

Fig. 1A

100

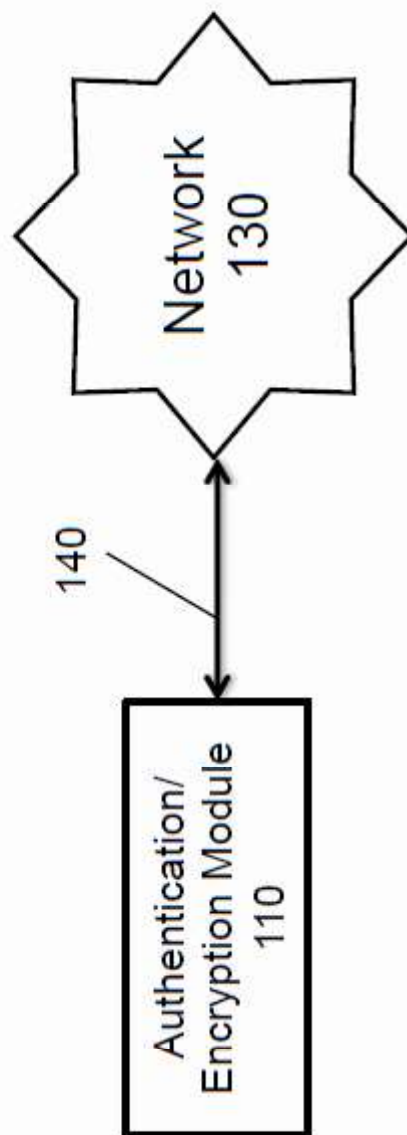


Fig. 1B

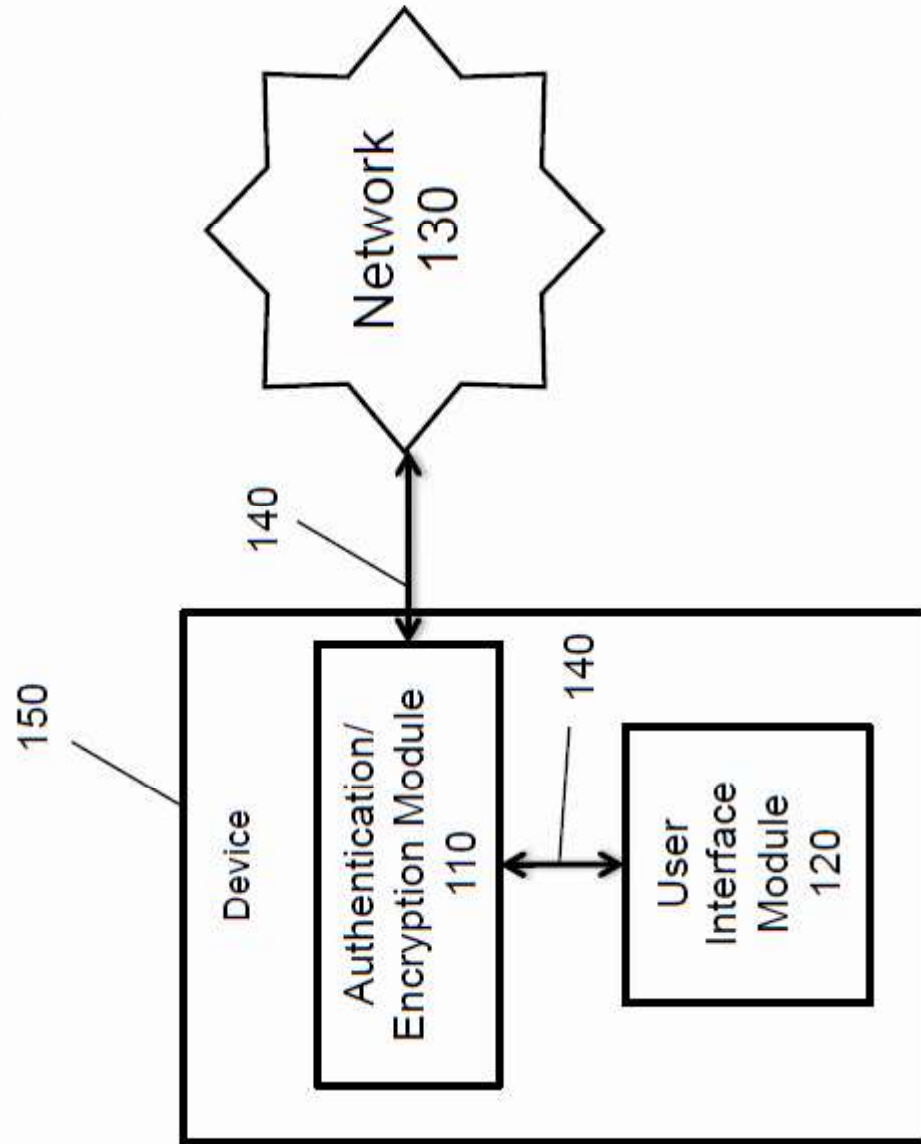


Fig. 1C

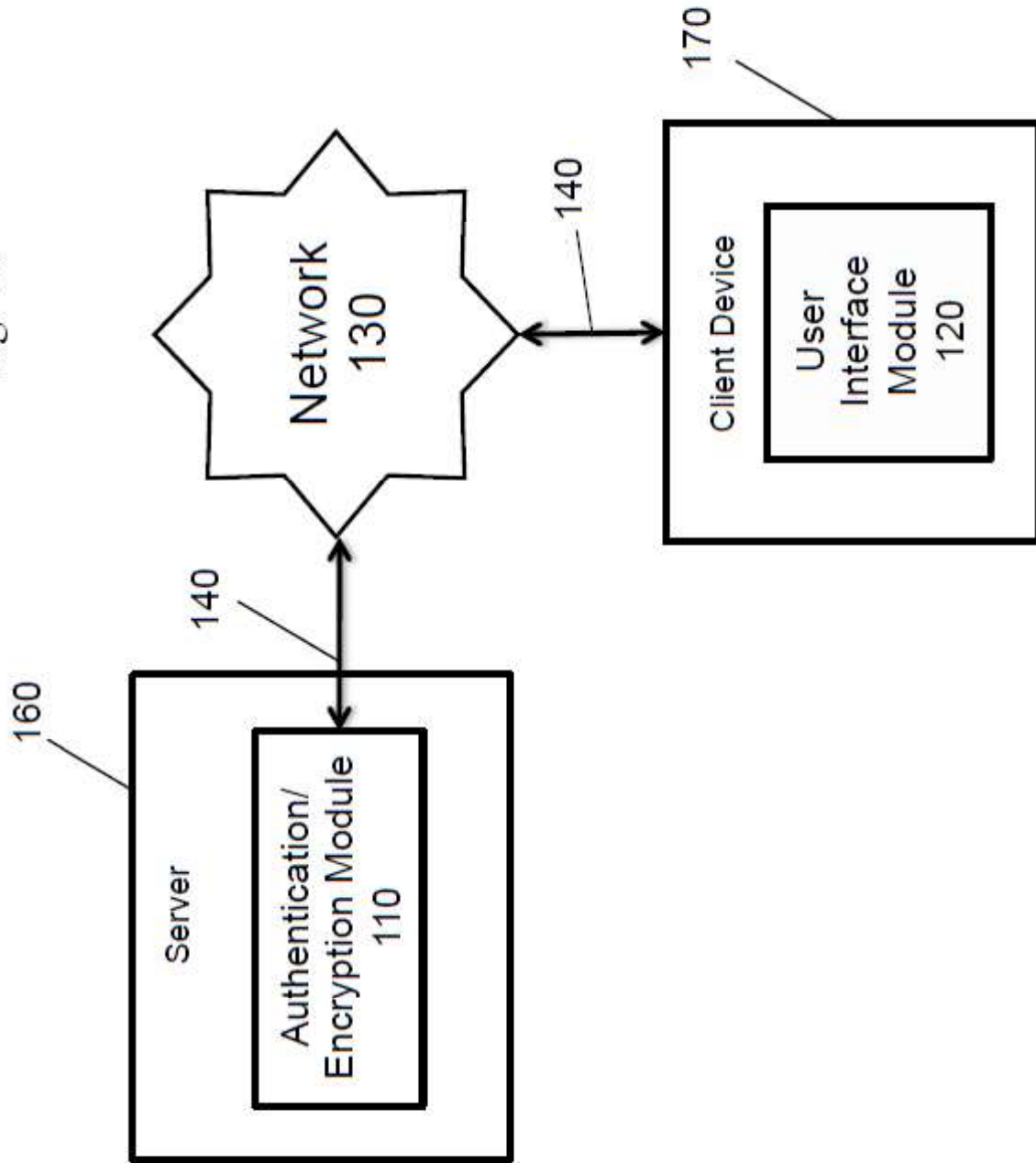


Fig. 1D

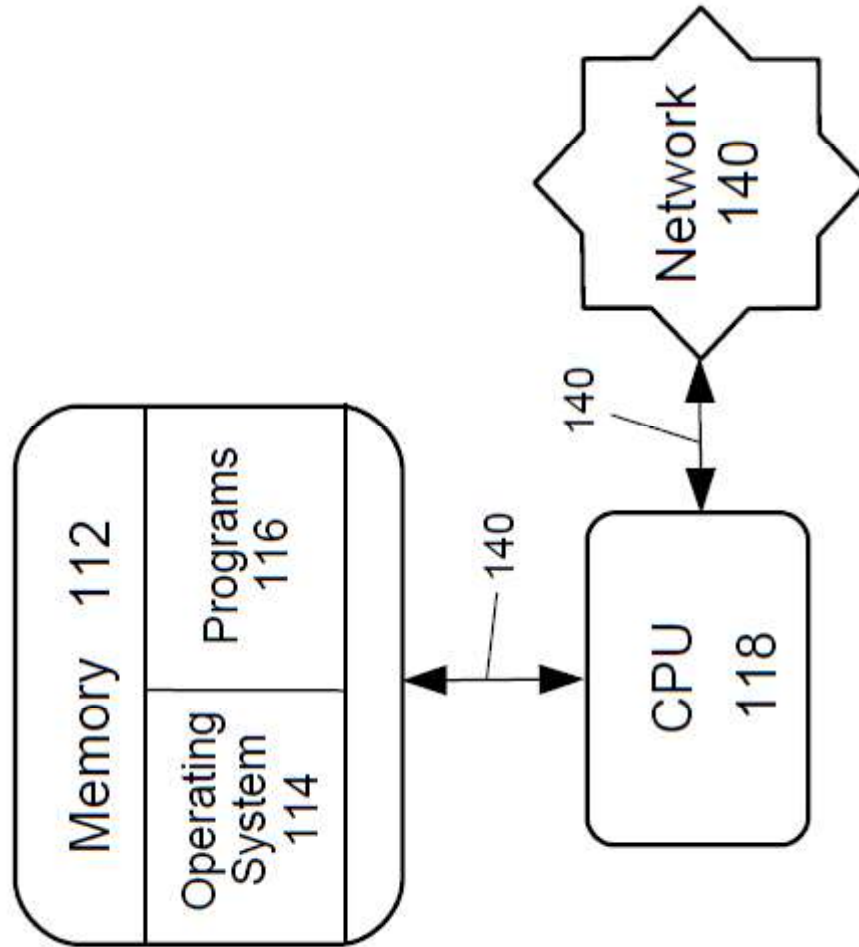


Fig. 2A

(GATE_4)

1st Dimension

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

2nd Dimension

5/43

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ~ ! @ # % ^ & * ; ?

3rd Dimension[illegible]

4th Dimension

+ × ÷ © ® ✂ ☒ 🌀 :: ♪ 🕊 ☯ ☆ ✖ ⚗ † ☸ ☐ ☢ ∑ ∞ π λ ψ ∫ \$ % & ' () * + , - . : ;

Fig. 2B

(GATE_5)

1st Dimension

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

2nd Dimension

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω 4

3rd Dimension

[illegible]

4th Dimension


















































5th Dimension

+ - × ÷ → ← ↗ ✂ ☒ ☹ ☺ ☯ ☼ ♂ ♀ ☐ ✖ ⚔

Fig. 3

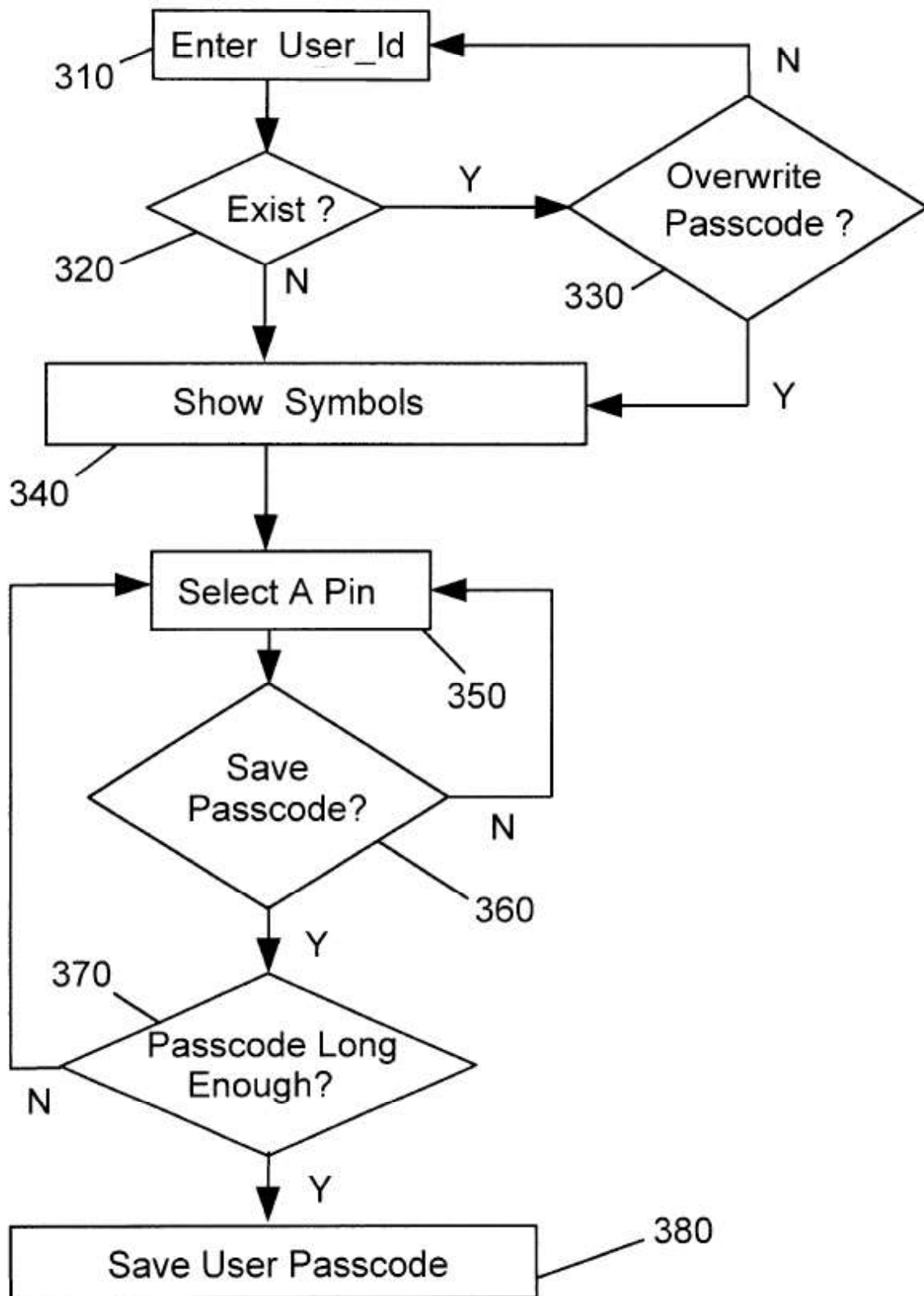


Fig. 4

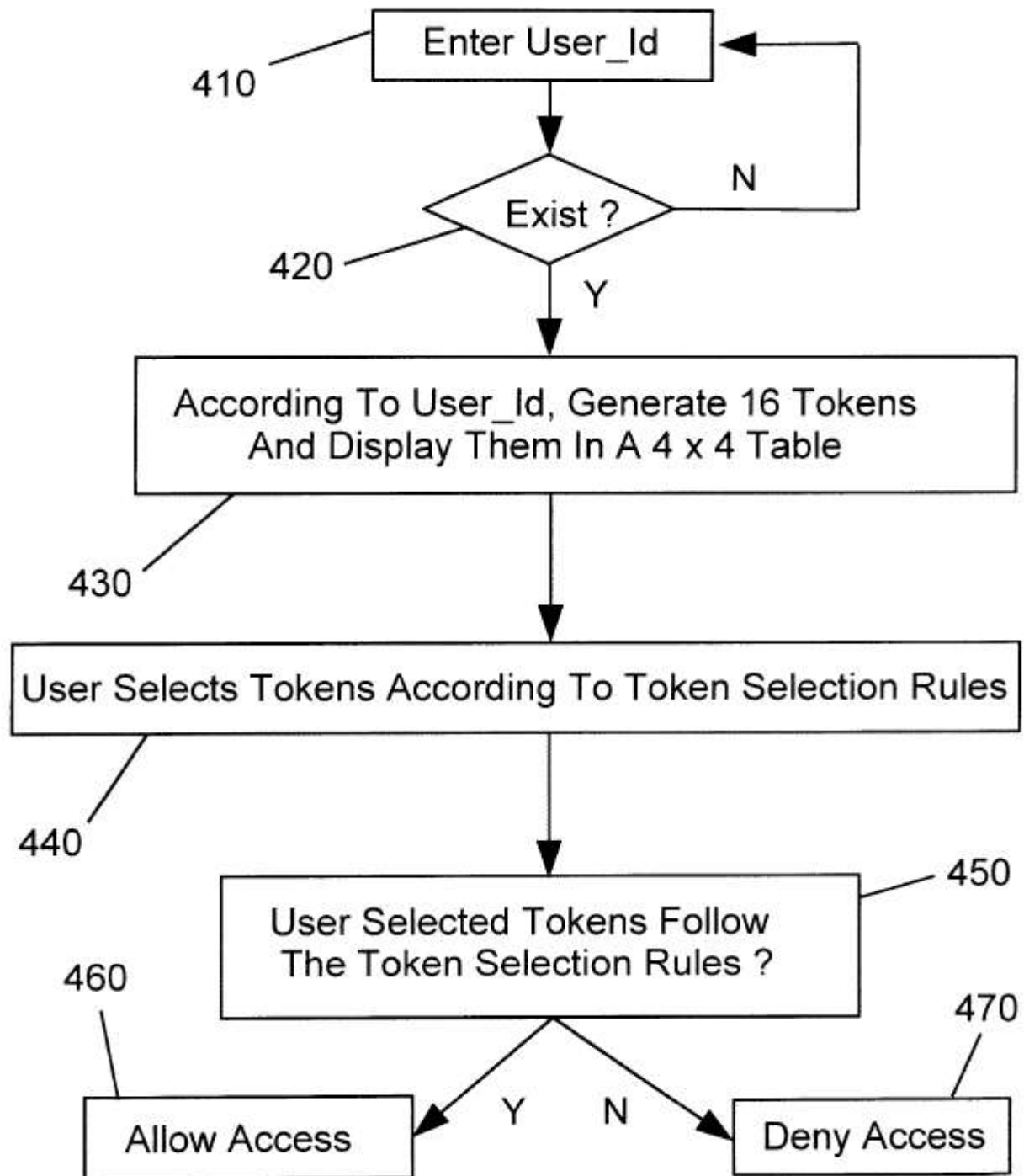


Fig. 5

(Token Generation Rules)

[For illustration, GATE_4 embodiment is used (see Fig. 2A)]

The goal is to generate 16 tokens with no redundant symbols in them, each token will have 4 symbols, one from each dimension. e.g. : Token_1(2 ② ♥ +), Token_2(30 ? ♠ λ), ... , Token_16(16 @ ✓ ☼)
At least one user pin* should be in one of the 16 tokens, possibly more, even all of them.

Notice the orders of symbols above, 1st one is always from 1st_Dim, 2nd one is from the 2nd_Dim ...

Rules And Steps

Example Results

- | | |
|--|---|
| [1] Create 4 Vectors each contains 36 symbols from one of the four dimensions described in FIG 2 : Available_Symbols for GATE_4.
Each time a symbol is removed from a Vector, its size will reduce by one.
The reason to remove symbols from the vectors is to avoid duplication.
I'll refer to the above vectors as dimensional vectors : V_1, V_2, V_3, V_4.
Each token generation step will remove one remaining symbol from each of the 4 above vectors, so that no two tokens will have the same symbols. | [1] V_1(1, 2, ... 35, 36)
V_2(①, ②, ... ; ?)
V_3(○, ●, ... °C, °F)
V_4(+, -, ... F, ∞) |
| [2] Get User_Id entered by the user during login process | [2] admin |
| [3] Get user passcode** from User_Id saved in memory during registration | [3] ① ♥ 2 ∞ |
| [4] Save a random pin from user passcode into : User_Pin_Vector*** | [4] ♥ |
| [5] Save a random number [from 1 to 16] to User_Pin_Show_Up_Location | [5] 7 |
| [6] for (i=1 ; i<=16 ; i++) do step [7] and [8] to generate 16 tokens | [6] |
| [7] Create an empty token : a vector to hold 4 symbols | [7] Token_i() |
| [8] for (j=1 ; j<=4 ; j++) do step [8.1] or [8.2] to add one symbol from each dimension to the token. Step [8.1] makes sure at least 1 user pin is used. | [8] Add 4 symbols to it. |
| [8.1] if i equals User_Pin_Show_Up_Location and ♥ from User_Pin_Vector is still in V_j, remove it from V_j and add it to current token. | [8.1] Token_7(15 ~ ♥ ☼) |
| [8.2] else remove one random symbol from V_j to add to the current token. Size of V_j will reduce by one after removing that symbol. | [8.2] Token_i(8 ② ★ ⊕) |
| [9] After the above steps, we will have 16 tokens with at least one user pin. | [9] Token_1, ..., Token_16 |

Note : * pin – each symbol in a passcode** is a pin. For instance ① is the 1st pin and ∞ is the 4th pin in the following passcode : ① ♥ 2 ∞

** passcode – similar to common sense password, but can contain symbols from the selections in each dimension [e.g. ① ♥ 2 ∞], the dimensions are described in FIGS. 2A and 2B.

*** User_Pin_Vector – a Vector in computer language Java which can contain any number of elements of any type, in this case a symbol.

Fig. 6

(Token Selection Rules)

[1] User will be shown 16 tokens in a 4 x 4 table, each token will have 4 symbols in it, for example, a token(2 ② ♥ ÷) will show up like this :

2	②
♥	÷

Each dimension has a fixed position in a token, upper left is for 1st dimension symbols, upper right is for 2nd dimension symbols, lower left is for 3rd dimension symbols, lower right is for 4th dimension symbols. Fixed position will help user quickly locate symbols in a token.

1 st D	2 nd D
3 rd D	4 th D

[2] User must follow the order of user passcode, e.g. ① ♥ 2 ☒, to select the tokens that contain user pins. For example, the table on the right has 16 tokens, at least one of them contains a user pin, user can follow the below steps to select tokens.

For easier reference, I'll name The rows : A, B, C, D, and columns : 1, 2, 3, 4

[3] Since user's 1st pin is ① which belongs to 2nd dimension, he can search for it in the upper right corner of each token, because it's not in any of the 16, he can and should pick any token in its place.

So I can pick A2 : (34 ② ♦ ♪).

Same with 2nd user pin : ♥,

because it's missing, I can pick

D3 : (16 ③ ● ☒). 3rd user pin 2 is also missing, so I'll pick B1 : (3 ④ ← ⑤). But the 4th user pin ☒ is in token A1, user must pick it to be valid, so I'll pick A1 : (13 & ② ☒).

So in the end, user picked the following 4 tokens : (34 ② ♦ ♪), (16 ③ ● ☒), (3 ④ ← ⑤) and (13 & ② ☒), these 4 tokens will be sent to the server for validation.

For GATE_5 : The process is the same as GATE_4, just add one more dimension. A sample token and the locations of the 5 dimensions are shown on the right. Now a token might be like this : (① σ 17 ● £).

①	σ
17	●
●	£

Each table has 16 tokens, but each token contains 5 symbols – one from each of the 5 dimensions from FIG. 2B.

1 st D	2 nd D
	3 rd D
4 th D	5 th D

Fig. 7

(Token Validation Rules)

For GATE_4 : continue with the example from FIG. 6.

[1] User must follow the order of user passcode ① ♥ 2 ∞ to select the tokens that contain user pins.

[2] The original 16 tokens are shown in the table on the right.

[3] The 4 tokens user selected were :
(34 M ♦ ♪), (16 B • ♡),
(3 Q ← ®) and (13 & ☺ ∞).

[4] The validation process will check them one by one, if one of them fails, user login request will be denied.

[5] If user selected more or less tokens than the pin count in his passcode, login request will also be denied.

[6] In my sample implementation user passcodes can be up to 6 pins long.

[7] Since user's 1st pin is ①, we will go through all the symbols in all the 16 tokens and see if the symbol ① exists. Since it didn't exist, user can and must pick a wild-card token in its place, and in our example, user's 1st selected token (34 M ♦ ♪) is valid.

[8] User's 2nd pin ♥ is also missing in all the 16 tokens, so the 2nd token that user picked : (16 B • ♡) is also valid, same with the 3rd token (3 Q ← ®) user picked.

[9] User's 4th pin was the symbol ∞, and when we go through all the symbols in the above 16 tokens, we can see it's in the A1 token, so in order to be valid, user must and can only pick A1, and in our example, he did pick (13 & ☺ ∞) as his 4th and last token, no more and no less than the pin count (4) in his original passcode, therefore, user login request is granted.

	1	2	3	4
A	13 & ☺ ☒	34 M ♦ ♪	29 P → ☢	28 ! ✓ ÷
	3 Q ← ®	36 V = ☯	6 : ↓ π	19 W X +
C	20 Z ☾ ☞	26 J °F €	17 S ♠ X	12 C ♥ ∞
	32 K ♠ \$	35 F ☆ ☼	16 B ● ☞	10 U °C ☎

For GATE_5 : The process is the same, just add one more dimension.

Fig. 8B GATE_4 Create_Id

☒ Demo
 GATE_4
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5

Create User Id & Pn
 Graphic Login
 Text Login

User Id admin G4 206
 G4 208
 Check Availability

G4 210
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

G4 220
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

G4 230
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

G4 240
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

Pick at least 4 pins from any of the above choices. There are 4 categories : [1] Numbers, [2] Characters, [3] Signs, [4] Other symbols.
 You can choose from the same category or different categories. And you can choose the same pin multiple times : ☹️♥️♫
 These pins will be your passcode when you login, pick something that is easy for you to remember and yet hard for anyone else to guess.

Selected Pins
 1
 2
 Save
 G4 270

Current

☐ Demo

Fig. 9B GATE_4_Graphic_Login

☒ Demo
 GATE_4
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5
 Create User Id & Pin
 Graphic Login
 Test Login

User Id admin G4 306
 Enter

Login Result
 Login Successful
 OK
 G4 380

[A] The table in the center has 4 x 4 = 16 token buttons. Each token button has 4 symbols. They are arranged into 4 groups :

[1] Numbers
[2] Characters
[3] Signs
[4] Other symbols

Mouse over the sample token above to see all the 36 items in each group.

The positions of the above groups are fixed in any token, so numbers will always show up at the upper left corner, characters will always show up at the upper right corner of a token, etc. This may help you quickly locate and enter your pin from the token table in the center →

e.g. If your passcode is : ① ♥ 2 ☒
Then the following tokens containing your pins and are, therefore, valid.

*	①	*	*	*	*	*	*	*	☒
*	*	*	*	*	*	*	*	*	*

24	Ⓔ	17	Ⓥ	25	Ⓐ	3	ⓧ
∴	♠	★	⊕	□	Ⓢ	←	Ⓜ
7	~	16	Ⓜ	1	Ⓚ	21	Ⓢ
♣	♠	≠	♣	×	☒	€	€
18	Ⓜ	31	Ⓜ	34	Ⓢ	6	Ⓢ
→	\$	♀	∞	=	♠	♠	†
15	Ⓢ	23	Ⓢ	27	?	9	Ⓢ
☒	-	☆	♠	♥	♠	✓	※

G4 320

[B] The rules are as follows :

[1] Enter your User_Id.

[2] Hit the [Enter] key or button.

[3] Follow the order of your pins.

[4] Pick the tokens that contain your pins from the token table ← in the center.

[5] If your pin is not on any token in the table, you can, and must, select any token for that pin.

[6] Click [Login] after you have selected all tokens.

There are 36 items in each group of symbols, but there are only 4 x 4 = 16 tokens in the table, so some symbols will be missing. This is a security feature, done on purpose to make guessing your passcode much harder.

Tokens

23	①	☆	♠	G4 350
27	?	♥	♠	G4 353
34	Ⓢ	=	♠	G4 356
1	Ⓚ	×	☒	G4 359
G4 362				
G4 365				
Login				G4 370

↑ Current

Fig. 9C GATE_4 Graphic_Login

☐ Demo

User Id admin G4 307

Login Result
 Login Failed ☒
 G4 381

[A] The table in the center has 4 x 4 = 16 token buttons. Each token button has 4 symbols. They are arranged into 4 groups :

[1] Numbers [2] Characters [3] Signs [4] Other symbols

Mouse over the sample token above to see all the 36 items in each group.

The positions of the above groups are fixed in any token, so numbers will always show up at the upper left corner, characters will always show up at the upper right corner of a token, etc. This may help you quickly locate and enter your pin from the token table in the center →

e.g. If your passcode is : ① ♥ 2 ☼
 Then the following tokens containing your pins and are, therefore, valid.

*	①	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*

	1	2	3	4
A	13 ①	4 ①	26 ①	10 ①
B	19 ②	25 *	14 ②	32 ②
C	31 &	21 ③	18 ③	5 ③
D	8 ④	9 ?	27 -	34 #

G4 321

[B] The rules are as follows :

[1] Enter your User_Id.

[2] Hit the [Enter] key or button.

[3] Follow the order of your pins.

[4] Pick the tokens that contain your pins from the token table ← in the center.

[5] If your pin is not on any token in the table, you can, and must, select any token for that pin.

[6] Click [Login] after you have selected all tokens.

There are 36 items in each group of symbols, but there are only 4 x 4 = 16 tokens in the table, so some symbols will be missing. This is a security feature, done on purpose to make guessing your passcode much harder.

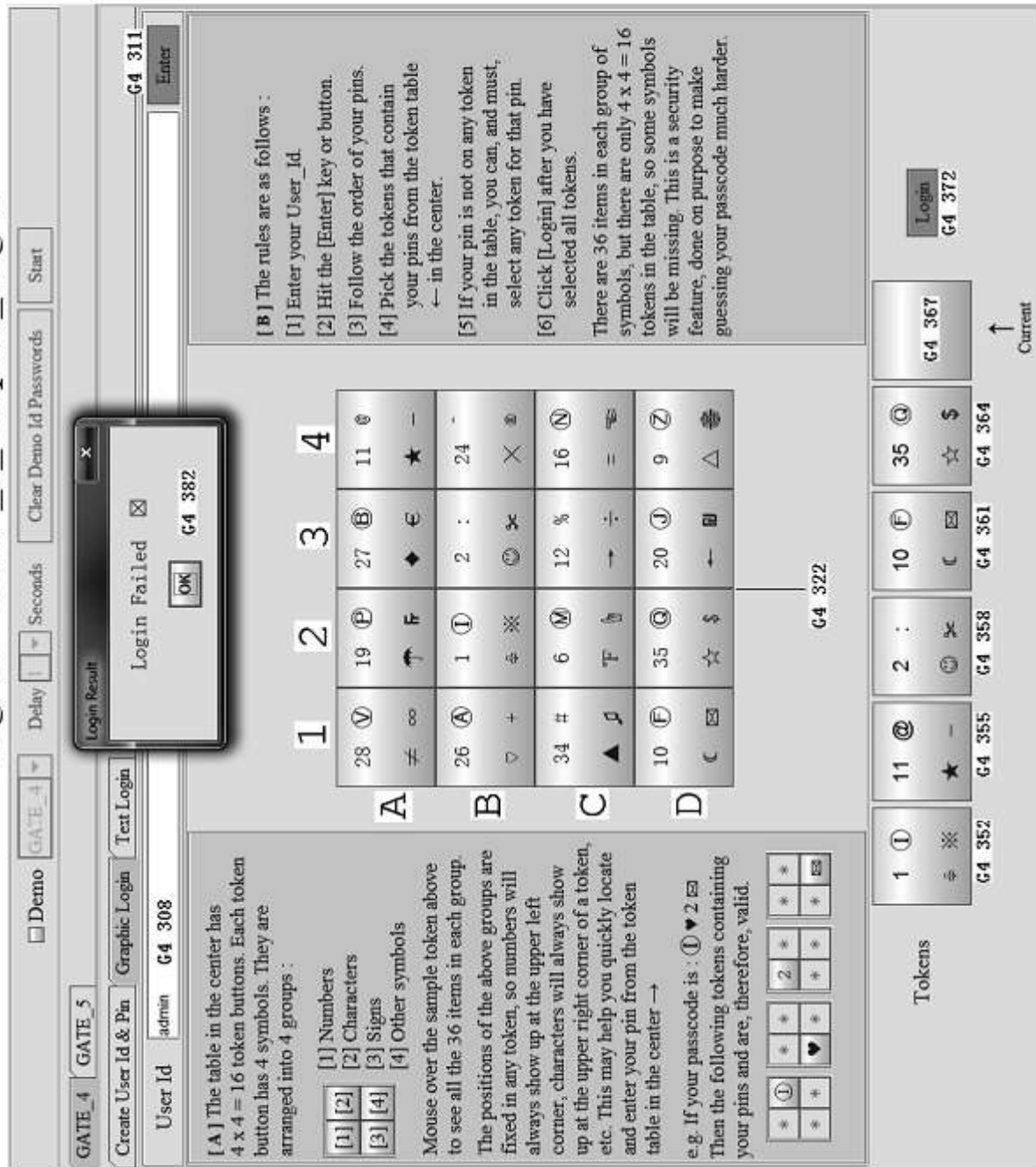
Tokens

13 ①	4 ①	5 ①	32 ②
13 ①	4 ①	5 ①	32 ②
13 ①	4 ①	5 ①	32 ②
13 ①	4 ①	5 ①	32 ②

G4 351 G4 354 G4 357 G4 360

↑ Current

G4 363 G4 366 G4 371



[illegible]

Fig. 10B GATE_4_Text_Login

☒ Demo GATE_4 Delay 1 Seconds Clear Demo Id Passwords Start			
GATE_4 GATE_5			
Create User Id & Pn Graphic Login Text Login			
User Id admin G4 406 G4 409 Enter			
Client Side Id : admin G4 411 Get following tokens from network [1] 24 ② ∴ ☆ [2] 17 ⑦ ★ ⇒ [3] 25 ⑧ □ ⊗ [4] 3 ⑩ ← ∫ [5] 7 - ● ⊗ [6] 16 ⑭ ≠ ♣ [7] 1 ⑥ × ⊗ [8] 21 ① € € [9] 18 ⑨ → \$ [10] 31 ③ ∅ ∞ [11] 34 ④ = ⑧ [12] 6 ⑤ ∠ † [13] 15 ⑯ ⊖ - [14] 23 ① ☆ ∫ [15] 27 ? ♥ ♣ [16] 9 ⑩ √ ※	Network Connection User_Id sent to server : admin G4 413 G4 422 Network passes info from server to client [1] 24 ② ∴ ☆ [2] 17 ⑦ ★ ⇒ [3] 25 ⑧ □ ⊗ [4] 3 ⑩ ← ∫ [5] 7 - ● ⊗ [6] 16 ⑭ ≠ ♣ [7] 1 ⑥ × ⊗ [8] 21 ① € € [9] 18 ⑨ → \$ [10] 31 ③ ∅ ∞ [11] 34 ④ = ⑧ [12] 6 ⑤ ∠ † [13] 15 ⑯ ⊖ - [14] 23 ① ☆ ∫ [15] 27 ? ♥ ♣ [16] 9 ⑩ √ ※	Server Side Id [Passcode] : admin [① ♥ 2 ⊗] G4 417 Generate from Passcode on file [1] 24 ② ∴ ☆ [2] 17 ⑦ ★ ⇒ [3] 25 ⑧ □ ⊗ [4] 3 ⑩ ← ∫ [5] 7 - ● ⊗ [6] 16 ⑭ ≠ ♣ [7] 1 ⑥ × ⊗ [8] 21 ① € € [9] 18 ⑨ → \$ [10] 31 ③ ∅ ∞ [11] 34 ④ = ⑧ [12] 6 ⑤ ∠ † [13] 15 ⑯ ⊖ - [14] 23 ① ☆ ∫ [15] 27 ? ♥ ♣ [16] 9 ⑩ √ ※	
[6] G4 426 Select tokens contain user pins G4 350 [①] found in [23 ① ☆ ∫] G4 353 [♥] found in [27 ? ♥ ♣] G4 356 [2] missing : replaced by wildcard G4 359 [⊗] found in [1 ⑥ × ⊗]	Send selected tokens from client to server G4 350 [23 ① ☆ ∫] G4 353 [27 ? ♥ ♣] G4 356 [34 ④ = ⑧] [wildcard] G4 359 [1 ⑥ × ⊗] [7] G4 427 [8] G4 428	Get selected tokens from network G4 350 [23 ① ☆ ∫] G4 429 G4 353 [27 ? ♥ ♣] G4 429 G4 356 [34 ④ = ⑧] G4 429 G4 359 [1 ⑥ × ⊗] G4 429	
Access Granted ^ ^ ^ !	Send login permission to client G4 431 G4 430 [11] [10]	Verify tokens from client G4 429 [9]	

Fig. 10C GATE_4_Text_Login

☐ Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start	
GATE_4 GATE_5	
Create User Id & Pn Graphic Login Text Login	
User Id admin G4 506	
Server Side G4 509	
Id [Passcode] : admin [0 2]	
Generate from Passcode on file	
[1] 13 ① .: ③ [2] 4 ④ ① [3] 26 ② ☆ [4] 10 ⑤ ② [5] 19 ⑥ ③ [6] 25 * ⑦ [7] 14 ⑧ ④ [8] 32 ⑨ ⑤ [9] 31 & ⑩ [10] 21 ⑪ ⑥ [11] 18 ⑫ ⑦ [12] 5 ⑬ ⑧ [13] 8 ⑭ ⑨ [14] 9 ? ⑮ [15] 27 ^ ⑯ [16] 34 # × ⑰	
Network Connection	
User Id sent to server : admin G4 513	
Network passes info from server to client	
[1] 13 ① .: ③ [2] 4 ④ ① [3] 26 ② ☆ [4] 10 ⑤ ② [5] 19 ⑥ ③ [6] 25 * ⑦ [7] 14 ⑧ ④ [8] 32 ⑨ ⑤ [9] 31 & ⑩ [10] 21 ⑪ ⑥ [11] 18 ⑫ ⑦ [12] 5 ⑬ ⑧ [13] 8 ⑭ ⑨ [14] 9 ? ⑮ [15] 27 ^ ⑯ [16] 34 # × ⑰	
Client Side	
Id : admin G4 511	
Get following tokens from network	
[1] 13 ① .: ③ [2] 4 ④ ① [3] 26 ② ☆ [4] 10 ⑤ ② [5] 19 ⑥ ③ [6] 25 * ⑦ [7] 14 ⑧ ④ [8] 32 ⑨ ⑤ [9] 31 & ⑩ [10] 21 ⑪ ⑥ [11] 18 ⑫ ⑦ [12] 5 ⑬ ⑧ [13] 8 ⑭ ⑨ [14] 9 ? ⑮ [15] 27 ^ ⑯ [16] 34 # × ⑰	
Select tokens contain user pins	
G4 351 [1] found in [13 ① .: ③] G4 354 [2] not found in [32 ⑨ ⑤] G4 357 [2] missing : replaced by wildcard G4 360 [2] found in [4 ④ ①]	
Send selected tokens from client to server	
G4 351 [13 ① .: ③] G4 354 [32 ⑨ ⑤] G4 357 [5 ⑬ ⑧] [wildcard] G4 360 [4 ④ ①]	
Get selected tokens from network	
G4 351 [13 ① .: ③] ✓ C21 G4 354 [32 ⑨ ⑤] ✓ C22 G4 357 [5 ⑬ ⑧] ✓ C23 G4 360 [4 ④ ①] ✓ C24	
Verify tokens from client	
Access Denied !	

Fig. 10D GATE_4 Text Login

☐ Demo GATE_4 Delay 1 Seconds Clear Demo Id Passwords Start			
GATE_4 GATE_5 Create User Id & Pm Graphic Login Text Login User Id admin G4 606 G4 609 Enter			
Client Side Id : admin G4 611	Network Connection User Id sent to server : admin G4 613	Server Side Id [Passcode] : admin [1 2 3] G4 615	
Get following tokens from network [1] 28 V ≠ ∞ [2] 19 P ↑ F [3] 27 B + € [4] 11 @ ★ - [5] 26 A ∇ + [6] 1 ① * ※ [7] 2 : @ ≈ [8] 24 ^ × ⊗ [9] 34 # ▲ J [10] 6 M P h [11] 12 % → + [12] 16 N = ∞ [13] 10 F € ☒ [14] 35 Q ☆ \$ [15] 20 ① ← ∞ [16] 9 Z Δ ⊗	Network passes info from server to client [1] 28 V ≠ ∞ [2] 19 P ↑ F [3] 27 B + € [4] 11 @ ★ - [5] 26 A ∇ + [6] 1 ① * ※ [7] 2 : @ ≈ [8] 24 ^ × ⊗ [9] 34 # ▲ J [10] 6 M P h [11] 12 % → + [12] 16 N = ∞ [13] 10 F € ☒ [14] 35 Q ☆ \$ [15] 20 ① ← ∞ [16] 9 Z Δ ⊗	Generate from Passcode on file [1] 28 V ≠ ∞ [2] 19 P ↑ F [3] 27 B + € [4] 11 @ ★ - [5] 26 A ∇ + [6] 1 ① * ※ [7] 2 : @ ≈ [8] 24 ^ × ⊗ [9] 34 # ▲ J [10] 6 M P h [11] 12 % → + [12] 16 N = ∞ [13] 10 F € ☒ [14] 35 Q ☆ \$ [15] 20 ① ← ∞ [16] 9 Z Δ ⊗	G4 621 G4 622 G4 623
Select tokens contain user pins [6] G4 626 G4 352 [①] found in [1 ① * ※] G4 355 [♥] missing : replaced by wildcard G4 358 [2] found in [2 : @ ≈] G4 361 [☒] found in [10 F € ☒] G4 364 [?] [35 Q ☆ \$] too many pin	Send selected tokens from client to server G4 352 [1 ① * ※] G4 355 [11 @ ★ -] [wildcard] G4 358 [2 : @ ≈] G4 361 [10 F € ☒] G4 364 [35 Q ☆ \$] [invalid]	Get selected tokens from network G4 352 [1 ① * ※] ☑ C31 G4 355 [11 @ ★ -] ☑ C32 G4 358 [2 : @ ≈] ☑ C33 G4 361 [10 F € ☒] ☑ C34 G4 364 [35 Q ☆ \$] ☑ C35	G4 627 G4 628 G4 629
Access Denied !	Send login permission to client G4 630	Verify tokens from client G4 629	G4 631 G4 632 G4 633

Fig. 11B GATE_5 Create Id

☒ Demo
 GATE_5
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5

Create User Id & Pm
 Graphic Login
 Text Login

User Id admin G5 206
 G5 208
 Check Availability

[1]	[2]	G5 210										G5 212																	
[3]		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
[4]	[5]																												

[1]	[2]	G5 220										G5 222																	
[3]		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ι		
[4]	[5]																												

[1]	[2]	G5 230										G5 232																	
[3]		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26		
[4]	[5]																												

[1]	[2]	G5 240										G5 242																	
[3]		○	●	△	▲	□	■	☆	★	♠	♣	♥	♦	♠	♣	♥	♦	♠	♣	♥	♦	♠	♣	♥	♦	♠	♣		
[4]	[5]																												

[1]	[2]	G5 246										G5 248																	
[3]		+	-	x	÷	←	→	↑	↓	∞	∫	∂	⊕	⊗	°C	°F	⊞	⊟	⊠	⊡	⊢	⊣	⊤	⊥	⊦	⊧			
[4]	[5]																												

Pick at least 4 pins from any of the above 5 categories : [1] English alphabet , [2] Greek characters , [3] Numbers , [4] Symbols , [5] Other signs. You can choose from the same category or different categories. And you can choose the same pin multiple times : ☺♥♣♠
 These pins will be your passcode when you login, pick something that is easy for you to remember and yet hard for anyone else to guess.

Select Pins		\$	=	M	C	2	☺	Save	G5 270
							↑	Current	

Fig. 12C GATE_5_Graphic_Login

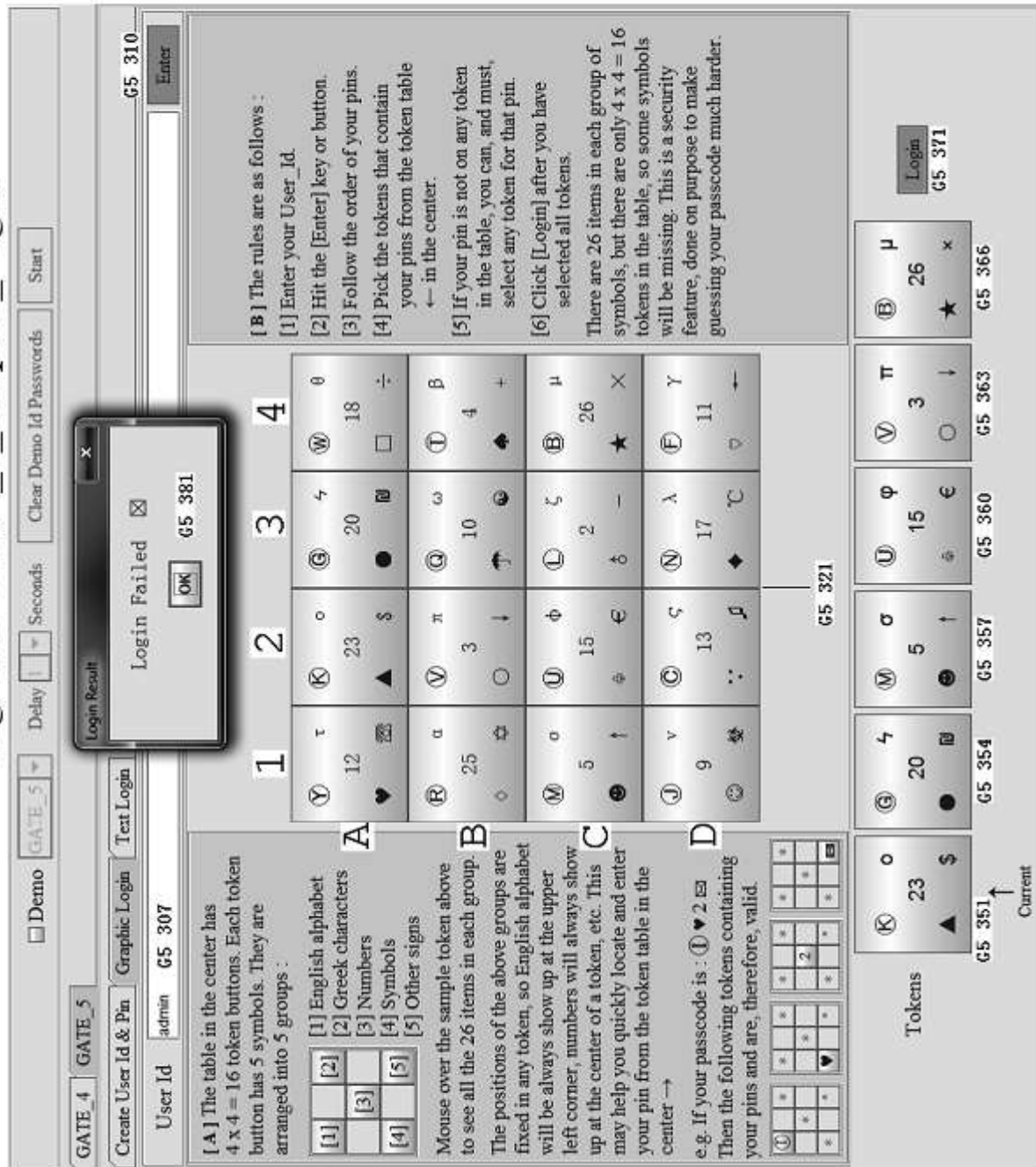


Fig. 12D GATE_5 Graphic_Login

☒ Demo
 GATE_5
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5
 Create User Id & Pn
 Graphic Login
 Text Login

User Id admin G5 308
 Enter

Login Result
 Login Failed
 OK G5 382

[A] The table in the center has 4 x 4 = 16 token buttons. Each token button has 5 symbols. They are arranged into 5 groups :

[1]	[2]	[3]	[4]	[5]
[1] English alphabet	[2] Greek characters	[3] Numbers	[4] Symbols	[5] Other signs

Mouse over the sample token above to see all the 26 items in each group.

The positions of the above groups are fixed in any token, so English alphabet will be always show up at the upper left corner, numbers will always show up at the center of a token, etc. This may help you quickly locate and enter your pin from the token table in the center →

e.g. If your passcode is : ① ♥ 2 ☼

Then the following tokens containing your pins are, therefore, valid.

①	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*

[B] The rules are as follows :

- [1] Enter your User_Id.
- [2] Hit the [Enter] key or button.
- [3] Follow the order of your pins.
- [4] Pick the tokens that contain your pins from the token table ← in the center.
- [5] If your pin is not on any token in the table, you can, and must, select any token for that pin.
- [6] Click [Login] after you have selected all tokens.

There are 26 items in each group of symbols, but there are only 4 x 4 = 16 tokens in the table, so some symbols will be missing. This is a security feature, done on purpose to make guessing your passcode much harder.

1	2	3	4
Ⓜ 15 ☆	ⓔ 4 ∴	ⓐ 6 ★	Ⓥ 22 ⚡
Ⓡ 21 Ⓡ	Ⓚ 19 ♣	Ⓞ 24 ♦	Ⓨ 14 ▲
Ⓣ 5 =	ⓐ 17 ☺	Ⓢ 16 ⚙	Ⓡ 2 ♀
Ⓛ 7 ♠	Ⓣ 23 ≠	Ⓢ 9 ∴	Ⓟ 26 ♥

G5 322

Tokens

Ⓢ 9 ∴	Ⓢ 15 ☆	Ⓢ 9 ∴	Ⓢ 2 ♀
G5 352	G5 355	G5 358	G5 361

↑ Current

Login G5 372

[illegible]

Fig. 13B GATE_5_Text_Login

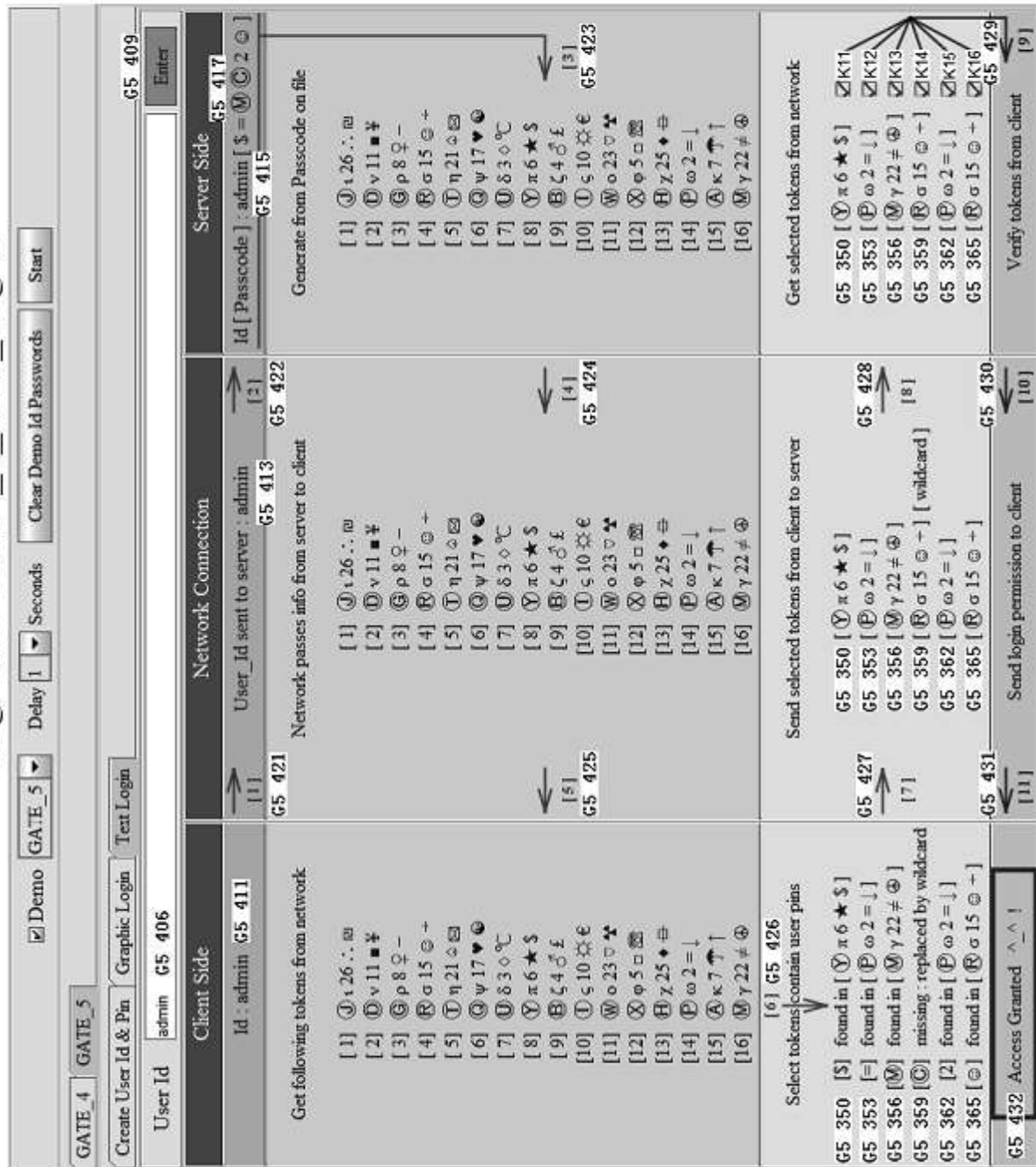


Fig. 13C GATE_5 Text_Login

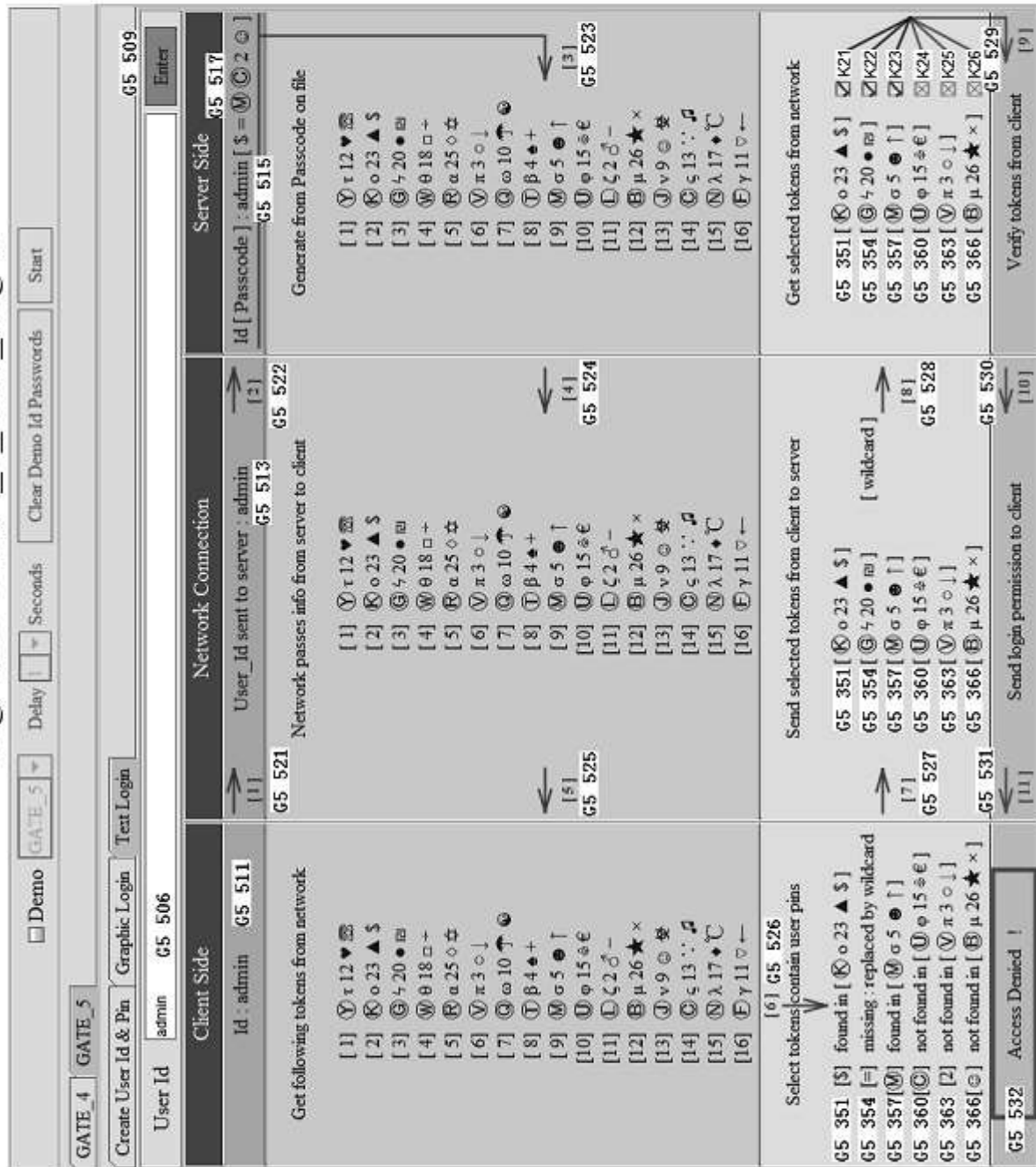


Fig. 13D GATE_5 Text Login

GATE_4		GATE_5		G5 609	
☒ Demo		Delay 1 Seconds		Clear Demo Id Passwords	
Create User Id & Pm		Graphic Login		Text Login	
User Id admin		G5 606		Enter	
Client Side		Network Connection		Server Side	
Id : admin G5 611		User Id sent to server : admin G5 613		Id [Passcode] : admin [\$ = M 2] G5 617	
Get following tokens from network		Network passes info from server to client		Generate from Passcode on file	
[1] M λ 15 ☆ F [2] E π 4 . : x [3] G δ 6 ★ @ [4] V β 22 & ¥ [5] R ξ 21 ∇ @ [6] K ρ 19 ♣ W [7] Q ε 24 ↓ [8] Y γ 14 ▲ £ [9] T ι 5 ← [10] Q κ 17 @ J [11] U α 16 ☆ + [12] H υ 2 □ → [13] L χ 7 ♣ ♣ [14] Q 4 23 ≠ # [15] @ ω 9 . : ∞ [16] P ϕ 26 ♥ P		[1] M λ 15 ☆ F [2] E π 4 . : x [3] G δ 6 ★ @ [4] V β 22 & ¥ [5] R ξ 21 ∇ @ [6] K ρ 19 ♣ W [7] Q ε 24 ↓ [8] Y γ 14 ▲ £ [9] T ι 5 ← [10] Q κ 17 @ J [11] U α 16 ☆ + [12] H υ 2 □ → [13] L χ 7 ♣ ♣ [14] Q 4 23 ≠ # [15] @ ω 9 . : ∞ [16] P ϕ 26 ♥ P		[1] M λ 15 ☆ F [2] E π 4 . : x [3] G δ 6 ★ @ [4] V β 22 & ¥ [5] R ξ 21 ∇ @ [6] K ρ 19 ♣ W [7] Q ε 24 ↓ [8] Y γ 14 ▲ £ [9] T ι 5 ← [10] Q κ 17 @ J [11] U α 16 ☆ + [12] H υ 2 □ → [13] L χ 7 ♣ ♣ [14] Q 4 23 ≠ # [15] @ ω 9 . : ∞ [16] P ϕ 26 ♥ P	
Select tokens contain user pins		Send selected tokens from client to server		Get selected tokens from network	
[6] G5 626 G5 352 [\$] missing : replaced by wildcard G5 355 [=] found in [T ι 5 ←] G5 358 [M] found in [M λ 15 ☆ F] G5 361 [@] found in [@ ω 9 . : ∞] G5 364 [2] found in [H υ 2 □ →] G5 367 [@] input missing		G5 352 [@ ω 9 . : ∞] [wildcard] G5 355 [T ι 5 ←] G5 358 [M λ 15 ☆ F] G5 361 [@ ω 9 . : ∞] G5 364 [H υ 2 □ →] G5 367 Pm missing		G5 352 [@ ω 9 . : ∞] G5 355 [T ι 5 ←] G5 358 [M λ 15 ☆ F] G5 361 [@ ω 9 . : ∞] G5 364 [H υ 2 □ →] G5 367 ? Verify tokens from client	
G5 632 Access Denied !		Send login permission to client		G5 629 [9]	

Fig. 14 GATE_4 Encryption

☒ Demo
 GATE_5
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5
 Create User Id & Pm
 Graphic Login
 Text Login
 Encryption

Plain Message
 secret
 G4 700
 Sender Passcode
 123
 G4 702
 G4 703
 Encrypt

[A] GATE system can be used for encryption, this is an example of using GATE_4 to encrypt a message.

Characters in a message are mixed in with other filler characters to hide the message.

Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens.

The table and key of each character is generated with sender passcode.

Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.

[B] As shown below, GATE_4 is used to decrypt a message.

After receiving all characters from a sender, it goes through each character to see if it's attached key can be authenticated against it's attached table of 16 tokens.

Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message.

Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.

Key Tokens
 Check

Encrypted Message Sent / Received
 Decrypted Message

Current
 G4 704
 G4 706
 Receiver Passcode
 123
 Decrypt

Fig. 15A GATE_4 Encryption

☒ Demo
 GATE_5
 Delay 1 Seconds
 Clear D

GATE_4
 GATE_5
 Create User Id & Pm
 Graphic Login
 Text Login
 Encryption

Check Successful ☒
 OK
 G4 730

Sender Passcode 123
 G4 702
 Encrypt

[A] GATE system can be used for encryption, this is an example of using GATE_4 to encrypt a message.

Characters in a message are mixed in with other filler characters to hide the message.

Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens.

The table and key of each character is generated with sender passcode.

Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.

S		G4 712	
2B J	26 V	1 &	9 %
☆	★	∞	♂
7 :	31 ©	5 K	23 A
⊗	♦	†	∇
4 !	30 *	17 X	2 U
○	✓	⊙	⊗
13 ?	27 L	22 H	21 T
■	π	÷	□
			●
			F

G4 720
 G4 722
 G4 724
 G4 714

Key Tokens
 1 &
 2 U
 4 I
 Check
 G4 726

↑ Current

Encrypted Message G4 710
 G4 704
 Decrypt

Decrypted Message
 Receiver Passcode 123
 G4 706
 Decrypt

[B] As shown below, GATE_4 is used to decrypt a message.
 After receiving all characters from a sender, it goes through each character to see if it's attached key can be authenticated against it's attached table of 16 tokens.
 Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message.
 Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.

Fig. 15B GATE_4_Encryption

☒ Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start	
GATE_4 GATE_5	
Create User Id & Pm Graphic Login Text Login Encryption	
Message 5 G4 750 Enter	
Receiver Side Passcode : 123 G4 760 Get following tokens from network [1] [28, ①, ☆, ∞] [2] [26, ⑦, ★, ∞] [3] [1, &, ♂, ④] [4] [9, %, →, ③] [5] [7, ∞, ☆, ※] [6] [31, ③, +, †] [7] [5, ⑧, ∇, \$] [8] [23, ⑥, ∅, ∞] [9] [4, 1, ∅, λ] [10] [30, *, √, ②] [11] [17, ⑨, &, ⑨] [12] [2, ⑦, ×, ②] [13] [13, ?, ■, π] [14] [27, ①, ♣, ∞] [15] [22, ⑧, □, ×] [16] [21, ⑦, •, F]	
Network Connection Tokens Sent With Message Network passes info from sender to receiver [1] [28, ①, ☆, ∞] [2] [26, ⑦, ★, ∞] [3] [1, &, ♂, ④] [4] [9, %, →, ③] [5] [7, ∞, ☆, ※] [6] [31, ③, +, †] [7] [5, ⑧, ∇, \$] [8] [23, ⑥, ∅, ∞] [9] [4, 1, ∅, λ] [10] [30, *, √, ②] [11] [17, ⑨, &, ⑨] [12] [2, ⑦, ×, ②] [13] [13, ?, ■, π] [14] [27, ①, ♣, ∞] [15] [22, ⑧, □, ×] [16] [21, ⑦, •, F]	
Sender Side Passcode : 123 G4 752 Generate from Passcode on file [1] [28, ①, ☆, ∞] [2] [26, ⑦, ★, ∞] [3] [1, &, ♂, ④] [4] [9, %, →, ③] [5] [7, ∞, ☆, ※] [6] [31, ③, +, †] [7] [5, ⑧, ∇, \$] [8] [23, ⑥, ∅, ∞] [9] [4, 1, ∅, λ] [10] [30, *, √, ②] [11] [17, ⑨, &, ⑨] [12] [2, ⑦, ×, ②] [13] [13, ?, ■, π] [14] [27, ①, ♣, ∞] [15] [22, ⑧, □, ×] [16] [21, ⑦, •, F]	
Receiver Side Passcode : 123 G4 762 Checking Key Tokens G4 720 [1] found in [1 & ♂ ④] G4 722 [2] found in [2 ⑦ × ②] G4 724 [3] missing : replaced by wildcard	
Receiver Side Result Key Tokens G4 764 G4 766 G4 768 Receiver Side Verify Key Tokens G4 720 [1 & ♂ ④] G4 722 [2 ⑦ × ②] G4 724 [4 1 ∅ λ]	
Message Valid G4 770 Message Valid ^ _ !	

Fig. 16A GATE_4 Encryption

☒ Demo
 GATE_5 ▾ Delay 1 ▾ Seconds Clear D

GATE_4 GATE_5

Create User Id & Pm Graphic Login Text Login Encryption

Plain Message secret G4 700

Sender Passcode 123 G4 702

Encrypt G4 703

Check Failed ☒ OK G4 731

[A] GATE system can be used for encryption, this is an example of using GATE_4 to encrypt a message.

Characters in a message are mixed in with other filler characters to hide the message.

Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens.

The table and key of each character is generated with sender passcode.

Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.

L G4 713			
30 U	18 ~	3 A	13 %
→ @	◇ †	● ※	= ☼
25 ©	24 N	35 M	36 ?
♣ £	☉ ☼	■ @	↑ ♪
33 I	29 X	19 D	34 W
△ ☼	□ F	▲ ☼	◇ ::
4 R	26 Y	9 K	10 Q
∴ ♪	★ +	♠ @	♣ ♣

G4 715

[B] As shown below, GATE_4 is used to decrypt a message.

After receiving all characters from a sender, it goes through each character to see if it's attached key can be authenticated against it's attached table of 16 tokens.

Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message.

Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.

Check G4 726

Key Tokens

19 D ▲ ☼ G4 721 G4 723 G4 725 G4 727

24 N ☼ ☼ G4 723 G4 725 G4 727

36 ? ☼ ♪ G4 723 G4 725 G4 727

Current ↑

Encrypted Message Sent / Received

L e Q W N C r M f Y e M H Q r G4 704

Decrypted Message

secret G4 708 Receiver Passcode 123 G4 706

Decrypt

Fig. 16B GATE_4_Encryption

☒ Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start															
GATE_4 GATE_5															
Create User Id & Pm Graphic Login Text Login Encryption															
Message L G4 751 Enter															
<table border="1"> <thead> <tr> <th>Receiver Side</th> <th>Network Connection</th> <th>Sender Side</th> </tr> </thead> <tbody> <tr> <td> Passcode : 123 G4 760 </td> <td> Tokens Sent With Message </td> <td> Passcode : 123 G4 752 </td> </tr> <tr> <td> Get following tokens from network [1] [30, 0, →, 0] [2] [18, -, 0, f] G4 790 [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] G4 792 [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0] </td> <td> Network passes info from sender to receiver [1] [30, 0, →, 0] [2] [18, -, 0, f] [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0] </td> <td> Generate from Passcode on file [1] [30, 0, →, 0] [2] [18, -, 0, f] [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0] </td> </tr> <tr> <td> Checking Key Tokens G4 721 [1] missing : replaced by wildcard G4 723 [2] missing : replaced by wildcard G4 725 [3] not found in [36 ? ↑ J] G4 727 [?] [19 0 ▲ 0] too many pin </td> <td> Receiver Side Result Key Tokens G4 765 G4 721 [19 0 ▲ 0] [wildcard] G4 767 G4 723 [24 0 0 0] [wildcard] [5] G4 725 [36 ? ↑ J] G4 727 [19 0 ▲ 0] [invalid] G4 769 </td> <td> Receiver Side Verify Key Tokens G4 721 [19 0 ▲ 0] ☒ C61 G4 723 [24 0 0 0] ☒ C62 G4 725 [36 ? ↑ J] ☒ C63 G4 727 [19 0 ▲ 0] ☒ C64 </td> </tr> <tr> <td colspan="2"> G4 771 Message Invalid ! </td> </tr> </tbody> </table>		Receiver Side	Network Connection	Sender Side	Passcode : 123 G4 760	Tokens Sent With Message	Passcode : 123 G4 752	Get following tokens from network [1] [30, 0, →, 0] [2] [18, -, 0, f] G4 790 [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] G4 792 [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0]	Network passes info from sender to receiver [1] [30, 0, →, 0] [2] [18, -, 0, f] [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0]	Generate from Passcode on file [1] [30, 0, →, 0] [2] [18, -, 0, f] [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0]	Checking Key Tokens G4 721 [1] missing : replaced by wildcard G4 723 [2] missing : replaced by wildcard G4 725 [3] not found in [36 ? ↑ J] G4 727 [?] [19 0 ▲ 0] too many pin	Receiver Side Result Key Tokens G4 765 G4 721 [19 0 ▲ 0] [wildcard] G4 767 G4 723 [24 0 0 0] [wildcard] [5] G4 725 [36 ? ↑ J] G4 727 [19 0 ▲ 0] [invalid] G4 769	Receiver Side Verify Key Tokens G4 721 [19 0 ▲ 0] ☒ C61 G4 723 [24 0 0 0] ☒ C62 G4 725 [36 ? ↑ J] ☒ C63 G4 727 [19 0 ▲ 0] ☒ C64	G4 771 Message Invalid !	
Receiver Side	Network Connection	Sender Side													
Passcode : 123 G4 760	Tokens Sent With Message	Passcode : 123 G4 752													
Get following tokens from network [1] [30, 0, →, 0] [2] [18, -, 0, f] G4 790 [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] G4 792 [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0]	Network passes info from sender to receiver [1] [30, 0, →, 0] [2] [18, -, 0, f] [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0]	Generate from Passcode on file [1] [30, 0, →, 0] [2] [18, -, 0, f] [3] [3, 0, •, ※] [4] [13, %, =, 0] [5] [25, 0, •, f] [6] [24, 0, 0, 0] [7] [35, 0, ■, 0] [8] [36, ?, ↑, J] [9] [33, 0, △, 0] [10] [29, 0, □, f] [11] [19, 0, ▲, 0] [12] [34, 0, 0, ::] [13] [4, 0, :, : , p] [14] [26, 0, ★, +] [15] [9, 0, •, 0] [16] [10, 0, ✕, 0]													
Checking Key Tokens G4 721 [1] missing : replaced by wildcard G4 723 [2] missing : replaced by wildcard G4 725 [3] not found in [36 ? ↑ J] G4 727 [?] [19 0 ▲ 0] too many pin	Receiver Side Result Key Tokens G4 765 G4 721 [19 0 ▲ 0] [wildcard] G4 767 G4 723 [24 0 0 0] [wildcard] [5] G4 725 [36 ? ↑ J] G4 727 [19 0 ▲ 0] [invalid] G4 769	Receiver Side Verify Key Tokens G4 721 [19 0 ▲ 0] ☒ C61 G4 723 [24 0 0 0] ☒ C62 G4 725 [36 ? ↑ J] ☒ C63 G4 727 [19 0 ▲ 0] ☒ C64													
G4 771 Message Invalid !															

Fig. 17 GATE_4 Encryption

☒ Demo
 GATE_5
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5
 Create User Id & Pn
 Graphic Login
 Text Login
 Encryption

Plain Message
 secret
 G4 700
 Sender Passcode
 123
 G4 702
 Encrypt

[A] GATE system can be used for encryption, this is an example of using GATE_4 to encrypt a message.

Characters in a message are mixed in with other filler characters to hide the message.

Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens.

The table and key of each character is generated with sender passcode.

Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.

T			
3 ①	35 ⑨	36 &	25 ⑩
∴ ∞	△ ⊗	∴ ∞	✓ ♣
33 ④	5 ①	4 ②	1 ^
★ ⊕	▲ β	□ √	≠ Σ
8 *	18 ⑤	12 ③	9 ?
← ⑧	② ⊗	→ ⑥	■ -
7 ②	32 :	27 #	11 ①
○ ⊗	↑ π	● ⊗	⊗ \$

[B] As shown below, GATE_4 is used to decrypt a message.

After receiving all characters from a sender, it goes through each character to see if it's attached key can be authenticated against it's attached table of 16 tokens.

Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message.

Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.

Key Tokens
 35 ⑨
 32 :
 3 ①
 ∴ ∞
 ↑
 Current

Encrypted Message Sent / Received
 Decrypted Message
 Receiver Passcode
 567
 G4 701
 Decrypt

☒ Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start	
Create User Id & Pin Graphic Login Text Login Encryption	G5 702 G5 703 Encrypt
Plain Message FYEO G5 700	Sender Passcode 123 Decrypt

[A] GATE system can be used for encryption, this is an example of using GATE_5 to encrypt a message. Characters in a message are mixed in with other filler characters to hide the message. Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens. The table and key of each character is generated with sender passcode. Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.	[B] As shown below, GATE_5 is used to decrypt a message. After receiving all characters from a sender, it goes through each character to see if it's attached key can be authenticated against it's attached table of 16 tokens. Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message. Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.
---	---

Key Tokens		Check
------------	--	-------

Encrypted Message Sent / Received	Current: ↑ G5 704 F I P R O J C Y N E B A O	G5 706 Decrypt
		Receiver Passcode 123

Fig. 19A GATE_5_Encryption

☒ Demo GATE_5 ▼ Delay 1 Seconds Clear D...		Create User Id & Pn Graphic Login Text Login Encryption	
GATE_4 GATE_5 Plain Message FVEO G5 700		Sender Password 123 G5 702 Encrypt	
[A] GATE system can be used for encryption, this is an example of using GATE_5 to encrypt a message. Characters in a message are mixed in with other filler characters to hide the message. Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens. The table and key of each character is generated with sender passcode. Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.		[B] As shown below, GATE_5 is used to decrypt a message. After receiving all characters from a sender, it goes through each character to see if its attached key can be authenticated against its attached table of 16 tokens. Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message. Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.	
Key Tokens G5 720 — Current G5 722 G5 724 G5 704		Check G5 726	
Encrypted Message Sent / Received G5 710		Receiver Password 123 G5 706 Decrypt	

Fig. 19B GATE_5_Encryption

☒ Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start	
GATE_4 GATE_5 Create User Id & Pm Graphic Login Text Login Encryption	
Message F G5 750 Enter	
Receiver Side Passcode : 123 G5 760 Get following tokens from network [1] [R] 0, 7, ■, 8 [2] [A] 1, 16, ·, + [3] [M] 1, 11, ✕, 0 [4] [Q] 0, 4, *, 1 [5] [T] 1, 22, 0, F [6] [X] 1, 20, 0, + [7] [S] 1, 1, 1, 1 [8] [B] 1, 21, 1, 1 [9] [Z] 1, 8, 0, 1 [10] [C] 1, 18, 1, 1 [11] [K] 1, 2, 0, 1 [12] [D] 1, 13, 1, ✕ [13] [N] 1, 3, 0, 1 [14] [E] 1, 15, 0, F [15] [U] 1, 19, 1, 1 [16] [W] 1, 17, 0, 1 Network Connection Tokens Sent With Message [1] [R] 0, 7, ■, 8 [2] [A] 1, 16, ·, + [3] [M] 1, 11, ✕, 0 [4] [Q] 0, 4, *, 1 [5] [T] 1, 22, 0, F [6] [X] 1, 20, 0, + [7] [S] 1, 1, 1, 1 [8] [B] 1, 21, 1, 1 [9] [Z] 1, 8, 0, 1 [10] [C] 1, 18, 1, 1 [11] [K] 1, 2, 0, 1 [12] [D] 1, 13, 1, ✕ [13] [N] 1, 3, 0, 1 [14] [E] 1, 15, 0, F [15] [U] 1, 19, 1, 1 [16] [W] 1, 17, 0, 1 Sender Side Passcode : 123 G5 752 Generate from Passcode on file [1] [R] 0, 7, ■, 8 [2] [A] 1, 16, ·, + [3] [M] 1, 11, ✕, 0 [4] [Q] 0, 4, *, 1 [5] [T] 1, 22, 0, F [6] [X] 1, 20, 0, + [7] [S] 1, 1, 1, 1 [8] [B] 1, 21, 1, 1 [9] [Z] 1, 8, 0, 1 [10] [C] 1, 18, 1, 1 [11] [K] 1, 2, 0, 1 [12] [D] 1, 13, 1, ✕ [13] [N] 1, 3, 0, 1 [14] [E] 1, 15, 0, F [15] [U] 1, 19, 1, 1 [16] [W] 1, 17, 0, 1	

GATE_4	GATE_5	Demo	GATE_5 ▾	Delay 1 ▾ Seconds	Clear De																				
Create User Id & Pn		Graphic Login		Text Login																					
Plain Message		FVEO G5 700		Encryption																					
[A] GATE system can be used for encryption, this is an example of using GATE_5 to encrypt a message. Characters in a message are mixed in with other filler characters to hide the message. Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens. The table and key of each character is generated with sender passcode. Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.		P <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>G5 718</th> <th>G5 713</th> </tr> </thead> <tbody> <tr> <td>N 15</td> <td>⊙ 6</td> </tr> <tr> <td>◆ x</td> <td>≠ ♣</td> </tr> <tr> <td>H β</td> <td>⓪ σ</td> </tr> <tr> <td>O 12</td> <td>∇ ↓</td> </tr> <tr> <td>⊙ 8</td> <td>π ∞</td> </tr> <tr> <td>⊙ 24</td> <td>ξ 14</td> </tr> <tr> <td>⊙ 11</td> <td>Ⓢ 5</td> </tr> <tr> <td>⊙ 2</td> <td>Ⓢ 1</td> </tr> </tbody> </table>		G5 718	G5 713	N 15	⊙ 6	◆ x	≠ ♣	H β	⓪ σ	O 12	∇ ↓	⊙ 8	π ∞	⊙ 24	ξ 14	⊙ 11	Ⓢ 5	⊙ 2	Ⓢ 1	G5 703 [B] As shown below, GATE_5 is used to decrypt a message. After receiving all characters from a sender, it goes through each character to see if its attached key can be authenticated against its attached table of 16 tokens. Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message. Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.		G5 706 Decrypt	
G5 718	G5 713																								
N 15	⊙ 6																								
◆ x	≠ ♣																								
H β	⓪ σ																								
O 12	∇ ↓																								
⊙ 8	π ∞																								
⊙ 24	ξ 14																								
⊙ 11	Ⓢ 5																								
⊙ 2	Ⓢ 1																								
Key Tokens <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>G5 721</th> <th>G5 723</th> <th>G5 725</th> <th>G5 704</th> </tr> </thead> <tbody> <tr> <td>Ⓟ α 11</td> <td>Ⓞ 2</td> <td>Ⓢ 3</td> <td>Ⓢ 1</td> </tr> <tr> <td>Ⓢ -</td> <td>△ ♣</td> <td>∇ ↓</td> <td>Ⓢ 1</td> </tr> </tbody> </table> Current		G5 721	G5 723	G5 725	G5 704	Ⓟ α 11	Ⓞ 2	Ⓢ 3	Ⓢ 1	Ⓢ -	△ ♣	∇ ↓	Ⓢ 1	Sender Passcode 123 G5 702 Encrypt		Check Failed ☒ G5 731 OK									
G5 721	G5 723	G5 725	G5 704																						
Ⓟ α 11	Ⓞ 2	Ⓢ 3	Ⓢ 1																						
Ⓢ -	△ ♣	∇ ↓	Ⓢ 1																						
Encrypted Message Sent / Received		Receiver Passcode 123		Decrypt																					

Fig. 20B GATE_5_Encryption

☒ Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start		
GATE_4 GATE_5		
Create User Id & Pm Graphic Login Text Login Encryption		
Message P G5 751 Enter		
Receiver Side Passcode : 123 G5 760 Network Connection Tokens Sent With Message Sender Side Passcode : 123 G5 752		
Get following tokens from network G5 792 [1] [N, 0, 15, ✖, ✕] [2] [P, a, 11, ✨, -] [3] [Q, o, 6, ≠, ♣] [4] [Y, 7, 2, △, ¥] [5] [B, β, 12, o, f] [6] [Z, 4, 3, ∇, j] [7] [G, α, 26, ∴, l] [8] [F, μ, 10, ♣, ∞] [9] [Q, 5, 8, ∴, ∞] [10] [W, π, 22, ●, ∞] [11] [Q, κ, 13, ●, -] [12] [Q, δ, 19, o, ⊗] [13] [I, ξ, 24, o, ♠] [14] [W, ω, 14, ■, ∞] [15] [S, ζ, 5, ●, ⊙] [16] [R, v, 1, ★, ←-] G5 790 [16] [R, v, 1, ★, ←-]	Network passes info from sender to receiver [1] [N, 0, 15, ✖, ✕] [2] [P, a, 11, ✨, -] [3] [Q, o, 6, ≠, ♣] [4] [Y, 7, 2, △, ¥] [5] [B, β, 12, o, f] [6] [Z, 4, 3, ∇, j] [7] [G, α, 26, ∴, l] [8] [F, μ, 10, ♣, ∞] [9] [Q, 5, 8, ∴, ∞] [10] [W, π, 22, ●, ∞] [11] [Q, κ, 13, ●, -] [12] [Q, δ, 19, o, ⊗] [13] [I, ξ, 24, o, ♠] [14] [W, ω, 14, ■, ∞] [15] [S, ζ, 5, ●, ⊙] [16] [R, v, 1, ★, ←-] [3] G5 759 [12] G5 757	Generate from Passcode on file [1] [N, 0, 15, ✖, ✕] [2] [P, a, 11, ✨, -] [3] [Q, o, 6, ≠, ♣] [4] [Y, 7, 2, △, ¥] [5] [B, β, 12, o, f] [6] [Z, 4, 3, ∇, j] [7] [G, α, 26, ∴, l] [8] [F, μ, 10, ♣, ∞] [9] [Q, 5, 8, ∴, ∞] [10] [W, π, 22, ●, ∞] [11] [Q, κ, 13, ●, -] [12] [Q, δ, 19, o, ⊗] [13] [I, ξ, 24, o, ♠] [14] [W, ω, 14, ■, ∞] [15] [S, ζ, 5, ●, ⊙] [16] [R, v, 1, ★, ←-] [11] G5 755
Checking Key Tokens G5 721 [1] not found in [P a 11 ✨ -] G5 723 [2] found in [Y 7 2 △ ¥] G5 725 [3] found in [Z 4 3 ∇ j]	Receiver Side Result Key Tokens G5 765 G5 767 [5] [16]	Receiver Side Verify Key Tokens G5 721 [P a 11 ✨ -] K61 G5 723 [Y 7 2 △ ¥] K62 G5 725 [Z 4 3 ∇ j] K63
G5 771 Message Invalid !	G5 769 [7]	

Fig. 21 GATE_5 Encryption

☒ Demo
 GATE_5
 Delay 1 Seconds
 Clear Demo Id Passwords
 Start

GATE_4
 GATE_5
 Create User Id & Pn
 Graphic Login
 Text Login
 Encryption

Plain Message
 FYEO G5 700
 Sender Passcode 123
 G5 702
 Encrypt

[A] GATE system can be used for encryption, this is an example of using GATE_5 to encrypt a message.

Characters in a message are mixed in with other filler characters to hide the message.

Each character is attached with a table of 16 tokens representing a lock. Each character is also attached with a key consisting of several tokens.

The table and key of each character is generated with sender passcode.

Valid characters from the original message are attached with valid keys, filler characters are attached with invalid keys.

n			
17	u	25	ω
3	ψ	24	ϕ
21	τ	13	μ
7	η	18	ζ
12	δ	4	γ
9	ξ	21	τ
22	χ	22	ϕ
10	θ	19	α
5	π	9	ξ
4	ε	2	β
11	φ	22	χ
2	α	10	θ
1	β	1	α
16	ω	15	δ
14	ϕ	12	γ
13	μ	11	φ
12	λ	10	θ
11	κ	9	ξ
10	ι	8	η
9	h	7	g
8	g	6	f
7	f	5	e
6	e	4	d
5	d	3	c
4	c	2	b
3	b	1	a
2	a	0	z
1	z	25	ω

[B] As shown below, GATE_5 is used to decrypt a message.

After receiving all characters from a sender, it goes through each character to see if it's attached key can be authenticated against it's attached table of 16 tokens.

Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message.

Each unauthenticated character is not part of the original message and will be skipped. Each authenticated character is part of the original message and will be kept and combined to reveal the original message.

Key Tokens
 9
 ξ
 22
 χ
 21
 τ
 2
 β
 1
 α
 16
 ω
 15
 δ
 14
 ϕ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ
 13
 μ
 12
 γ
 11
 φ
 10
 θ
 9
 ξ
 8
 η
 7
 g
 6
 f
 5
 e
 4
 d
 3
 c
 2
 b
 1
 a
 z
 25
 ω
 24
 ϕ
 23
 μ
 22
 χ
 21
 τ
 20
 θ
 19
 α
 18
 ζ
 17
 u
 16
 ω
 15
 δ
 14
 ϕ