage (or volume) of the detector in the total representative space.

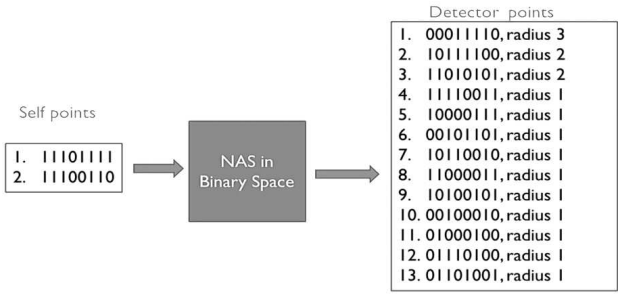

Fig. 10: **Sample detector space Generated using binary implementation of NAS. The configuration are Dimension=8, total self-points=2, Detectors minimum coverage=0.60, self-point radius=1.**

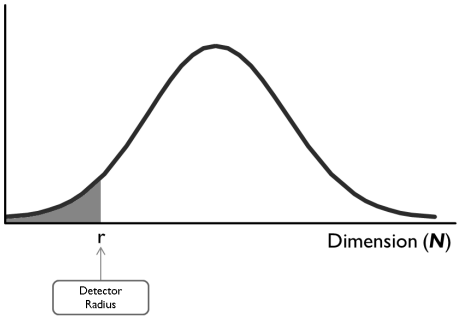

Fig. 11: **The gray area shows the volume of the detector with radius r.**

4.3 Modification of the self-region for R-NAS and B-NAS

To remove some existing self-points in real and binary space implementations, the marked to be removed self-points are simply removed from the existing set of self-points. If the number of removed self-points exceeds a threshold, then the detector set is generated again to cover up the empty spaces and to create larger detector spheres. The pseudo-code is given in appendix for removing a self-point(Fig. 28). According to pseudo-code, if the total number of self-points after the removal is less than a given threshold value, the detector set will be generated again with the given list of self-points. Otherwise, existing detector set can be considered as current detector set.

The case is slightly different if some new self-points are added in the self-region. All detectors are removed that collide with the new self-points and then detectors are generated again with the new self-point set and the existing detector set to cover the vacant non-self-region.

When the number of added self-points crosses a threshold limit, the detector set is generated again from the scratch to get rid of the smaller detectors and merge them in larger ones. This idea is shown in the given pseudo-code in Appendix (Fig. 29).

To update some existing self-points, the old self-point entries are removed and the updated self-point entries are added in the self-region. Corresponding steps for removal and addition of self-points are taken as described previously. Again, the total detector set is regenerated when the number of self-point update crosses a threshold to minimize the number of smaller detectors. This concept is shown in the given pseudo-code in Appendix (Fig. 30).

4.4 G-NAS Implementation

As discussed in the previous section, grid approach uses a two-dimensional $N \times N$ Grid to model the total space. At first, the self-region is mapped to the grid space and later the detectors are formed from the un-occupied non-self-region.

4.4.1 Represent the space in Grid

The Grid is stored as a two dimensional array of integers with a specified dimension. Self-points that are mapped to particular grids (grids are identified by row number and column number), increases the occupancy value of that grid by 1. Hence, any grid of ith row and jth column having occupancy value ($M_{i,j}$) greater than 1 means the grid is not empty and some $M_{i,j}$ self-points are mapped to that particular grid. The non-self-region grids are easily identified when their occupancy value is 0; indicating no self-points being mapped to that grid.

4.4.2 Confusion parameter in G-NAS

In grid space implementation, there are space size for a two-dimensional $N \times N$ Grid is N^2grids, but, the total possible space size is 2^n, where n is the length of the representation string of a self-point. Therefore, every grid is mapped by $(2^n/N^2)$ possible points. For every grid containing m self-points, $((2^n/N^2)$ -$m) \approx (2^n/N^2)$ points are considered as the confusion parameter.

4.4.3 Detection of Non-Self-Region in G-NAS

The grids that are not covered by the self-points are considered as the non-self-region. Hence, the non-self-region is created in constant time after getting the grids occupied by self-points. This is one of the advantages of grid based implementation over other implementations of NAS. Detectors can be generated by scanning the grid once and having consecutive grids in a row with occupancy value 0 considered as one detector. Run-length of each detector is determined by the count of consecutive grids in the detector.

In grid implementation, non-self-grids cover the total non-self-region. If only a fixed percentage of the anti-password region is desired to be covered, some detectors are marked for removal. In order to do that, detectors with minimum run-length are removed from detector list at first and the new achieved coverage is calculated. This process will go on until the expected coverage is achieved. This whole process can be performed with a single scan of the whole Grid and removing the detectors with the given run-length calculated at first. In the Grid based implementation, it is possible to increase or decrease the detectors' coverage with only a single scan through the Grid. It also indicates another advantage over the hypersphere based implementations [24], where the detectors are required to be generated from the beginning to get the necessary coverage value.

4.4.4 Saving the Detector Space

Detectors generated from the grid space implementation are stored by their run-length and starting index. In the implementation presented in this work, the detectors are saved in row-wise format and, in each row, the detector is stored with the starting column index and run length. The benefit of using row-wise format is, to find a particular point in the space, the row of that point can easily be determined and the search space is reduced to that row only to check whether it falls in a detector grid or self-point grid. This approach also does the compression of the space to store the whole detector region and it can be easily reconstructed from the saved information of the detector space.

4.4.5 Visual representation of a simulation of G-NAS

As discussed above, initially self-points are mapped in the grid space. The simulation involves a grid with 32×32 dimension and 500 self-points. It uses 256 bit SHA-256 encryption algorithm to get self-points. Then, the detectors are created to cover the non-self-space. After the creation of detectors, some detectors are removed from the detector list. This is done so that detectors do not cover the entire non-self-region. Some regions are left uncovered to induce ambiguity for attackers so that they cannot find out the self-region quickly. Fig. 12 shows the generated self-points (dark gray), detectors (black) and candidate detector for removal from

detector list (light gray - all detectors with run-length 1 and 2 in this example) in the grid. Only the surviving detectors (black blocks) will be saved for the NAS. To increase the obfuscation level, smaller detectors (light gray blocks) can be omitted for saving space.

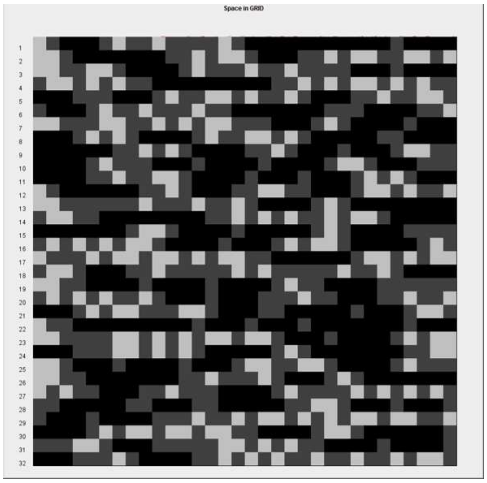

Fig. 12: Self-points (dark-gray), detectors(black), and candidate detectors marked for removal (light gray) of run-length up to 2 in the 32×32 Grid. Here 390 grids are dark gray grids, 412 grids are black grids, and 222 grids are light-gray grids with run-length 1 and 2.

4.4.6 Mapping of Self-points to Grid

Three different approaches of grid based implementation have been implemented and tested. They are:

- Mod-based approach
- Two-layer approach
- XOR-based approach

All grids are 2-dimensional with dimensions of $2^i \times 2^i$, where the value of i is dependent on the number of user information (self-points). In each of the cases, the hashed self-point string is divided into two halves. But the data in each half is processed differently in these implementations.

4.4.7 Mod-based G-NAS

In this method, each part of the divided hashed string of the self-point is considered. Last i bits are taken from the binary representation of each of the parts. This value denoted by these i bits provides the modulus if the string was divided by 2^i. Then both of the parted string is converted into corresponding values. This self-point is mapped in the grid using those computed values. Other bits are simply discarded from any computation. Here the projection scheme is fixed for every possible self-points unless the grid dimension is changed.The pseudo-code of the Mod-based projection algorithm showing the above concept is given in appendix (Fig. 31).

An example of the mod based approach of the Grid based implementation is illustrated in the Fig. 13. For example, it is assumed that the hashing function produces a binary string of length 16. The string is divided in two parts. Suppose the dimension of the grid is 16×16 $(= 2^4 \times 2^4)$. So, the last four bits from each part is considered. The value of these four bits resembles the reminder if the part is divided by $16(= 2^4)$. In this example, the first part produces $(1101)_2$ i.e. 13 and the second part produces $(1001)_2$, i.e. 9, so, the string will map to the cell $(13, 9)$ in the grid.

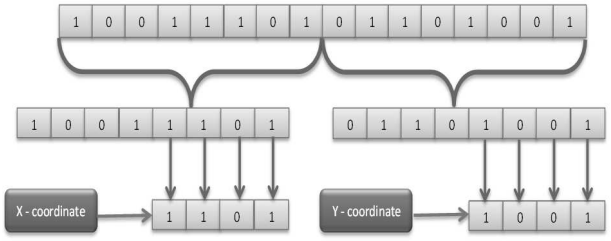

Fig. 13: **Mod based approach of mapping points in the Grid.**

4.4.8 Two-layer based G-NAS

Each part of the divided hashed string is further divided in two equal sub-parts. Last $i/2$ bits are taken from the binary representation of each of the sub-parts. Both of the $i/2$ bit length binary substrings are concatenated to from a i bit length binary string. This binary string is converted into corresponding value. The same procedure is done with the other part of the hashed string. Both the values are then used to map the self-point in the grid. All other bits are discarded. Again, the projection scheme is fixed for every possible self-points unless the grid dimension is changed. The pseudo-code of the two-layer based projection algorithm showing the above concept is given in appendix (Fig. 32).

An example of the two layer based approach of the Grid based implementation is illustrated in the Fig. 14. For example, it is assumed that the hashing function produces a binary string of length 16. The string is

divided in two parts. Each part is again divided into two equal sub-parts. Suppose the dimension of the grid is 16×16 ($= 2^4 \times 2^4$). So, the last two bits from each sub-parts is taken in consideration. The same procedure is repeated in the other part of the string. In this example, the first part produces $(0101)_2$, i.e. 5 and the second part produces $(0101)_2$, i.e., 9. So, the string will map to the cell (5,9) in the grid.

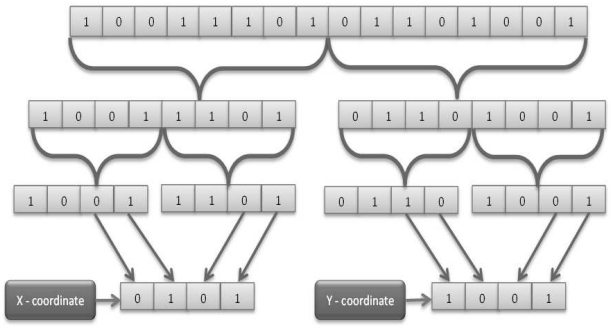

Fig. 14: **Two-layer based approach of mapping points in the Grid.**

The pseudo-code of the two-layer based projection algorithm is given in appendix (Fig. 32).

4.4.9 XOR-based G-NAS

Initially, i positions are randomly selected from the hashed string. Then, for each half of the hashed string, every j (power of 2) consecutive bits are XOR-ed and replaced by the result. That leaves each of the halved hashed binary string compressed in a ratio of j to 1. From the compressed string, bits from the selected positions are concatenated. That produces a binary string of length i. This binary string is converted into corresponding value. Both values are used to map the self-point mapped in the grid. All the self-point hashed string uses the same i positions selected at the beginning. More bit information is involved in the mapping of self-points in this projection approach. This projection scheme chooses new random positions every time the scheme is initiated. So this scheme will also produce different projection for the same password in separate implementations. The pseudo-code of the XOR- based projection algorithm showing the above concept is given in appendix (Fig. 33).

An example of the XOR-based approach of the Grid based implementation is illustrated in the Fig. 15. For example, it is assumed that the hashing function produces a binary string of length 32. This example shows a scenario with a compression ratio of 2: 1. Every two consecutive bits are XOR-ed to produce the new string. So the new string has a length of 16. The string is divided into two parts. Suppose the dimension of the grid is 16×16 ($= 2^4 \times 2^4$) and four random bit positions are 1, 4, 6 and 7. So, bits from those positions are taken in order. The same procedure is repeated in the other part of the string. In this example, the first part produces $(1111)_2$ i.e. 15 and the second part produces $(0101)_2$ i.e. 5. So, the string will map to the cell (15, 5) in the grid.

The pseudo-code of the XOR based projection algorithm is given in appendix (Fig. 33).

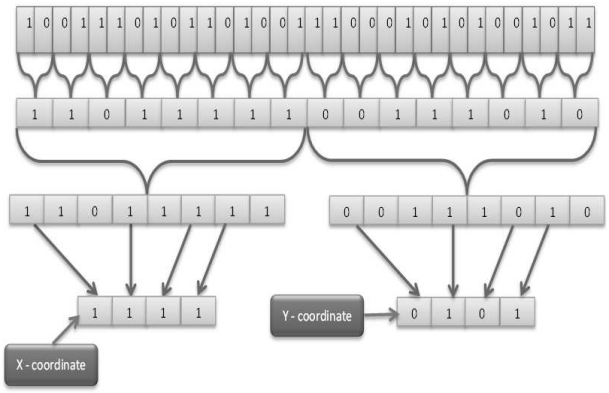

Fig. 15: **XOR based approach of mapping points in the Grid.**

5 Architectural Details about Implementing Prototype System

For prototype purposes, all the implementation has been designed and implemented in a virtual infrastructure. Every server and access terminal is deployed as a virtual machine with network interfaces. Through these interfaces, each VM is connected to virtual switches that represent the network segments as shown in Fig. 16.

In the network layout, there are three main network segments, which are administrative network, datacenter internal network, and access network. These network segments are created using three Firewalls (FW) / Intrusion Prevention Systems (IPS), which are present in all three segments.

The administrative network is basically a private network consisting of an administrator machine, routers, FW/IPS Segment 3, and FW/IPS Management Interface. The administrator device has access to Lv2

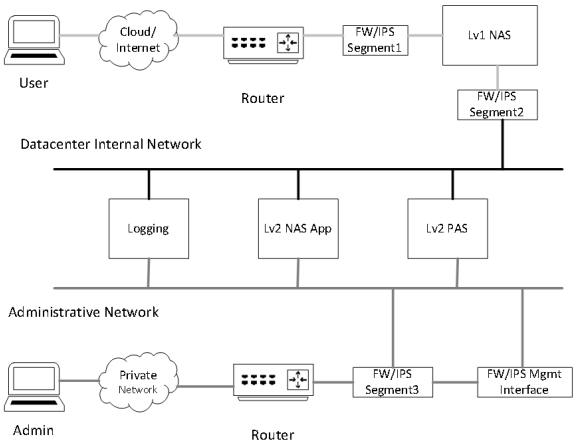

Fig. 16: **Different Components shown in the Network Layout.**

NAS App server to create user accounts as well as to create or update the anti-password database. Any newly created or modified user accounts are stored in Lv2 Positive Authentication Server (PAS). These user accounts are used to generate anti-password data using Lv2 NAS App server and pushed to Lv1 NAS Server.

The data-center internal network has four components which are: Logging Server, Lv2 NAS App Server, Lv2 PAS, and FW/IPS Segment 2. Lv2 NAS App Server and Lv2 PAS are mentioned while describing the administrative network, whereas these two components are crucial for the functionality of the system.

As for the access network, it has direct access to the clients from the outside network such as the Internet. This network has three components which are Lv1 NAS Server, FW/IPS Segment 1, and router. As mentioned above, Lv1 NAS Server receives anti-password data from Lv2 NAS App server, and this data is used to validate the authenticity of the clients at the access network.

The NAS is composed of seven different components. These are as follows:

1. NAS Clients
2. Lvl1 NAS Detector Server
3. Lvl2 NAS Application Server
4. Lvl2 Positive Authentication Server
5. NAS Log Server
6. NAS Scheduler Database
7. NAS Database

The details of each of these components are discussed in the next sections.

5.1 Layer 1 Components

5.1.1 NAS Clients

NAS Web Client

The login page requests username and password for users that want to access the system. After the credentials are submitted, the page then uses them to generate the self-point for that user using the same process that was used by the Level two server that created the detectors. The dynamic salt for each user is looked-up in memcached using the digest that results from hashing the concatenation of user and static salt. After obtaining the dynamic salt, it is concatenated to the input password, then it is hashed and the result is concatenated to the username and hashed again to obtain the digest. Depending on the type of implementation chosen, the appropriate algorithm to process the digest is used. The resulting self-point is then compared against the set of detectors to look for a match. If a match is found, the login is unsuccessful. If it does not match any detectors, it passes on to the next layer. The same web page executes a call to a stored procedure on the second layer to get the positive authentication of the user.

The login page returns the result of both the negative and positive authentications along with the summary of the detector results and the self-point. In a production implementation, this data should only be used for diagnostic purposes and the user should only see the result of the authentication, either success or failure.

NAS Windows Client

The Windows 8 Client is running the pGina Credential Provider replacement to allow easy access to the login interface. We have coded a pGina plugin in C# to contact the Lvl1 NAS Detector Server and the Lvl2 Positive Authentication Server. This allows us to use any authentication mechanism on the server and return if the user is allowed or denied access. Due to the simple socket connection of the client, the Lvl1 NAS Detector Server can be reused from previous proof of concept with only minor code changes.

5.1.2 Lvl1 NAS Detector Server

Lvll1 NAS Detector Server is a PHP server that communicates with Memcached to get the current NAS Detectors. The user account information that is provided to NAS Windows Client is validated against the detectors. If there is a match, the user access is not allowed.

If there is not a match, the user credentials are then compared to the Lvl2 Positive Authentication Servers (PAS) user account information. Currently, we are using a Microsoft Active Directory server for positive authentication.

5.2 Layer 2 Components

5.2.1 Lvl2 NAS Application Server

We implemented the Layer2 Negative Authentication Application (L2NAA) for Linux 2.4/2.6 platform including CentOS 6.4. The implementation of the L2NAA is designed to manage user account, create an anti-password database, push the anti-password database to Layer 1 Server, view statistics, and logs, and configure input and preference parameters.

5.2.2 Lvl2 Positive Authentication Server

In the NAS architecture, Microsofts Active Directory Server fulfills the requirement of a positive authentication source. As part of the NAS implementation, Active Directory LDIF entries were updated to contain the *userPassword* attribute. This attribute was used to provide the user accounts hashed password. The hashed password will be used to create the self-points needed by the NAS application. In order to use the *userPassword* attribute (i.e., to make it retrievable by the directory search), we had to set the *dSHeuristic* to false for the *fUserPwdSupport* entry.

5.2.3 NAS Log Server

This module shows the number of anti-password based on different algorithms. Logs are stored in ArcSight Logger. Logs stored in ArcSight Logger can be used for future forensic analysis. Arcsight Logger dashboards have been configured to show the most relevant information from the status messages. Column graphs show users trying to log in in Lvl1NAS and Lvl2PAS. Pie charts show the successful vs. failed login attempts in Lvl1 and Lvl2. Going forward, a better coverage of Lvl1 will reduce the failures in Lvl2.

5.2.4 NAS Scheduler Database

The quartz framework is used to schedule and run the jobs in the system. We have a fixed number of jobs in the system. These jobs are configurable based on the valid *'cron'* expression.

5.2.5 NAS Database

This module is used to generate and store anti-password database using different algorithms such as BNAS, R-NAS, and GNAS.

5.3 Account Creation and Authentication Flow

The account creation and authentication processes are shown in Fig. 17. There are two sections below that give a description of the account creation and authentication flow. To achieve an account creation, the administrator must complete the five steps mentioned below to upload the negative database to NAS L1 Server.

When clients request to access resources they go through layers 1 and 2, which is called a client authentication process. Below there is a description of this process given in three steps, and there is also an illustration found in Fig. 17.

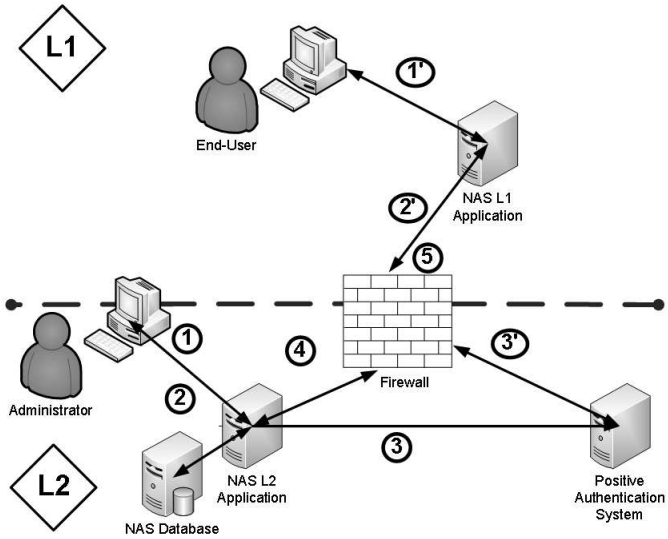

Fig. 17: **The Steps of Authentication process in NAS.**

5.3.1 Account Creation Flow

The administrator creates user accounts using steps 1, 2, 3, 4, and 5, and these steps are described below:

- 1: After the user account is created, static salt is concatenated with User-name (ID) and password for self-point creation.
- 2: Hashed ID (i.e. static salt, user name, and password) passed to NAS Database.

– 3: User account information is passed to the Positive Authentication System (PAS).
– 4: Detectors are created using the Hashed user information from NAS Database.
– 5: Detector list is passed to NAS L1.

5.3.2 Authentication Process Flow

The clients are authenticated using steps 1', 2', and 3', and these steps are described below:

– 1': ID and password are provided to NAS Ll for authentication.
– 2': ID and password are concatenated with static salt (i.e. application private key).This information is converted to a point and checked with the list of detectors. If the point does not fall into any detectors, the user credentials are passed through the firewall. Otherwise, this access request is denied.
– 3': PAS user credentials are validated, and if valid, an authentication token is passed.

6 Experimental Design and Results

6.1 Experimental Setup

A dataset of 33 million user passwords is used as the source of the passwords used for simulations. This dataset is extracted from SkullSecurity [32] website. This is a dataset of exposed user password of RockYou site [7]. As these passwords belonged to a real authentication system, so this dataset can be used to simulate a real password-based authentication system. To generate the usernames, another dataset of most common surnames of U.S. is used [5]. This dataset includes all surnames with over 0.001% frequency in the US population during the 1990 census. The performances of each configuration are indicated by the detector coverage. Detector coverage is the percentage of spaces covered by the algorithms. In other sense, it is the probability of a particular algorithm configuration of detecting an unauthorized user. All rates are calculated from the average of 20 runs by using Monte Carlo Simulation [28] with one hundred thousand trials for each configuration. The coverage by the detectors is calculated by using the following ratio:

$$\frac{\text{total number of trials detected by the generated detectors}}{\text{total number of trials}}$$

The False rate can also be calculated by subtracting the coverage from 1 (one).

6.2 Evaluation Criteria

Three different implementations of NAS presented in this paper are evaluated based on the following criteria:
- The coverage of the approaches with different number of self-points (positive password) with varying their confusion parameter.
- False positive rate.
- The time required generating the set of detectors.
- Detection time.
- Detectors' storage size.

The details results for three different approaches are discussed separately in later subsections. The comparison of three implemented approaches are illustrated in 6.6.

6.3 R-NAS Results

Experiments for real space implementation of NAS are done with different number of self-points generated from the mentioned dataset. The First experiment was conducted to show the relationship between self-point radius and the detector coverage. This experiment was carried out with 1000, 3000 and 5000 self-points. The result is summarized in Table 1 and showed in Fig. 18. From Fig. 18, it can be seen that detector coverage

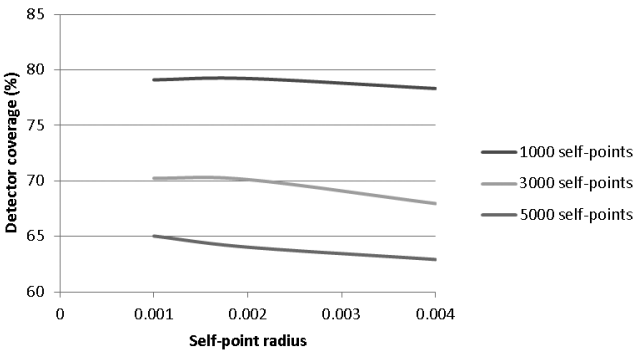

Fig. 18: **Scatter diagram for coverage by detectors with different self-point radius.**

decreases slightly with the increment of the self-point radius. This self-point radius is also known as the confusion parameter. The reason behind this result is as follows: self-point radius with larger value will occupy more spaces leaving less free spaces, causing detectors to have slightly smaller radius. So the detector cover-

Table 1: **Difference in detector coverage with different self-point radius.**

	Detector Coverage(%)		
Self-point radius	**With 1000 self-points**	**With 3000 self-points**	**With 5000 self-points**
0.001	79.11	70.25	65.04
0.002	79.23	70.13	64.04
0.004	78.33	67.97	62.93

age decreases a slightly with the increase of self-point radius.

Next result shows the relationship between the actual detector coverage achieved with the expected detector coverage by the R-NAS algorithm. The expected coverage was varied from 85% to 98% for 1000, 3000 and 5000 self-points. The result is summarized in Table 2 and graphically represented in Fig. 19. We get the lower detector coverage value as many detectors are overlapped one another and the overlapping region covers the same space. But in our coverage calculation, only overlaps with two detectors are measured. Hence the actual detector coverage value is lower in comparison with the expected coverage value.

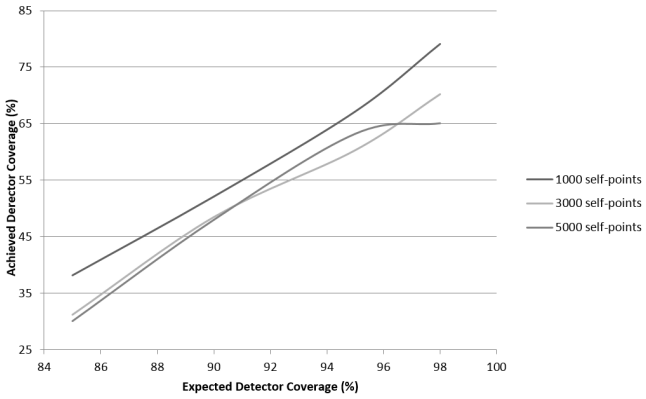

Fig. 19: **Scatter diagram for achieved detector coverage by the generated detectors with different expected detector coverage.**

6.4 B-NAS Results

For binary space implementation, the average count of detectors is compared with different coverage value for detectors. The simulation is done with 1000 and 3000 self-points and two different hashing algorithms. The result is shown in Fig. 20. It is evident from the figure that, with the increase of the password points (self-points), the total number of detectors increases. Also, it is noted that with the increase of the minimum coverage for a fixed number of self-points, the total number of detectors are increased. Additionally, the dimension of binary space plays a role in the total detector count. It is derived from the figure that, for 256 dimension space, the total number of detectors is less in number than that of 128 dimension space.

The reason is as follows. As the number of self-points and confusion parameter are fixed, for changing 256 dimension from 128 dimension, the total number of points covering the non-self-space are increased exponentially. Hence the detectors created in 256 dimensions have a higher radius which covers more space in comparison with that of 128 dimensions. So the total numbers of detectors are going down in 256 dimensions to cover the same minimum coverage. One further observation is with the increase of self-points, the total number of detectors are also increasing for different encryption algorithms.

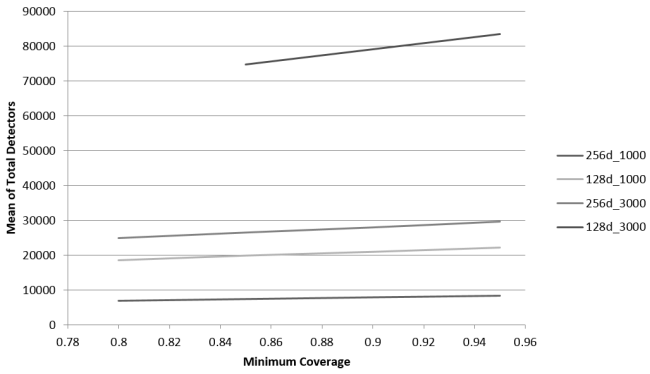

Fig. 20: **Scatter diagram for total number of detectors with different expected coverage value.**

Mean detection rate for B-NAS implementation is compared with various values of minimum coverage. The results are shown in Fig. 21. From Fig. 21, it is clear that the detection rate increases linearly with the increase of minimum coverage. However, the rate of increase is not constant in different scenarios. This is ob-

Table 2: **Difference in achieved detector coverage with different expected detector coverage**

	Detector Coverage(%)		
Self-point radius	**With 1000 self-points**	**With 3000 self-points**	**With 5000 self-points**
85	38.12	31.15	30.08
90	52.08	48.47	47.98
95	67.15	60.32	63.18
98	79.11	70.25	65.04

vious in the sense that with the increase in dimension and number of self-points; the total points covered are increased not in a fixed ratio.

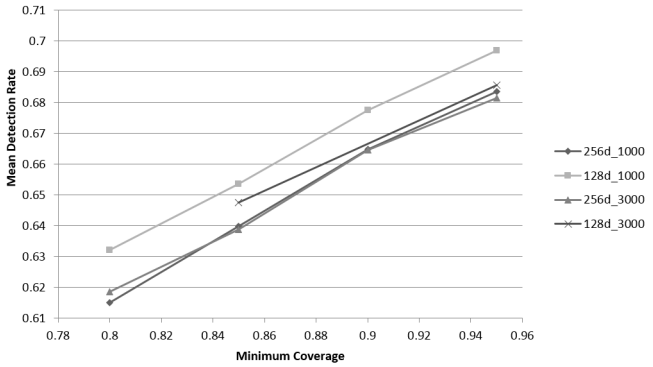

Fig. 21: **Scatter diagram for mean detection rate with different expected coverage.**

For a fixed dimension, the mean detection rate is higher for smaller self-points. With the increase of the number of self-points, the total non-self-space is reduced and hence, the generated detectors has more overlap for higher self-points which causes the detection rate to go down. For a fixed number of self-points, the mean detection rate is higher for 128 dimension than 256 dimension space. This result can be explained with the help of a total number of detectors count. As the total number of detectors is high for 128 dimensions, the proportion of not covered non-self-space is reduced for this scenario and hence the detection rate goes high.

6.5 G-NAS Results

Experiments were carried out using the dataset mentioned above to generate the statistical properties of the discussed algorithms. For each simulation, 20000 passwords and usernames are randomly chosen from the datasets and combined to form the authentication credential (Self-points). SHA-256 is used to hash the authentication credentials so that the spread of the hashed string is even over the space. Along with detector coverage, detection rate and false rate are also calculated to indicate the performance of the implementation. The detection rate is the calculated detector coverage with trials. False rate is the percentage of invalid credentials not detected by the detectors. Grid dimension is varied from 2^6 to 2^{11} to get the detection rates and false rates for each algorithm.

Different sizes of self-point samples were projected into grids of different dimensions to get the appropriate dimension for a particular size of self-point list. The dimensions of grids are varied from $(2^5 \times 2^5)$ to $(2^{11} \times 2^{11})$. Coverage of the self-region is calculated in each simulation and results of these simulations are shown in Fig. 22. These simulations are done with the mod based approach. Other two approaches also show very similar pattern with the mod based approach. Another important point is that encryption size does not matter in the self-grid coverage in Grid implementation for any approach. The reason is, always, some specific bits are used from the whole encrypted bit string for mapping. From Fig. 22, the appropriate grid dimension for some specific size of self-points can be found. For example, if the targeted coverage by self-region is below 10%, then for a self-region of 10000 points, the grid dimension will be $2^9 \times 2^9$ or 512×512.

For a given number of password points, the detection rate (the percentage of non-password points is detected by the available detectors) increases with the increase of grid dimension. Also, the false positive rate (the percentage of non-password points falls in the self-grids) decreases with the grid dimension.

The trend for detection rate and false positive rate follow the same pattern in the case of removing of some small detectors (ex: removal of detectors with run-length 1 and 2). The Fig. 23 shows the effect of change of Grid size on Detection Rate and False Positive Rate (FPR) without any removal of detectors and removal after detectors of RL 1 and RL 2 for a fixed set of 20000 password points in mod based projection. From the Fig. 23, it can be seen that the detection rate graphs before removal of any detector, after removal of RL 1 detectors and after removal of RL 2 detectors almost overlapped over each other. It is also true for the false positive rate

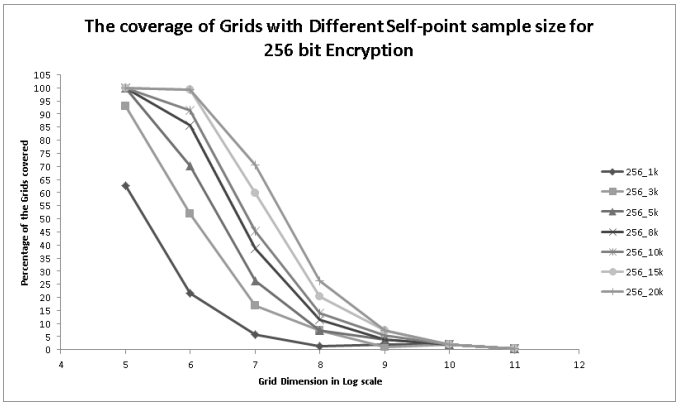

Fig. 22: **Coverage of self-region in different grid sizes for various sample sizes of self-points for** 256 **bit encryption.**

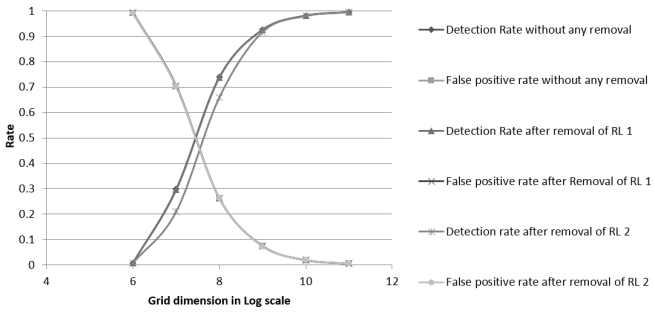

Fig. 23: **Effect of grid size on detection rate and false positive rate without any removal of detectors and removal after detectors of RL** 1 **and RL** 2 **for a fixed set of** 20000 **password points in mod based projection.**

graphs. Therefore, from the Fig. 23, it can be concluded that, after removing the small length detectors the grid still shows the same pattern for detection rate and false positive rate as before.

6.6 Comparison Among Approaches of NAS

Each of the different implementations of NAS has its own advantages and disadvantages. Table 3 summarizes the comparison of different methods of NAS. As shown in Table 3, real space and binary space model use probabilistic approach for detector generation and maps the same region of the model space for the self-points every time if the parameters to create the self-points remain the same. Detectors are generated as hyperspheres for both models and detectors may overlap each other. To update the existing detectors and self-points, comparison with all available self-points and detectors is needed for both real and binary space model. The grid space model uses a deterministic approach for detector generation and some variant implementation (XOR method) of grid space model maps different regions for the same set of self-points each time the self-points are generated. Detectors and self-points are generated as non-overlapping grid cells in the grid space and only one comparison with a grid cell is needed to update an existing grid space model.

Table 3 shows the coverage by detectors and number of detectors generated for each space model implementations for ten thousand self-points with SHA-256 hashing. The real and binary space based implementations generate a lot of detectors but achieves detector coverage of about 68%. These two implementations need more space to store higher number of detectors in compared with grid implementation. The grid implementation generates relatively small number of detectors and achieves great detector coverage of about 90%. Binary and real space implementation have very low false positive while grid space implementation has a relatively higher false positive rate. Grid space implementation generates the same set of detectors for same set of self-points with same configuration parameters but real and binary space implementation will generate different sets of detectors in every run of the algorithm. However, in various runs of G-NAS, the configuration can be changed to set different sets of detectors for the same login information. Binary space implementation maps each different username-password pair to a unique point in the model space. Real space implementation maps 2^{216} different username-password point to a single point (for 4 dimensional model considering 4 points after decimal point). Grid implementation maps 2^{238} different username-password points to a single grid cell (for a $2^9 \times 2^9$ grid) in the corresponding space models. B-NAS and R-NAS approach require more space and time to generate detectors than G-NAS. G-NAS approach (deterministic solution) can be configured to a specific grid size that can control the amount of space used by the detectors and self-points. Hence, system administrators can easily tune the parameter (grid size) that best fit the requirement of any system. Moreover, the detection rates of the G-NAS is 95% with 95% level of significance that can be found using χ^2-test of the goodness of fit, which proves the effectiveness and validity of the proposed approach.

In this current implementation, we tested the time requirement (third evaluation criteria) for generating a large number of negative detectors using three different

Table 3: **Comparison among different NAS approaches**

	Binary Space	**Real Space**	**Grid Based**
Approach	Non-deterministic	Non-deterministic	Deterministic
Mapping	All info	Curtailed info. (Can be varied)	Curtailed info. (Can be varied
Update of password profile	Comparison with all existing Self-Point and Detectors	Comparison with all existing Self-Point and Detectors	Comparison with at most one Self Grid or one Detector
Regeneration of password profile	Maps same region for fix points	Maps same region for fix points	Maps different region for self points (Random bit)
Space size	Fixed	Can be Varied (Decimal places)	Varied (On password profile size) - Small
Detector and Self-point	Hypercube	Hypersphere	Rectangle
Overlap of detectors	Considered	Considered (50% of radius)	No Overlap
Coverage	62.26%	68.17%	95%(95% level of significance)
Dimension of Space	256	4	2
No. of Detectors	116,382	100,000	514
Total Password Space	2^{256} points	1000^4 points	2^{18} cells
Mapping of points	1:1	$1{:}2^{216}$	$1{:}2^{238}$
Self-Region Coverage(%) **or False Positive Rate** (%)	7.85×10^{-22}	4.93×10^{-6}	1.42

Table 4: Time required to create 1 million negative detectors using three versions of NAS

Three Implementations of NAS	**Time Taken**
R-NAS	53.17 ms
B-NAS	135.35 ms
G-NAS	3 ms

Table 5: Time Statistics for verifying 100k authentication requests in the First layer of NAS approaches

Three Implementations of NAS	**Detection time**
R-NAS	Average: 7.8 ms and Standard deviation: 2.3 ms
B-NAS	Average: 6.3 ms and Standard deviation: 2.5 ms
G-NAS	Average: 5.2 ms and Standard deviation: 1.3 ms

Table 6: The space required to store 100k detectors in the first layer by three different NAS approaches

Three Implementations of NAS	**Required Space**
R-NAS	100k *(5*4 bytes) $\approx$ 200 MB
B-NAS	100k*(32 + 4) bytes $\approx$ 360 MB
G-NAS	100k*(8 bytes) $\approx$ 8 MB

versions of NAS. The negative detectors are created in the server side applications and pushed to the first layer of NAS to handle all the incoming password requests. To test the time requirement, we used DELL PowerEdge R530 Server with 44 cores at 2.2 GHz and 320 GB RAM. The results are shown in Table 4. According to the table, the required time is quite reasonable for generating 1 million detectors. These generated detectors will be stored in the server side and hence, there is no burden on the client side regarding storage space. In addition, due to lower requirement of detectors' generation time, system administrators can easily generate a new set of detectors and thereby reduce the chance of compromise the mapping of detector space in the first layer.

To test the average detection time (fourth evaluation criteria) required for an authentication request, an experiment is conducted with 1 million detectors in the first layer for 100k authentication requests in three different versions of NAS implementation. This test determines the efficacy of adding one more layer in the actual authentication process. Same server configuration is used to run this experiment. The results are shown in Table 5. According to the table, the required time to test 100k authentication requests takes couple of milliseconds. These values signifies that incorporating a negative layer does not make significant overhead in detecting the authentication attempts for verification.

To compare the usability of NAS approaches, the required space (fifth evaluation criteria) is calculated to store 100k detectors and the results are shown in Table 6. According to the Table, it is clear that there is not much overhead of space for implementing NAS as an authentication approach. As these created detectors will be stored in the server side, the client application performing authentication does not experience any difference in terms of memory requirements.

In summary,the overall detectors coverage is higher with lesser number of detectors in G-NAS in compared with B-NAS and R-NAS for the considered dataset. The coverage values of three approaches can be increased with the decrease of the confusion parameter for B-

NAS and R-NAS and with the increase of the grid dimension for G-NAS. Reducing the confusion parameter triggers a higher number of detectors to be generated in B-NAS and R-NAS. Similarly, increasing the grid dimension requires generating more grid cells in G-NAS. These issues play a role in determining the value of the last three evaluation criteria mentioned in section 6.2. Table 4 highlights the comparison of three approaches based on third evaluation criteria. According to the table, G-NAS takes the lowest time to generate the required number of detectors to achieve better negative space coverage. Table 5 and Table 6 show the comparison of these approaches based on fourth and fifth evaluation criteria respectively. In both cases, G-NAS takes lesser time and space in compared with B-NAS and R-NAS implementation. B-NAS and R-NAS perform almost equally for last three evaluation criteria.

The above mentioned experimental results proves the efficiency of NAS concept in terms of various evaluation criteria. These results also demonstrate the validity and reliability (in terms of false positive rate mentioned in Table 3) of the three different implementations of NAS.

6.7 Password cracking test

In order to verify the efficacy of the proposed NAS approach, the password cracking test is designed to simulate the guessing attacks. 100k different access requests are made to test the NAS approaches and the number of times these requests fall into the self-region are counted. Generally, the first layer detects and block all the invalid requests (those which fall into the anti-password space or negative space). The results of password cracking test for R-NAS and B-NAS are shown in Table 7 and Table 8.

In R-NAS scenario, for 100k test points, only 1 point pass through the second layer and falls into the self-region. It means one invalid request is considered as a valid request. This scenario happens if the dimension of the space is small (here 3 points) but with the increase of the space dimension, no invalid request has fallen into the self-region. For B-NAS no invalid requests fall into the self-region. In general, If the self-radius (confusion parameter) is significantly small, the possibility of mapping of the invalid requests fall into the self-points region is very less. Hence, with the proper tuning of the confusion parameter for B-NAS and R-NAS, the password guessing attack can be fully prevented in the proposed approach.

The proposed NAS approach also prevents the side channel attacks. All the access requests are initially verified in the first layer of NAS. If the first layer cannot catch an incoming request, it moves the request to the second layer for additional checking. From an intruder perspective, the information contained in the first layer is the commonly available information to be compromised. First layer of NAS contains only the detectors information and no information regarding the positive password space. In the proposed three implementations of NAS, the total number of detectors are exponentially larger in comparison with the number of self-points (passwords). The created set of detectors is hashed output. Hence, compromising the set of detectors provide the attackers a list of hashed outputs. The strength of the hashed output can be increased by choosing the higher valued hash functions (SHA-256 or SHA-512). As the chance of hash collision is very few, the probability of the valid passwords having the same hash as the compromised detector points is close to zero. Again as the hashing is considered as an irreversible process, it is not possible for the attackers to get the actual string of characters that construct the detectors.

In addition, the set of detectors are updated on a regular basis (they will change after every run of B-NAS and R-NAS due to the non-deterministic approach) and hence, any extracted information regarding the possible password space may not be applicable after a particular period. Hence, with the incorporation of two layers in NAS and putting only the non-self-information in the first layer, the side channel attack can be significantly reduced. It is true that the proposed NAS may not prevent all the types of side-channel attacks but with the two layer based approach, it makes harder to access and compromise the positive information of the legitimate users.Another advantage of this two layer approach is that the unsuccessful access requests can be later be used for forensic analysis to extract any pattern of the attackers to further strengthen the design of NAS.

7 Comparison with Positive Authentication System (PAS)

In our proposed system, the first layer composed of anti-password information (negative detectors). The more secured second layer contains the positive password space. In general, passwords are first hashed and represented in a high dimensional space. According to our design, anti-passwords are less risky to reveal passwords as compared to a hashed password file.
In the case of an offline attack on the first layer, when the attacker gets access to the identification information file, the Anti-P file is less vulnerable to the revelation of the password than the password file. From the password file, the attacker knows which hashes are valid so that

Table 7: **Result of Password Cracking Test for R-NAS implementation of NAS**

Hash Dimension	**Hashing Algorithm**	**Points considered after decimal**	**Number of Self Points (passwords)**	**Self-radius**	**Coverage**	**No of attempts falling on self-point**
256	SHA-256	3	1000	0.001	76.87	1
256	SHA-256	4	1000	0.0001	75.734	0
256	SHA-256	5	1000	0.00001	76.58	0
128	MD5	3	1000	0.001	76.476	1
128	MD5	4	1000	0.0001	77.24	0
128	MD5	5	1000	0.00001	76.982	0

Table 8: **Result of Password Cracking Test for B-NAS implementation of NAS**

Hash Dimension	**Hashing Algorithm**	**Total Number of self-points (passwords)**	**Confusion Parameter or self-radius**	**Expected Coverage**	**Detectors' actual coverage**	**No of attempts falling on self-point**
256	SHA-256	1000	5	0.8	0.612402	0
256	SHA-256	1000	5	0.85	0.636626	0
256	SHA-256	1000	5	0.9	0.656294	0
256	SHA-256	1000	5	0.95	0.676946	0
128	MD5	1000	5	0.8	0.624925	0
128	MD5	1000	5	0.85	0.647056	0
128	MD5	1000	5	0.9	0.667413	0
128	MD5	1000	5	0.95	0.687952	0

he/she can try with a rainbow file. In the case of Anti-P file, the attacker can know about the region where valid hash exists; however, he/she has no idea, which are valid ones or how many of them are valid. This is shown in Fig. 24. In this figure, the left-side diagram shows password file and the right-side diagram shows the anti-password file containing detectors. In our proposed design, the negative space is not exactly complimentary of positive password space. For example, if 93% of total possible password space is covered by detectors or anti-passwords, it does not imply that the remaining 7% region falls into a valid password region. If the actual password space lies in 4% of the total password space, it is almost impossible to the attackers to find out which actual 4% of total 7% space contains the actual passwords. Hence, the proposed system provides more obfuscation to the attackers to compromise on the actual positive password space.

In the case of an online attack on the first layer and compromised of the negative layer, the attacker can only see the output decisions from the negative authentication module. Through Capturing successful guesses from the negative authentication layer, the attacker cannot be sure to have guessed a real valid password. However, he/she may get an idea of positioning of password and anti-password space from those successful and unsuccessful guesses. But, this does not come to help,

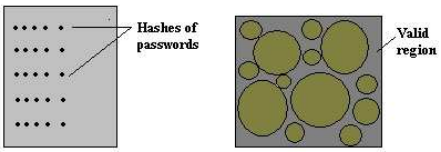

Fig. 24: Comparison of Information from password file and anti-password file. Exact valid hashes are found from the password file. But, anti-password file only reveals a region, from which any point (a point corresponds to a hash) can be valid or invalid.

as long as the hash function used in the system is irreversible, randomized, and uniform.

In addition, due to the lower requirement of detectors' generation time in NAS approaches, system administrators can quickly generate a new set of detectors and thereby reduce the chance of compromise the mapping of detector space in the first layer. This is also applicable for password reset scenario as system administrators can easily generate the new set of detectors within very short time and push the new detector set in the first layer. All these aspects prove the efficiency of NAS in terms of computational cost, and the proposed

approaches can perform better than existing positive password based methods.

8 Related contemporary Authentication Systems

Three implementations of NAS model are designed to deny illegal access requests and provide an invisible shield against various password based attacks in order to secure positive authentication information. Binary and real space based implementations provide non deterministic approaches to generate detectors while grid-based implementation provide a deterministic process to generate the set of detectors. Hence, system administrators can choose different implementations of NAS based on their coverage and detector generation principle. In general, G-NAS achieves higher coverage of non-self space with lower required time (3 ms) to generate the set of detectors as well as lower space to store the detectors (8 MB to store 100k detectors) and to provide less time (on average 5.2 ms for 100k requests) to check authentication requests. However, with the same configuration of the user accounts, the G-NAS approach always generate the same set of detectors, while B-NAS and R-NAS generate different detectors' set with every run of the algorithm. Hence, the non-deterministic approaches provide more diversity in the first layer and provide more obfuscation to the attackers in case of breach of the first layer.

The reliability of the proposed NAS approaches are measured. The reliability values of the proposed NAS approaches are extremely high like 99.89% for R-NAS, 99.78% for G-NAS and B-NAS (using the same dataset) with very nominal standard deviation of 0.013%. For this purpose, we have collected 20,000 random samples from the set of 100K password data and conducted the χ^2 test for the variance (σ^2) using the following hypothesis: $H_0 : \sigma^2 = \sigma_0^2$
$H_a : \sigma^2 < \sigma_0^2$ (for a lower one tailed test); $\sigma^2 > \sigma_0^2$ (for an upper one tailed test), and $\sigma^2 \neq \sigma_0^2$ (for a two tailed test) with T statistic, $T = (N-1)(\frac{s}{\sigma_0^2})$, where N is the sample size and s is the sample standard deviation. Here, $(\frac{s}{\sigma_0^2})$ compares the ratio of the sample standard deviation to the target standard deviation. Hence, the proposed NAS methodologies are extremely reliable in almost all considered circumstances and datasets. This test thus proves reliability of the NAS approaches and the corresponding reliability values follow the normal distribution, i.e., the probability of major fluctuations in the reliability values is very small (almost tending to 0). Apart from that, the NAS approaches are featured in MIT Geospatial Lab [1] [2] and it is thoroughly tested (both practically and statistically) for reliability with various datasets and found the methods are extremely reliable with insignificant deviations from the above-mentioned reliability values.

Moreover, the proposed NAS approaches can accurately detect the imposters in a significantly less amount of time (see Table 7 and Table 8), which makes these proposed approaches very reliable and feasible ways to do user authentication. In our prototype implementations of the G-NAS, B-NAS and the R-NAS approaches, the execution times are 2.4ms, 6ms and 3ms respectively (from the second execution onwards) in local-area settings along with the higher throughput rates in several standard HTTPS servers with very less standard deviation of 0.006% (tested using the χ^2 test for the variance with T statistic with 99% confidence) which are significantly better than the other contemporary approaches. The efficient time requirements for detector generation and verifying authentication requests and the lower overhead of storing the detectors in the first layer make the proposed NAS approach more viable to be used as a contemporary password based approaches.

Honeywords [25] system is considered as one of the related approach which uses some extra false credentials information along with positive password information. This system does not have any preliminary checks for authentication requests like NAS first layer. Again, both the actual and honeywords are stored in the same location or server. Only honeyword checker is stored on the separate server. In the proposed NAS approach (G-NAS for example), there is no chance of false negative. But for honeyword model, an alarm may be raised by a valid user by mistake if she chooses one of the honeyword as her password. Some other implementations use decoy passwords [1] as false credentials like honeywords which suffer the same issue as honeywords.

Shadow password concepts in UNIX [16] is one type of protection mechanism that stores the encrypted password file in a different location and only is accessed from system's root directory by an administrative user. This approach prevents an attacker who gains access to a system as a regular user and tries to steal the password file containing all the hashed passwords of the users. But the actual password and the shadow password file both locate on the same server. Hence, if that server is compromised, the password credential are compromised as well. On the other hand, the design of NAS is motivated to create a secure layer before the actual (positive) authentication information to prevent invalid

[1] http://geospatial.mit.edu/
[2] https://vimeo.com/98054594

access requests to pass through into the system in the first place.

Prior works of PYTHIA system [15] have explored the use of remote cryptographic services to harden keys derived from passwords or otherwise improve resilience to compromise.However, the modern PYTHIA system transcends existing designs to simultaneously support granular rate limiting, efficient key rotation, and cryptographic erasure. This set of features, which stems from practical requirements in applications such as enterprise password storage, proves to require a new cryptographic primitive that we refer to as a partially oblivious PRF [15]. Other related work on optimal distributed password verification system [6] mainly represents a highly efficient cryptographic protocol to protect user passwords against server compromise by distributing the capability to verify passwords over multiple servers. It is primarily a single-round password verification protocol and requires from each server only one exponentiation in a prime-order group.

On the other hand, our proposed NAS approaches present three different extremely reliable contemporary approaches for preventing guessing and side channel attacks by incorporating the concepts of negative and positive password spaces, which would significantly increase the level of obfuscation against all illegitimate users. The present NAS approaches have been implemented in two layers of which, the first layer is occupied by the detectors (non-self) and the second layer is occupied by the self-region(hashed and salted passwords). All the three NAS approaches are rigorously implemented and tested for performance, which establishes their better reliability. Hence, the NAS approaches could be considered as potential alternative approaches to defend users from the guessing and the side channel attacks.

An experiment is designed using a publicly available dataset [17] on three different NAS implementations and Pythia to show the wide applicability of both of these technologies as a robust authentication system.

8.1 Dataset Description and Conducted Experiment

This dataset contains mouse dynamics for ten different users[17].It consists of mouse movement events and left and right mouse button pressed and released events of various users. Two separate datasamples for each user were collected, one containing origianal owners of the respective accounts and the other where the identity of the current user is not known. For convenience, they will be referred to as S_1 and S_2 respectively. As mentioned by Fulop et al. [17], any illegal activities present in S_2 do not reflect malicious activities but instead usages by unauthorized users performing unspecified administrative tasks.

Several labels were found in the dataset that describes if a session was legal or not. These labels contain only labels for the 'public' part of S_2. As such, during the cleaning of the data, we ignore unlabeled samples that were found in S_2 since we would not know the identity of the user for that session. Hence, for the training data, we would only label the samples as according to who is belonging to whereas for the test set, we will include an additional label, which describes the legality of the sample.

The experiment with this dataset proves the applicability of the NAS approaches (B-NAS, R-NAS, and G-NAS) in a different domain called the mouse dynamics analysis that involves monitoring the way a user moves the mouse and the corresponding data have been used for authentication purpose. Here, we are mainly focusing on a specific single-stage action called the "mouse move," which is defined as the general mouse movement between two points [20]. The experiment can be extended to the other multi-stage mouse actions like drag and drop, point and click, etc. Table mentioned in [20] shows the values (mainly, numbers) of the extracted features from the data set. The user specific mouse movement action can be uniquely identified using sixty-six features [20] that have been extracted from the Balabit data set. Using these features, a specific users can be identified (active authentication). Next, these extracted feature values are user specific and must be protected from any type of adversarial intervention, and hence, we have used the NAS approaches for this purpose. Simultaneously, we have also used the PYTHIA approach for a comparative study.

Using the NAS approaches mentioned before, we have created positive and negative spaces for storing all the features (related to the mouse move action for all the 10 users) with suitable obfuscations (see section 7. In this experiment, we have kept all the feature values of a specific user together (positive spaces) and generate the corresponding negative spaces using the NAS approaches. If an adversary wants to replicate the legitimate user's mouse movement action, he must have a clear idea about the specific feature values with which the user action can be regenerated and, consequently, the adversary can be misclassified as a legitimate user. Quite similarly, using the PYTHIA approach [15], we have tried to hide the user specific mouse movement action related features (for given users) and the resulting encrypted values are stored (shown in Fig. 25). This experiment has been performed using a system with Intel core i7 (7th Gen, 32 GB RAM) processor with

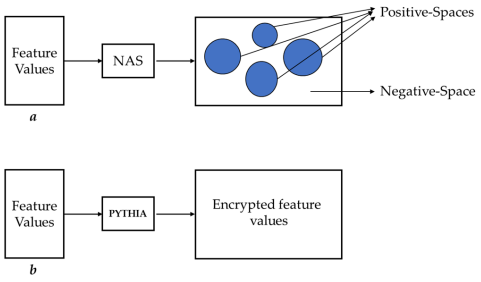

Fig. 25: Outline of the proposed experiment; a. Experiment with the NAS approaches, where the feature values have been stored in the positive spaces (blue circles), and the negative space encircles the self-regions; b. Experiment conducted using the PYTHIA approach

GTX 1070. The Balabit dataset[17] has been divided in training and the testing sets. Here, we have used the scikit-learn library to train a Random Forest classifier in Python that uses 512 number of trees and we used the Gini impurity splitting criterion. We have also fixed the maximum depth of a tree in the Random Forest to be 60, with the minimum samples per leaf node to be 1 and the minimum samples required for any node splitting to be 2. All these values were derived empirically through testing on the training and validation sets.

Now, we have tried to breach the user specific feature values stored using the NAS approaches and the PYTHIA system. The result of these two approaches are presented in Table 9. This comparative study is mainly focused on the accuracy, false positive, false negative and turnaround time (aka detection time). However, both of the approaches (NAS and PYTHIA) show almost similar accuracy in authenticating users. In some cases, we found the accuracy of the G-NAS is slightly better than the other approaches. This result can be data specific and may be different for some other data sets. However, in the accuracy of the NAS approaches is quite in conformity with the previous experimental results (see Section 6.6). Additionally, the turnaround times for the NAS approaches are similar to the results shown in Table 5. Hence, it can be concluded that the NAS approaches are quite resilient for various types of datasets and can be considered as very trusted and alternative defensive methods against the adversarial interventions. Finally, both of these methods are trusted and can be used as the potential alternative approaches.

One intrinsic limitation of NAS implementations is the creation of higher number of anti-passwords (detectors) because of the sparsity of hashed passwords in the representation space. Some other techniques are also existing to obfuscate information like Collisionful hashing [21], Bloom Filter [34], Cuckoo Hashing [29], etc. Although these methods are somehow space efficient in comparison with our NAS approach, some security issues still exist with those methods. Hashed output of login information can easily be changed or altered for these methods. But in NAS, encrypted negative information (use of static and dynamic salt along with one-way hash functions) is comparatively harder to alter and any compromise of this negative information will not reveal the positive password (user credentials) information. The relevance of Negative Authentication remains valid in the case of compromising of password database.

8.2 Attack Scenarios

Two attack scenarios relating to passwords are described here:

8.2.1 Side channel Attacks

If an adversary somehow compromises the first layer of NAS, all the negative space (i.e. non-self-space covered by the detectors) are revealed. But these leaked information do not contain any information regarding the positive password space. As in the design of NAS, obfuscation is added for the attackers by not covering the complete negative space with the detectors. Hence, complementing the compromised negative space with the possible total password space (this process will take a significant amount of time depends on the dimension of the password space and all the configuration parameters are stored in the second layer) will not disclose the actual password space. To reveal the actual password space, the attackers has to try brute force approach to check which areas of the complemented space is the actual positive space. Again, the randomness of the mapping of all self-points in the password space is uniformly distributed (as one of the properties of one-way hash function). This design approach makes the process harder for the attackers to pinpoint the exact location of the password point to reveal the actual password for a particular user.

In the case of Honeyword based system, the first layer contains the set of Honeywords and the actual password for all the users. If the attackers compromise that layer, they will actually get the real password along with some Honeywords and they will know the length of the passwords which provide a vital information to reduce their time to try brute-force attack.

Table 9: A comparison among the NAS approaches and the PITHYA system.G-NAS has been found to be performing the best (in fact, slightly better than the PYTHIA system). However, the result can be data specific and may be different for other data sets.

	NAS Approaches			PYTHIA
	G-NAS	R-NAS	B-NAS	
Accuracy (approx.)	98%	95%	95%	97%
False positive	0.12	0.21	0.19	0.19
False Negative	0.12	0.36	0.39	0.23
Detection time (approx.)	5.27 ms (avg.) with standard deviation: 1.35 ms	7.85 ms (avg.) with standard deviation: 2.63 ms	6.3 ms (avg.) with standard deviation: 2.85 ms	6.17 ms (avg.) with standard deviation: 1.81 ms

8.2.2 Guessing Attacks

Guessing attack is the most common attack in the password based systems. For NAS approach, the password cracking test is designed to check how well it is performed against the guessing attacks, which is shown in earlier section. For Honeyword systems, if two users have the same password, this is very likely that the Honeywords of them are also very similar according to the design of Honeywords. Hence, revealing the Honeywords for all the users provide more useful knowledge to the attackers to apply guessing attacks. On the other hand, for NAS, for the same password for two different users, the randomness of hashed output is uniformly distributed in the representative password space which make the job harder for the attacker to detect the same password.

9 Conclusion

The negative authentication approach can detect and filter out most of the invalid requests, and hence lower the probability of making guessing requests to access the positive authentication data. In case of comprise of first layer (negative detector region), an attacker can have access of file containing anti-password information. From there, he can only know about the region where valid hash exists; however, he has no idea, which of them are valid and how many of them are valid. Hence, exposing the negative detector upfront reduces the overall password cracking risks. Also, the password file that contains the authentication data is kept in another secure server and is not exposed to the outside of NAS, which assures the security of the password file (positive information). More rigorous security analysis of these three approaches will be done in future.

Three implementations of NAS described in this paper explore the details of negative information based authentication system through the context of current cyber-attacks. All these models of NAS reduce the false positive rate of authentication requests (deterministic method performs best among the three) and increase the detection rate of invalid requests. The approached design, integration, and deployment of these three models clearly show the applicability of these new ways of authentication system with all types of existing password based authentication systems. The results demonstrated that combining the negative approach with the existing authentication system provides significant benefits to overcome certain password based attacks.

Acknowledgements This work was supported by IARPA Seedling program and Cooperative Agreement (Number N66001-12-C-2003) administered by the ONR SPAWAR Systems Center. Points of view and opinions on this paper are those of the author(s) and do not necessarily represent the position or policies of the United States. The authors are very thankful to the reviewers for their valuable feedback and thoughtful suggestions to improve the quality of the manuscript.

References

1. Bojinov, H., Bursztein, E., Boyen, X., Boneh, D.: Kamouflage: Loss-resistant password management. In: Computer Security–ESORICS 2010, pp. 286–302. Springer (2010). URL `http://crypto.stanford.edu/~dabo/papers/passwordmgr.pdf`. Accessed: 2017-01-24.
2. Bond, M.: Comments on gridsure authentication. Online at http://www. cl. cam. ac. uk/~ mkb23/research/GridsureComments. pdf (2008). URL `https://www.cl.cam.ac.uk/~mkb23/research/GridsureComments.pdf`. Accessed: 2017-01-24.
3. Bonneau, J.: Guessing human-chosen secrets. Ph.D. thesis, University of Cambridge (2012). URL `https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-819.pdf`. Accessed: 2017-01-24.
4. Brants, T., Franz, A.: The google web 1t 5-gram corpus version 1.1. LDC2006T13 (2006). URL `https://catalog.ldc.upenn.edu/ldc2006t13`. Accessed: 2017-01-24.

5. Butler, R.: List of the 1000 most common surnames in the u.s. (2009). URL http://names.mongabay.com/most_common_surnames.htm. Accessed: 2017-01-24.
6. Camenisch, J., Lehmann, A., Neven, G.: Optimal distributed password verification. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 182–194. ACM (2015). URL http://dl.acm.org/citation.cfm?id=2813722. Accessed: 2017-01-24.
7. Cubrilovic, N.: Rockyou hack: From bad to worse, dec 2009 (2009). URL https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/. Accessed: 2017-01-24.
8. Dasgupta, D., Azeem, R.: An investigation of negative authentication systems. In: Proceedings of the 3rd International Conference on Information Warfare and Security, pp. 117–126 (2008). URL http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=14EA96BC1BB9B1B9B8EFB47ECE961758?doi=10.1.1.372.1491. Accessed: 2017-01-24.
9. Dasgupta, D., Ferebee, D., Saha, S., Nag, A.K., Madero, A., Sanchez, A., William, J., Subedi, K.P.: G-nas: A grid-based approach for negative authentication. Symposium on Computational Intelligence in Cyber Security (CICS) at IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, Orlando, Florida (2014). URL http://ieeexplore.ieee.org/document/7013362/. Accessed: 2017-01-24.
10. Dasgupta, D., Forrest, S.: An anomaly detection algorithm inspired by the immune system. In: Artificial immune systems and their applications, pp. 262–277. Springer (1999). Accessed: 2017-01-24.
11. Dasgupta, D., Ji, Z., Gonzalez, F.: Artificial immune system (ais) research in the last five years. In: Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, vol. 1, pp. 123–130 Vol.1 (2003). URL http://ieeexplore.ieee.org/document/1299565/. Accessed: 2017-01-24.
12. Dasgupta, D., Saha, S.: Password security through negative filtering. In: Emerging Security Technologies (EST), 2010 International Conference on, pp. 83–89. IEEE (2010). URL http://dl.acm.org/citation.cfm?id=1902111. Accessed: 2017-01-24.
13. De Castro, L.N., Timmis, J.: Artificial immune systems: a new computational intelligence approach. Springer (2002). URL http://www.springer.com/us/book/9781852335946. Accessed: 2017-01-24.
14. Esponda, F., Ackley, E.S., Helman, P., Jia, H., Forrest, S.: Protecting data privacy through hard-to-reverse negative databases. In: Information Security, pp. 72–84. Springer (2006). URL https://crypto.stanford.edu/portia/papers/HardNDB.pdf. Accessed: 2017-01-24.
15. Everspaugh, A., Chaterjee, R., Scott, S., Juels, A., Ristenpart, T.: The pythia prf service. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 547–562 (2015). URL https://www.usenix.org/node/190917. Accessed: 2017-01-24.
16. Feldmeier, D.C., Karn, P.R.: Unix password security-ten years later. In: Advances in CryptologyCRYPTO89 Proceedings, pp. 44–63. Springer (1990). URL http://www.cs.technion.ac.il/~cs236350/Material/unix-password-security-ten.pdf. Accessed: 2017-01-24.
17. Flp, ., Kovcs, L., Kurics T.and Windhager-Pokol, E.: Balabit mouse dynamics challenge data set (2016). URL https://github.com/balabit/Mouse-Dynamics-Challenge
18. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: IEEE Computer Society Symposium on Research in Security and Privacy, pp. 202–202. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (1994). URL http://dl.acm.org/citation.cfm?id=884218. Accessed: 2017-01-24.
19. Fossi, M., Johnson, E., Turner, D., Mack, T., Blackbird, J., McKinney, D., Low, M.K., Adams, T., Laucht, M.P., Gough, J.: Symantec report on the underground economy. Symantec Corporation (2008). URL http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf. Accessed: 2017-01-24.
20. Gamboa, H., Fred, A.: A behavioral biometric system based on human-computer interaction. In: Proceedings of SPIE, vol. 5404, pp. 381–392 (2004)
21. Gong, L.: Collisionful keyed hash functions with selectable collisions. Information Processing Letters **55**(3), 167–170 (1995). URL http://www.sciencedirect.com/science/article/pii/002001909500085Q. Accessed: 2017-01-24.
22. Hofmeyr, S.A., Forrest, S.: Architecture for an artificial immune system. Evolutionary computation **8**(4), 443–473 (2000). URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.486.3902&rep=rep1&type=pdf. Accessed: 2017-01-24.

23. Ji, Z.: Negative selection algorithms: From the thymus to v-detector. Ph.D. thesis (2006). URL http://dl.acm.org/citation.cfm?id=1237333. AAI3230960
24. Ji, Z., Dasgupta, D.: V-detector: An efficient negative selection algorithm with probably adequate detector coverage. Inf. Sci. **179**(10), 1390–1406 (2009). DOI 10.1016/j.ins.2008.12.015. URL http://www.sciencedirect.com/science/article/pii/S0020025508005434. Accessed: 2017-01-24.
25. Juels, A., Rivest, R.L.: Honeywords: Making password-cracking detectable. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 145–160. ACM (2013). URL https://people.csail.mit.edu/rivest/pubs/JR13.pdf. Accessed: 2017-01-24.
26. Kanerva, P.: Sparse distributed memory. MIT press (1988). URL https://mitpress.mit.edu/books/sparse-distributed-memory. Accessed: 2017-01-24.
27. Khalil, G.: Password securitythirty-five years late. URL https://www.sans.org/reading-room/whitepapers/basics/password-security-thirty-five-years-35592 (2014). URL https://www.sans.org/reading-room/whitepapers/basics/password-security-thirty-five-years-35592. Accessed: 2017-01-24.
28. Metropolis, N., Ulam, S.: The monte carlo method. Journal of the American statistical association **44**(247), 335–341 (1949). URL http://homepages.rpi.edu/~angel/MULTISCALE/metropolis_Ulam_1949.pdf. Accessed: 2017-01-24.
29. Pagh, R., Rodler, F.F.: Cuckoo hashing. Journal of Algorithms **51**(2), 122–144 (2004). URL http://www.it-c.dk/people/pagh/papers/cuckoo-jour.pdf. Accessed: 2017-01-24.
30. Perlroth, N.: Hackers in china attacked the times for last 4 months. NY Times, Jan **30** (2013). URL http://www.nytimes.com/2013/01/31/technology/chinese-hackers-infiltrate-new-york-times-computers.html. Accessed: 2017-01-24.
31. Schechter, S., Herley, C., Mitzenmacher, M.: Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In: Proceedings of the 5th USENIX conference on Hot topics in security, pp. 1–8. USENIX Association (2010). URL https://www.microsoft.com/en-us/research/publication/popularity-is-everything-a-new-approach-to-protecting-passwords-from-statistical-guessing-attacks/. Accessed: 2017-01-24.
32. SkulSecurity: Password-skullsecurity (2011). URL https://wiki.skullsecurity.org/Passwords. Accessed: 2017-01-24.
33. Smith, R.E.: Authentication: from passwords to public keys. Addison-Wesley Longman Publishing Co., Inc. (2001). URL http://dl.acm.org/citation.cfm?id=501593. Accessed: 2017-01-24.
34. Song, H., Dharmapurikar, S., Turner, J., Lockwood, J.: Fast hash table lookup using extended bloom filter: an aid to network processing. ACM SIGCOMM Computer Communication Review **35**(4), 181–192 (2005). URL http://conferences.sigcomm.org/sigcomm/2005/paper-SonDha.pdf. Accessed: 2017-01-24.
35. Zheng, Y., Matsumoto, T., Imai, H.: Structural properties of one-way hash functions. In: Advances in Cryptology-CRYPT090, pp. 285–302. Springer (1991). URL https://pdfs.semanticscholar.org/ed78/92387cd971e26241eb34f779a01807cb143c.pdf. Accessed: 2017-01-24.

Input:
e_{old} : old entry to be updated
e_{new} : updated new entry
d : set of detectors
s : set of self-points
n : no of update operation after NAS is ran last time
p : predefined reset percentage
Output:
d: set of detectors

```
remove e_old from s
if ( n > p * s)
   s = s ∪ {e_new}              //add e_new to s
   d ← ϕ                        // empty set of detectors
   d = NAS(s)                   // run NAS
else
   for all detector in d
      if e_new is in detector
         d = d – {e_new}        // remove detector from d
   s = s ∪ {e_new}              //add e_new to s
   n = n+1                      // increase n by 1
   d = NAS(s, d)                //run NAS with existing s and d
return d
```

Fig. 30: **Pseudo-code for updating of existing self-points in R-NAS and B-NAS**

***Input**:*
S: set of self-points in binary vector format
d: *Integer* – dimension of the self-point vector
N: *Integer* – dimension of the Grid
***Output**:*
A two-dimensional array representing Grid space
$grid \leftarrow \Phi$
$len_{half} = d/2$;

for i ← 1to *N*;
  for j ← 1to *N*;
    $grid[i][j] \leftarrow 0$;

Repeat for every S_i in *S*
  $X \leftarrow S_i[1 : len_{half}]$;
  $Y \leftarrow S_i[len_{half} + 1 : d]$;
  row ← convertToInteger(*X*);
  col ← convertToInteger(*Y*);
  $grid[row][col] \leftarrow grid[row][col] + 1$;
end Repeat

return *grid*;

Fig. 31: **Pseudo-code for detector generation algorithm in NAS using mod based G-NAS model.The 'convertToInteger'function takes a string input consisting only numbers and produce the integer value of the string.**

Input:
S: set of self-points in binary vector format
d: *Integer* – dimension of the self-point vector
N: *Integer* – dimension of the Grid
k: *Integer* – square root of dimension of the Grid (Grid dimension: 2^k)
Output:
A two dimensional array representing Grid space
$grid \leftarrow \Phi$
$len_{half} = d/2$;
for i ← 1to *N*;
  for j ← 1to *N*;
    $grid[i][j] \leftarrow 0$;
Repeat for every S_i in *S*
  $X1 \leftarrow S_i[1 : len_{half}/2]$;
  $X2 \leftarrow S_i[len_{half}/2 + 1 : len_{half}]$;
  $Y1 \leftarrow S_i[len_{half} + 1 : 3 * len_{half}/2]$;
  $Y2 \leftarrow S_i[3 * len_{half}/2 + 1 : d]$;
  S1 ← substring(*X1*, *k*/2);
  S2 ← substring(*X2*, *k*/2);
  S3 ← substring(*Y1*, *k*/2);
  S4 ← substring(*Y2*, *k*/2);
  row ← convertToInteger(concate(*S1*, *S2*));
  col ← convertToInteger(concate(*S3*, *S4*));
  $grid[row][col] \leftarrow grid[row][col] + 1$;
end Repeat
return *grid*;

Fig. 32: **Pseudo-code for detector generation algorithm in NAS using two layer based G-NAS model. The 'convertToInteger'function takes a string input consisting only numbers and produce the integer value of the string. The 'concate'function concatenates two strings into one single string.**

Input:
S: set of self-points in binary vector format
n: *Integer* – dimension of the self-point vector
r: *Integer* – compression ratio
N: *Integer* – dimension of the Grid
k: *Integer* – square root of dimension of the Grid (Grid dimension: 2^k)
Output:
A two dimensional array representing Grid space
grid ← *Φ*
$len_{half} = d/2$;
for i ← 1to *k*
  pos (*i*) ← random(1, len_{half});
for i ← 1to *N*;
  for j ← 1to *N*;
    grid[*i*][*j*] ← 0;
Repeat for every S_i in *S*
  X ← S_i [1 : len_{half}];
  Y ← S_i [len_{half} + 1 : *d*];
  j ← 1;
  for i ← 1 to len_{half}
    $X_{cmprsd}(j)$ ← XOR(*X*(*i*), *X*(*i* + 1), ..., *X*(*i* + *r* -));
    $Y_{cmprsd}(j)$ ← XOR(*Y*(*i*), *Y*(*i* + 1), ..., *Y*(*i* + *r* -));
    j ← *j* + 1;
    i ← *i* + r;
  for i ← 1to *k*
    S1(*i*) ← X_{cmprsd}(*pos*(*i*));
    S2(*i*) ← Y_{cmprsd}(*pos*(*i*));
  row ← convertToInteger(*S1*);
  col ← convertToInteger(*S2*);
  grid[*row*][*col*] ← *grid*[*row*][*col*] + 1;
end Repeat
return *grid*;

Fig. 33: **Pseudo-code for detector generation algorithm in negative detection using XOR based G-NAS. The 'convertToInteger'function takes a string input consisting only numbers and produce the integer value of the string. The 'concate'function concatenates two strings into one single string.**

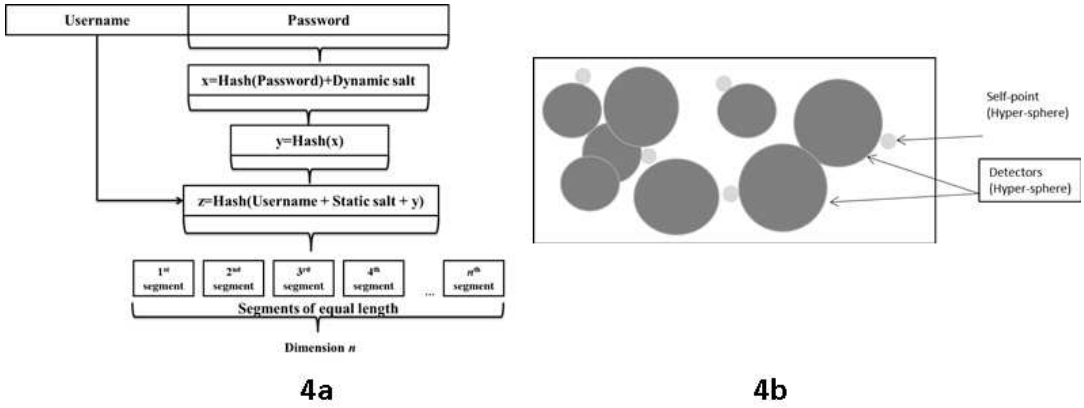

Fig. 4: **The dimension concept of the real space based implementation in figure 4a and two dimensional projection of self-regions and detectors in Real Space in figure 4b.**

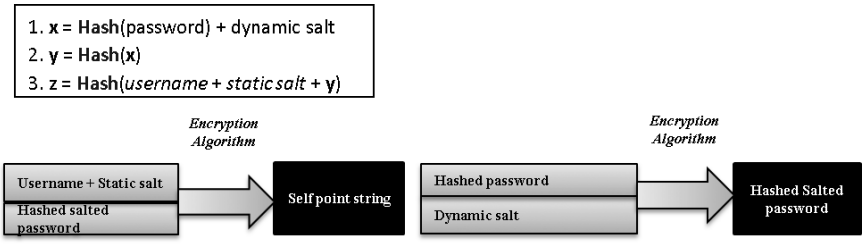

Fig. 8: **Creation of hash string from the username-password pair. '+' sign indicates concatenation operation among strings and 'z' represents the derived hashed string. For hashing SHA-128, SHA-256, SHA-512 can be used.**

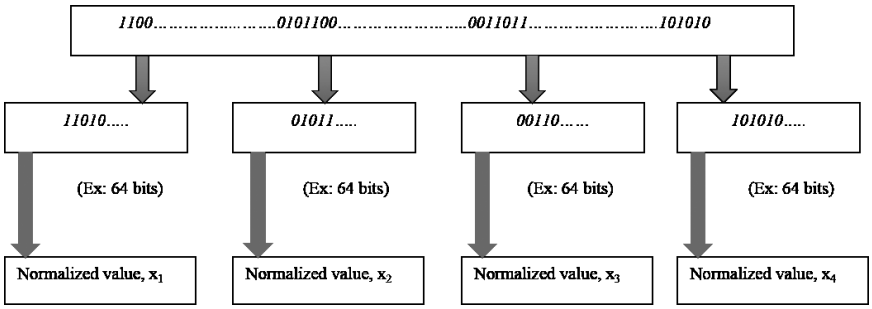

Fig. 9: **Formulation of a self-point in four dimensional real space from a 256 bit hashed self-point string.**

```
V_Real_Space_Detector - Set()
Input:
S: Set of self points
n: Dimension of the space
r_s: Confusion parameter
F: Maxmimum Failed Attempt
Output:
A set of V - detectors
1: D ← ∅
2: d ← 0
3: f ← 0
4: Repeat
5: if f = F, exit
6:    r ← dimention of the space
7:    x ← random sample point in Real Space with n dimension
8:    Repeat for every d_i in D = {d_1, d_2, d_3, ...}
9:      d_e ← Euclidian distance between x(d_i) and x, where x(d_i) is the location of d_i
10:       if d_e ≤ r(d_i) then, where r(d_i) is the radius of detector d_i
11:       f ← f + 1
12:           go to 6:
13:   Repeat for every s_i in S
14:       d ← Euclidian distance between s_i and x
15:       if d - r_s < r then r ← d - r_s
16:       if r <= 0 then
17:           f ← f + 1
18:           go to 6:
19:   if r > 0 then
20:    Repeat for every d_i in D = {d_1, d_2, d_3, ...}
21:       d_diff ← |d_i − r|
22:     d_center ← distance between center point of d_i and x
23:     if d_center ≤ d_diff
24:       f ← f + 1
25:       go to 6:
26:       overlap ← |r − d_diff|
27:       if (overlap > 0.5 * min(r_{d_i}, r))
28:           r ← r − (overlap − 0.5 * min(r_{d_i}, r))
29.  f ← 0
30: D ← D ∪ {< x, r >}, where < x, r > is a detector with location x and radius r
31:   if d >= c,  exit
32: return D
```

Fig. 26: **Pseudo-code for detector generation algorithm in NAS using real space model.**

```
V-Binary Detector-Set()
Input :
S: set of self samples
n: Dimension of the space
r_s: Confusion parameter
c_0: Minimum Coverage
F_0: Maxmimum Failed Attempt
Output :
A set of V-detectors
1: D ← ∅
2: d_0 ← 0
3.: f_0 ← 0
4: Repeat
5. if f_0 = F_0, and D.size > S.size, exit
6:    r ← n
7:    x ← random sample point with n dimension in Binary value in each dimension
8:    Repeat for every d_i in D = {d_i . i = 1, 2, ...}
9:      d_d ← Hamming distance between x(d_i) and x, where x(d_i) is the location of d_i
10:       if d_d ≤ r(d_i) then, where r(d_i) is the radius of detector d_i
11:       f_0 ← f_0 + 1
12:           go to 6:
13:   Repeat for every s_i in S
14:       d ← Haming distance between s_i and x
15:       if d - r_s − 1 < r then r ← d - r_s − 1
16:       if r <= 0 then
17:           f_0 ← f_0 + 1
18:           go to 6:
19:   if r > 0 then
20:   Repeat for every d_i in D = {d_i . i = 1, 2, ...}
21        d_diff ← |d_i − r|
22:       dsum ← d_i + r
23:     dcenter ← distance between centerpoint of d_i and x
24:     if  dcenter ≤ d_diff or  dcenter ≤ dsum/2
25:       f_0 ← f_0 + 1
26:       go to 6:
27: f_0 ← 0
28: D ← D ∪ {< x, r >}, where < x, r > is a detector with location x and radius r
29:   Calculate detector coverage and add it to d_0.
30:   if d_0 >= c_0,  exit
31:  return D
```

Fig. 27: **Pseudo-code for detector generation algorithm in NAS using binary space model.**

```
Input:
e : entry to be deleted
d : existing set of detectors
n : no of self-points from last time NAS is run
s : set of existing self-points
p : predefined reset percentage
Output
1. d ← set of detectors
2. s = s – {e}          // remove e from s
3. if |s| < p * n
4.     then d = NAS(s) // run NAS with empty detector set and the self-points 's'
5. return d
```

Fig. 28: **Pseudo-code for deletion of existing self-points from the set of self-points in R-NAS and B-NAS**

```
Input
e : entry to be added
d : set of detectors
s : set of self-points
p : predefined reset percentage
n0 : no of self-points when NAS (not using existing entities) is ran last time
n : no of self-points added after NAS (not using existing entities) is ran last time
Output
d: set of detectors

if n > p * n0
   d ← ϕ
   s = s U {e}          // add e to s
   d = NAS(s)  // run NAS with s
else
   for all detector in d
      if e is in the detector set
         d = d – {e} //remove detector from d
   s = s U {e}              // add e to s
   n = n + 1                //increase n by 1
   d = NAS(s, d)            // run NAS with existing s and d
return d
```

Fig. 29: **Pseudo-code for adding of existing self-points to the set of self-points in R-NAS and B-NAS**