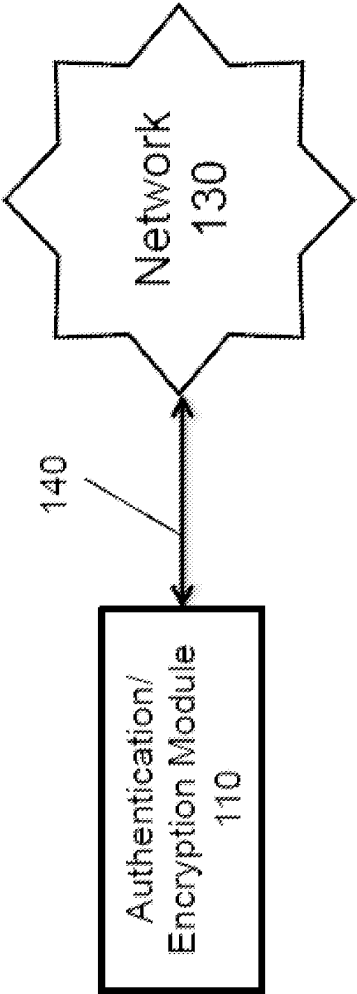


Fig. 1A

100



2/43

Fig. 1B

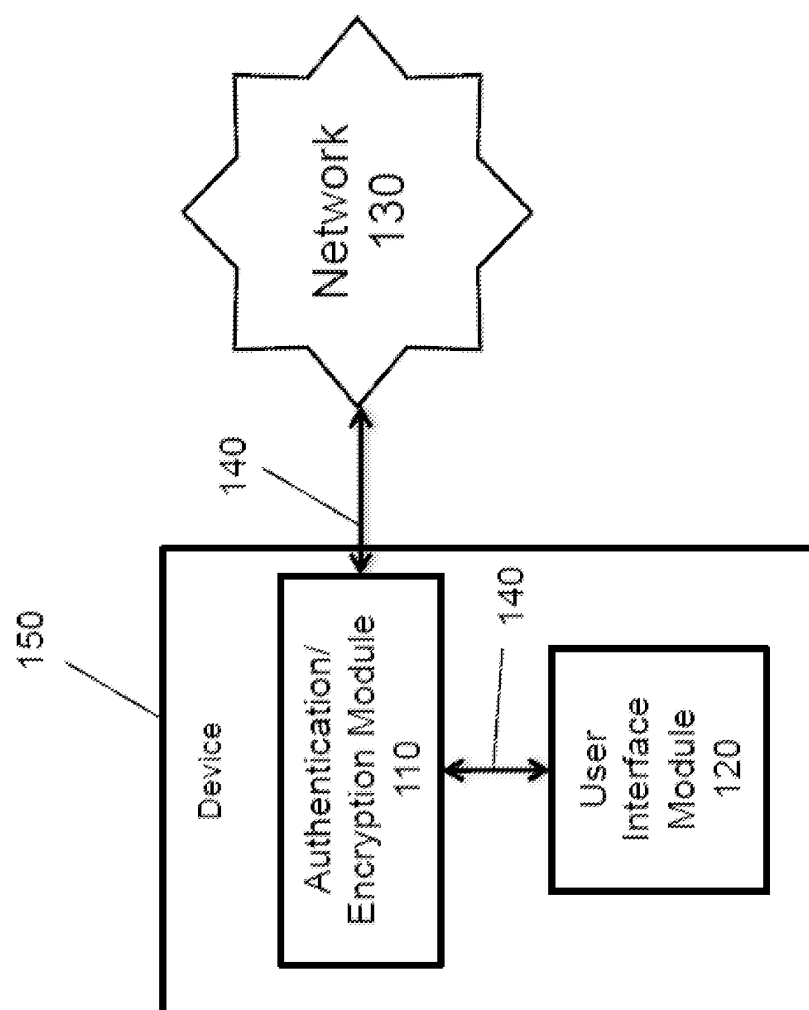


Fig. 1C

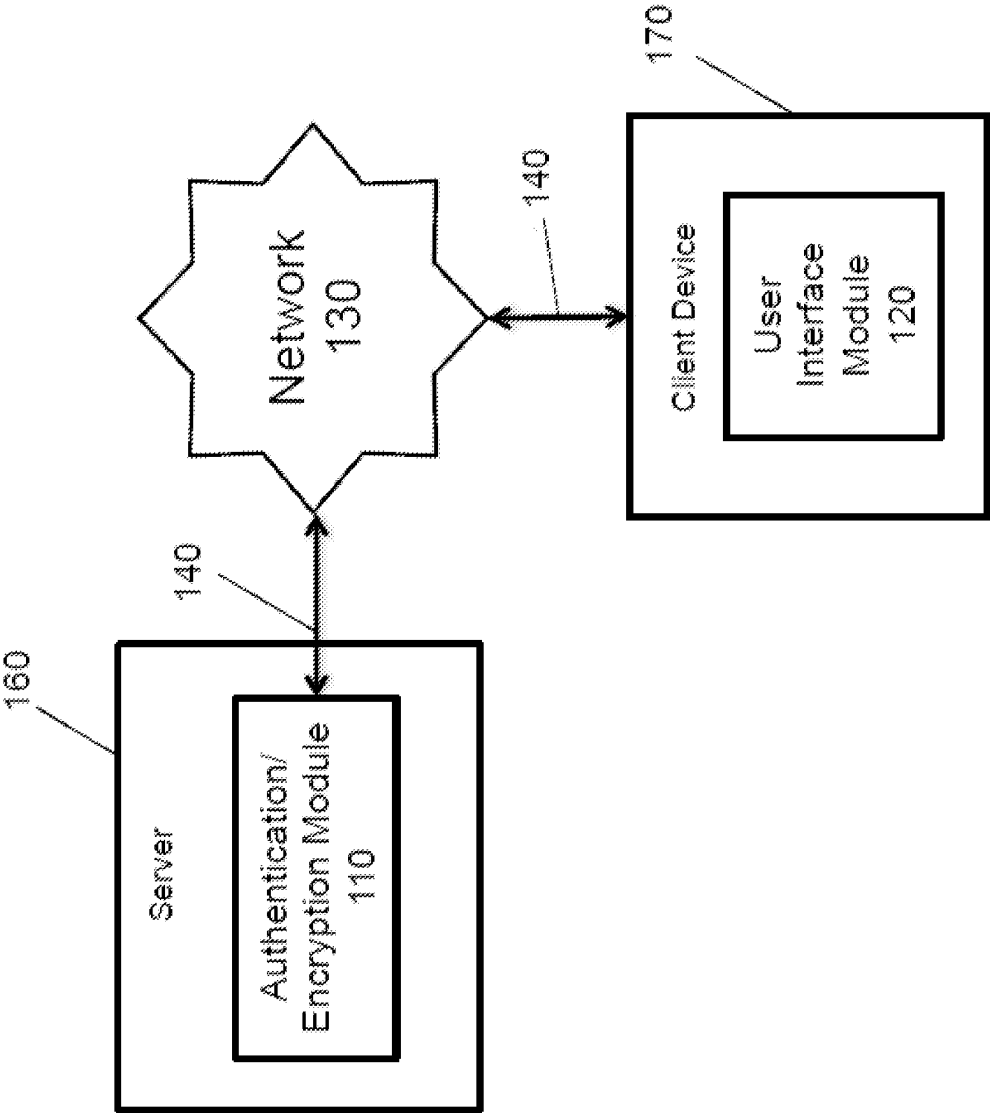


Fig. 1D

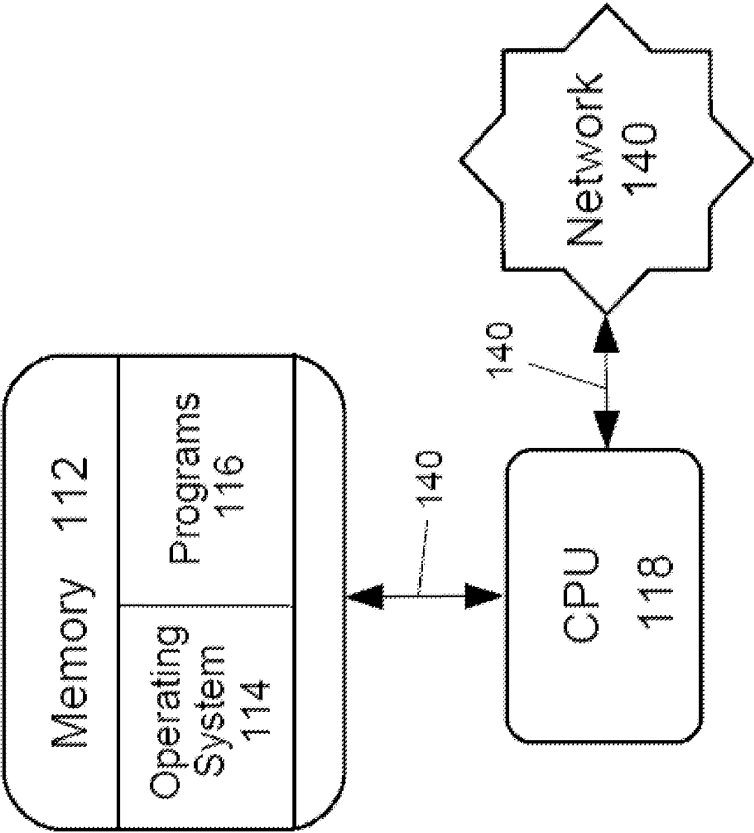


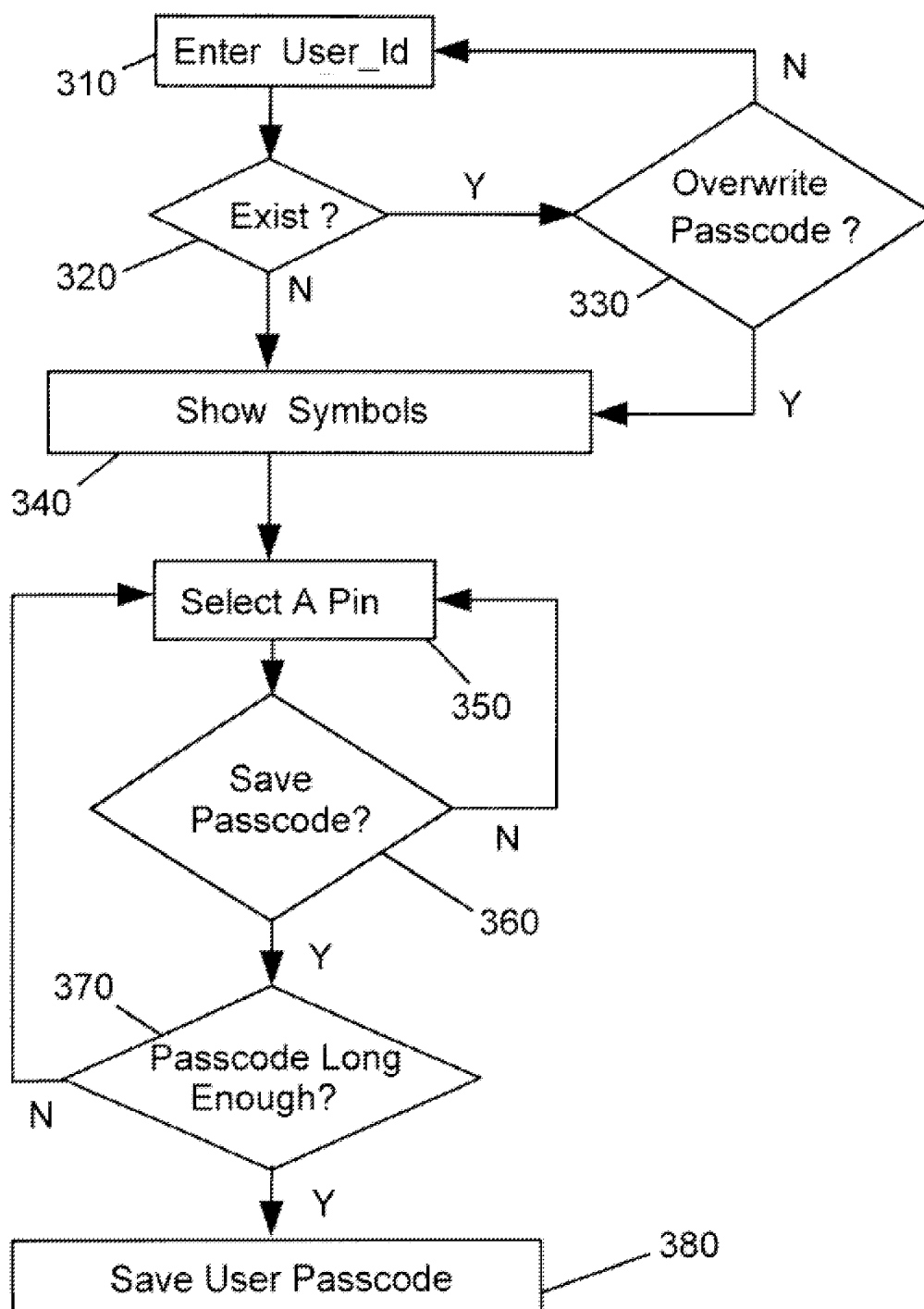
Fig. 2B

(GATE_5)

<u>1st Dimension</u>																									
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
<u>2nd Dimension</u>																									
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	ς
<u>3rd Dimension</u>																									
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
<u>4th Dimension</u>																									
○	●	△	▲	□	■	☆	★	♠	♣	♥	♠	♣	♥	♠	♣	♥	♠	♣	♥	♠	♣	♥	♠	♣	♥
<u>5th Dimension</u>																									
+	-	x	÷	←	→	↑	↓	↖	↗	↘	↙	↕	↔	↞	↠	↡	↢	↣	↤	↥	↦	↧	↨	↩	↪

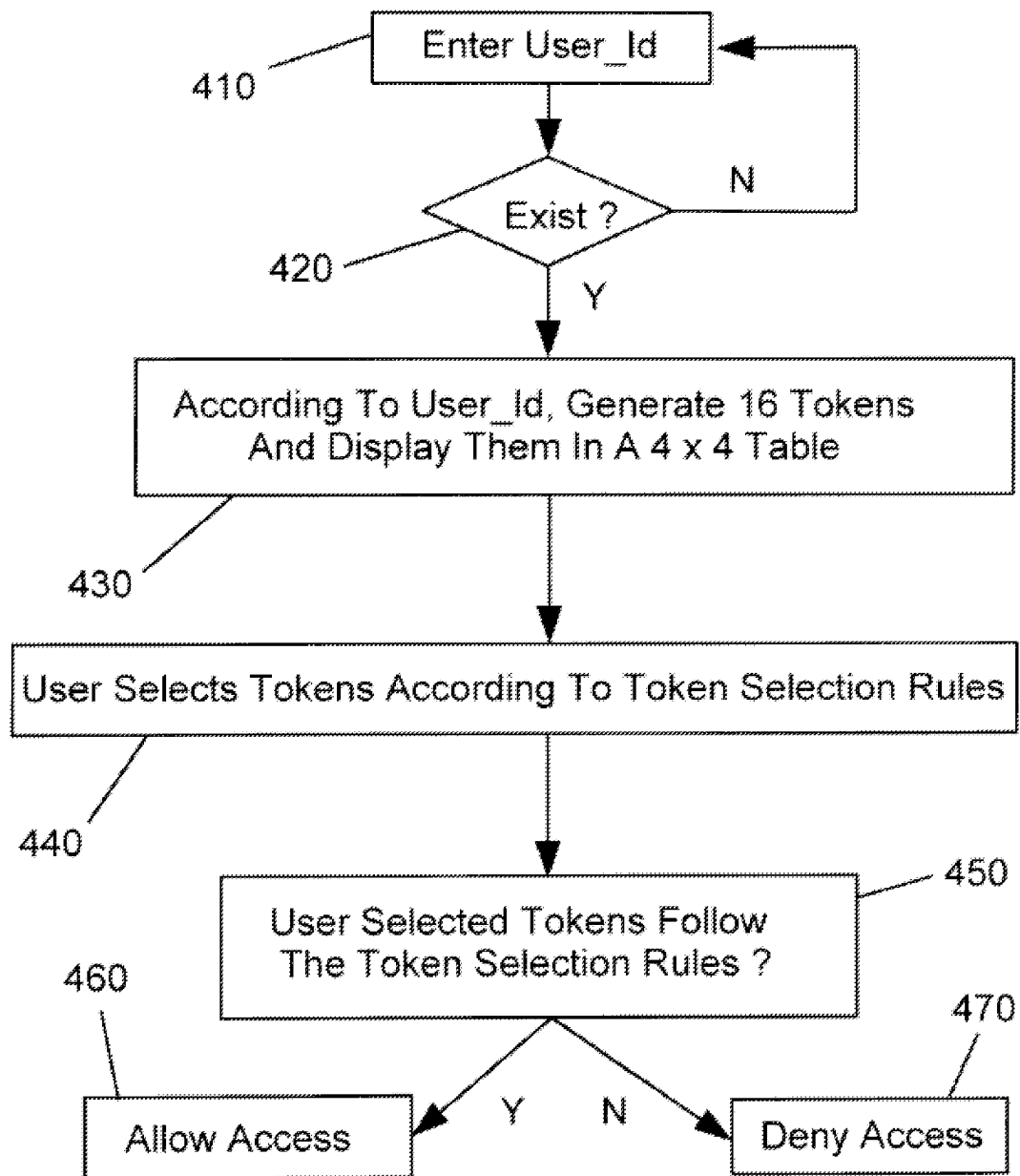
7/43

Fig. 3



8/43

Fig. 4



9/43

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 ☺ ♥ +), 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(Ⓐ, Ⓑ, ... ; ?)
V_3(☺, ★, ... ™, ™)
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] ⓐ ♥ 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 Ⓐ ★ ☹) |
| [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 ⓐ is the 1st pin and ☹ is the 4th pin in the following passcode : ⓐ ♥ 2 ☹

** passcode – similar to common sense password, but can contain symbols from the selections in each dimension [e.g. ⓐ ♥ 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.

10/43

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
♡	÷
1 st D	2 nd D
3 rd D	4 th D

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.

[2] User must follow the order of user passcode, e.g. ① ♡ 2 ③, 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 ③ ♡ ♪).

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 ③ ← ③). But the 4th user pin ③ is in token A1, user must pick it to be valid, so I'll pick A1 : (13 & ③ ③).

So in the end, user picked the following 4 tokens : (34 ③ ♡ ♪), (16 ③ ♡ ♪), (3 ③ ← ③) and (13 & ③ ③), 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 ♡ £).

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

	1	2	3	4
A	13 & ☺ ☒	34 ③ ♡ ♪	29 ③ → ☢	28 ! ✓ ÷
B	3 ③ ← ③	36 ③ = ☯	6 : ↓ π	19 ③ X +
C	20 ③ G ☒	26 ③ T €	17 ③ ♠ X	12 ③ ♥ ③
D	32 ③ ♠ \$	35 ③ ☆ ☢	16 ③ ● ☒	10 ③ °C ☒

①	σ
17	♡
●	£
1 st D	2 nd D
3 rd D	4 th D
5 th D	

11/43

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 ♠ ♦ ♣), (16 ♠ • ♠),
(3 ♠ ← ♠) 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 ♠ ♦ ♣) is valid.

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

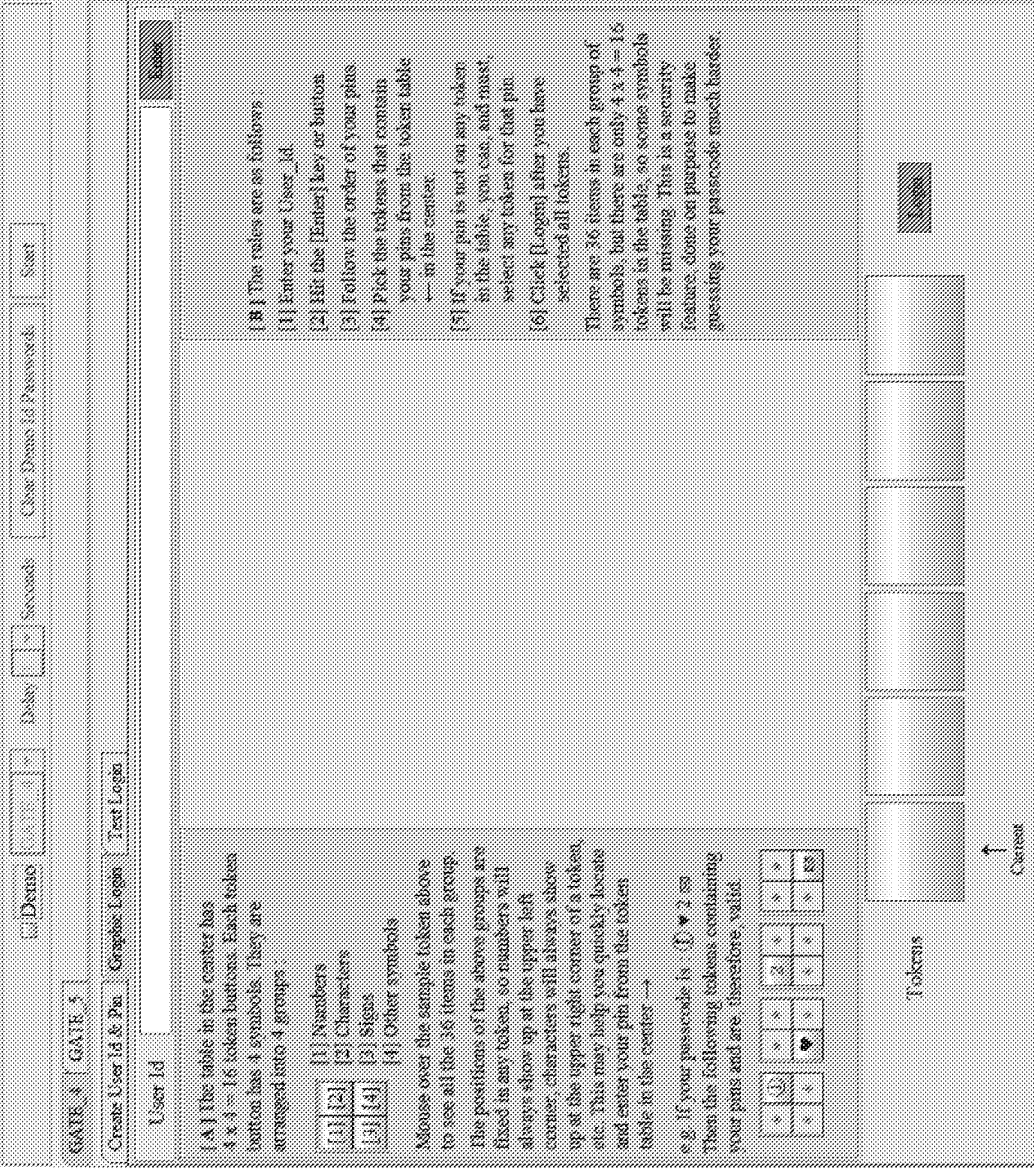
	1	2	3	4
A	13 &	34 ♠	29 Ⓢ	28 !
	Ⓢ Ⓢ	♦ ♣	→ Ⓢ	✓ ÷
B	3 Ⓢ	36 Ⓢ	6 :	19 Ⓢ
	← Ⓢ	= Ⓢ	↓ π	x +
C	20 Ⓢ	26 Ⓢ	17 Ⓢ	12 Ⓢ
	♣ ♣	F €	♠ x	♥ ♣
D	32 Ⓢ	35 Ⓢ	16 Ⓢ	10 Ⓢ
	♠ §	☆ ♣	● ♣	℃ ♣

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

Fig. 8A GATE_4 Create_Id

[illegible]

Fig. 9A GATE_4_Graphic_Login



16/43

Fig. 9C GATE_4_Graphic_Login

☐ Demo

GATE_4 | **GATE_5**

User Id: admin C4 307

Login Failed

C4 310

C4 321

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 $4 \times 4 = 16$ 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
A	13 ①	4 ②	26 ③	10 ④
B	19 ⑤	25 ⑥	14 ⑦	32 ⑧
C	31 ⑨	21 ⑩	18 ⑪	5 ⑫
D	8 ⑬	9 ⑭	27 ⑮	34 ⑯
	⑰	⑱	⑲	⑳

Tokens:

13 ①	32 ⑮	5 ⑩	4 ②	C4 366
⑰	⑱	⑲	⑳	C4 363
C4 351	C4 354	C4 357	C4 360	C4 371

Current

[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 ③ ④
 Then the following tokens containing your pins are, therefore, valid:

Fig. 9D GATE_4_Graphic_Login

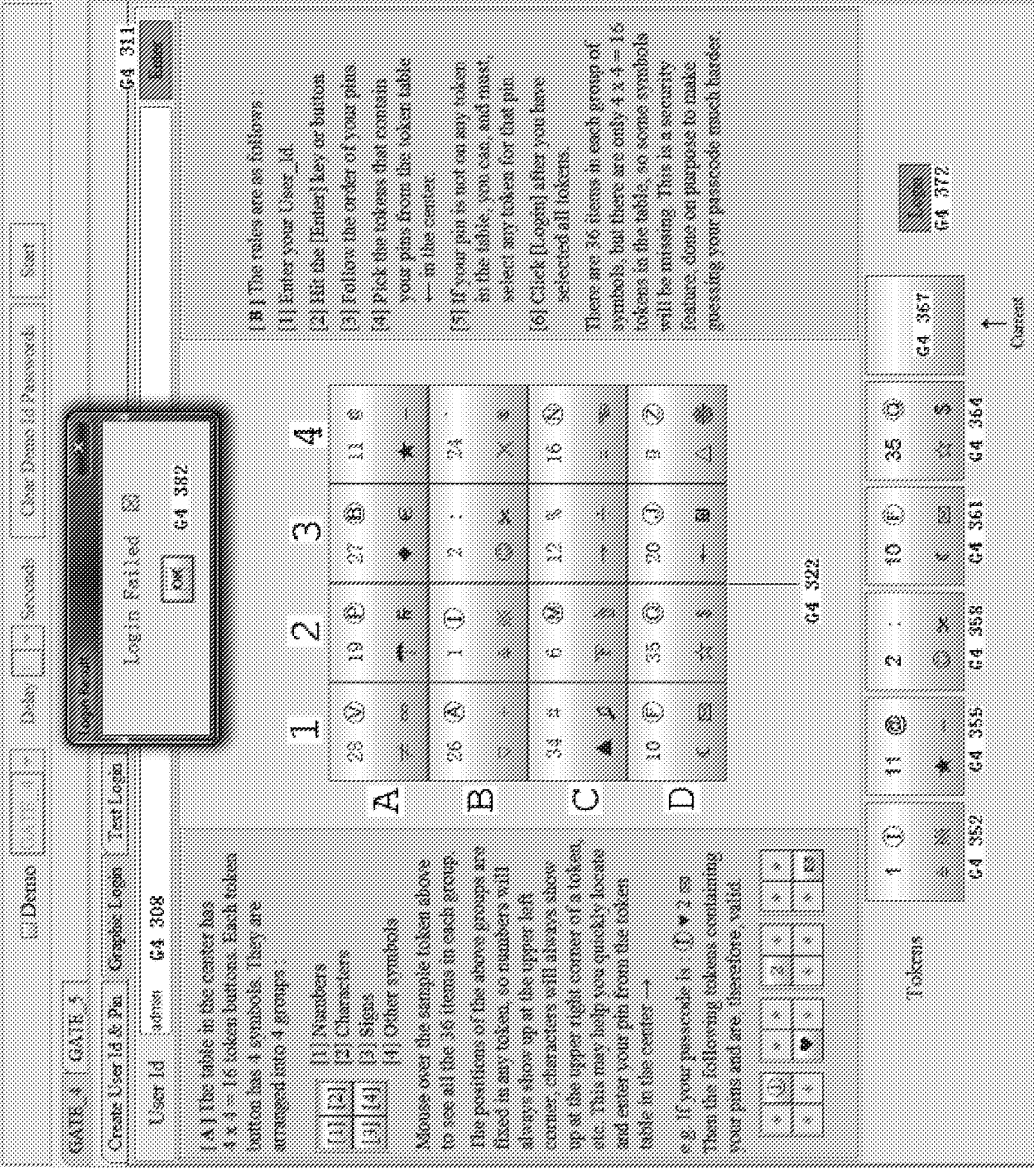
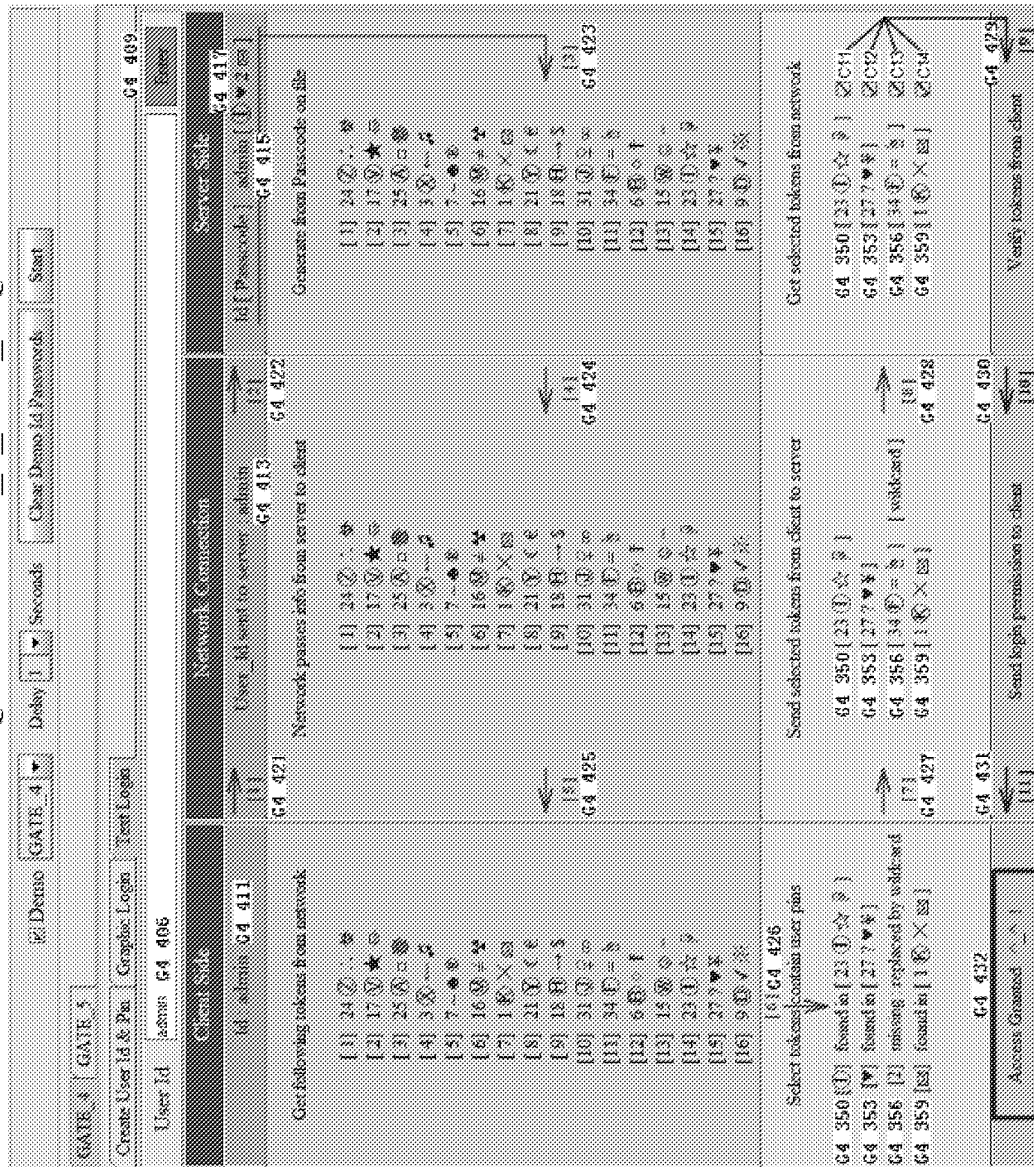


Fig. 10A GATE_4 Text_Login

☐ Demo GATE 4 GATE 5		Delay Seconds Clear Demo Id Passwords Start	
Create User Id & Pin Graphic Login Text Login		User Id Enter	
Client Side		Network Connection	
Id [Password]		User_id sent to server	
Server Side		Id [Password]	

19/43

Fig. 10B GATE_4 Text_Login



21/43

Fig. 10D GATE_4 Text_Login

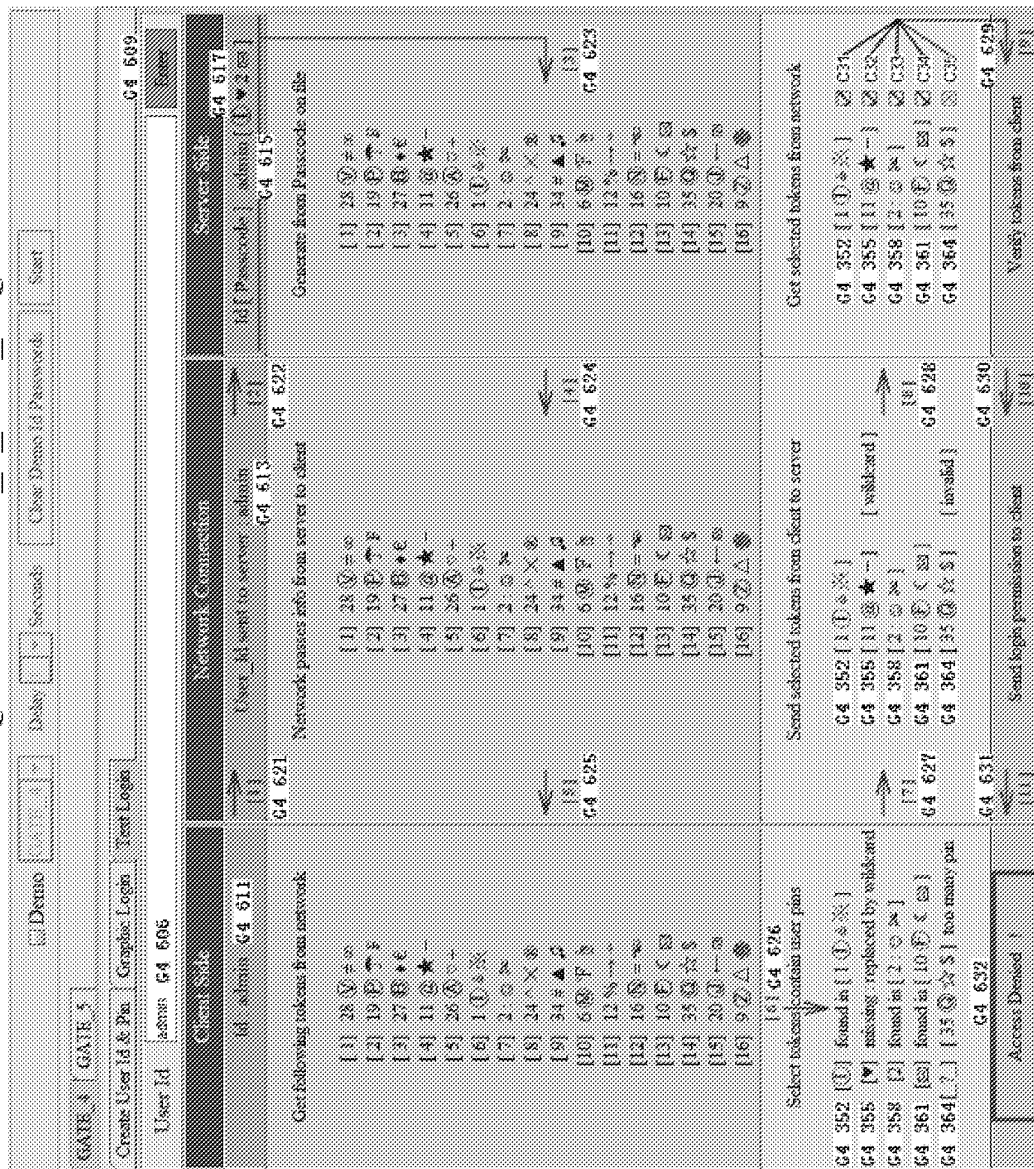


Fig. 11A GATE_5 Create Id

[illegible]

Fig. 12A GATE_5_Graphic_Login

GATE_4

GATE_5

Create User Id & Pin

Graphic Login

Test Login

Delay

Seconds

Start

Close Demo Id Password

User Id

Enter

[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 25 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 3
Then the following tokens containing your pins and are therefore, valid

1

2

3

4

5

1

2

3

4

5

1

2

3

4

5

1

2

3

4

5

1

2

3

4

5

[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.

Tokens

Current

Login

Fig. 12B GATE_5_Graphic_Login

[illegible]

26/43

Fig. 12C GATE_5_Graphic_Login

GATE_4 **GATE_5** ☐ Demo

Create User Id & Pin **Graphic Login** **Test Login**

User Id admin GS 307

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

Mouse over the sample token above to see all the 25 items in each group.

[B] 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.

[C] If your passcode is: 1 2 3 4
 Then the following tokens containing your pins and are therefore, valid

[D] If your passcode is: 1 2 3 4

[E] 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
- (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 25 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
12	23	34	45
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23
34	45	56	67
78	89	90	01
23	34	45	56
67	78	89	90
01	12	23	34
56	67	78	89
90	01	12	23

27/43

Fig. 12D GATE_5_Graphic_Login

Demo GATE_5 Delay 1 Seconds Clear Demo Id Passwords Start

Create User Id & Pin Graphic Login Test Login

User Id admin GS 308

Login Failed OK GS 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] English alphabet
 [2] Greek characters
 [3] Numbers
 [4] Symbols
 [5] Other signs

Mouse over the sample token above to see all the 25 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 3 Then the following tokens containing your pins and are therefore, valid:

Tokens

GS 352 GS 355 GS 358 GS 361 GS 364 GS 367 GS 372

GS 322

GS 311 Enter

[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	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22
ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22
ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22
ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22	ⓐ 15 ⓑ 4 ⓓ 6 ⓔ 22

Fig. 13A GATE_5_Text_Login

[illegible]

29/43

Fig. 13B GATE_5_Text_Login

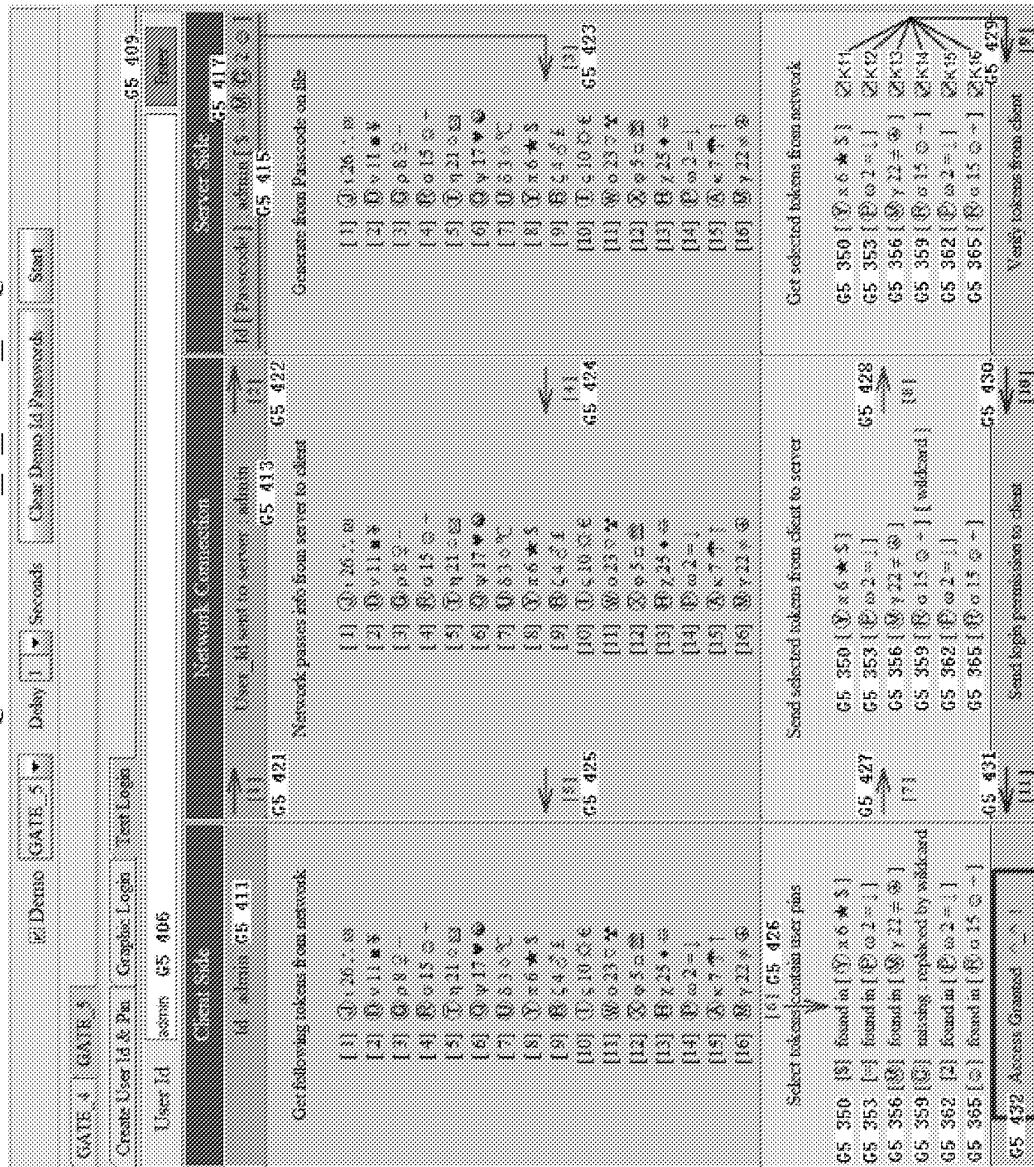


Fig. 14 GATE_4_Encryption

[E] Demo		GATE_4	GATE_5	Delay 1	Seconds	Clear Demo Id Passwords	Start
Create User Id & Pin	Graphic Login	Test Login	Encryption				
Plain Message		secret		G4 700		Sender Passcode 123	
[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.				G4 702 Encrypt			
Key Tokens				 Check			
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.		Receiver passcode is used in the authentication process. Receiver passcode should be the same as sender passcode to recover the original message.		G4 703 Decrypt			
Encrypted Message Sent / Received		Current 		G4 704 Receiver Passcode 123 Decrypt			

33/43

Fig. 15A GATE_4_Encryption

GATE_4

GATE_5

Demo

GATE_5

Delay 1

Seconds

Clear

Check Successful

G4 730

Create User Id & Pin

Graphic Login

Test Login

Encryption

G4 700

Plain Message

secret

G4 700

Sender Passcode

123

G4 702

Encrypt

[1.1] 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.

[1.8] 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

G4 720

G4 722

G4 724

1

2

4

1

X

@

O

A

Check

G4 726

G4 712			
26 ①	26 ②	1 4	9 8
☆	★	∞	♣
7 :	21 ③	5 ④	23 ⑤
0 ⑥	♣	†	0 ⑦
4 :	30 *	17 ⑧	2 ⑨
○ A	✓	♣	+
13 ?	27 ⑩	22 ⑪	21 ⑫
■ 0	♣	-	×
G4 714			

Encrypted Message: G4 710

Sent / Received

Current

↑

G4 704

L	Q	V	N	E	M	I	H	Q	I
①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩

G4 706

Decrypted Message

secret

G4 708

Receiver Passcode

123

G4 706

Decrypt

34/43

Fig. 15B GATE_4_Encryption

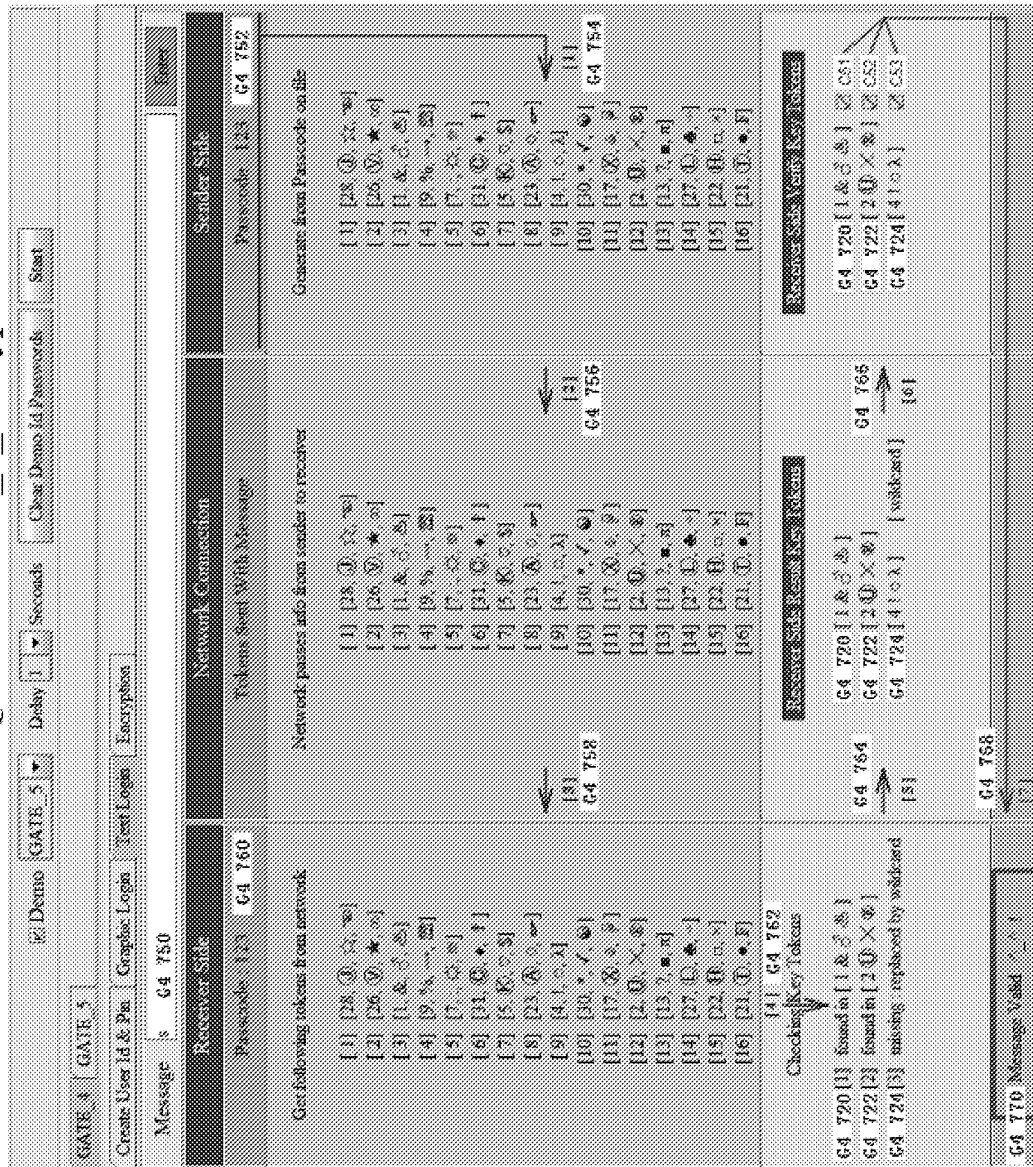


Fig. 16A GATE_4_Encryption

[X] Demo GATE_4 GATE_5 Delay I Seconds Clear D																			
Create User Id & Pin Graphic Login Test Login Encryption																			
Plain Message secret G4 700 [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.	Key Tokens ↑ Current <table border="1" style="width: 100%; text-align: center; font-size: small;"> <thead> <tr> <th>G4 721</th> <th>G4 723</th> <th>G4 725</th> <th>G4 727</th> </tr> </thead> <tbody> <tr> <td>19 Ⓚ ▲ ●</td> <td>24 Ⓢ ○ ☼</td> <td>36 ? ↑ β</td> <td>19 Ⓚ ▲ ●</td> </tr> </tbody> </table> <table border="1" style="width: 100%; text-align: center; font-size: x-small;"> <thead> <tr> <th>L</th> <th>G4 713</th> </tr> </thead> <tbody> <tr> <td>30 Ⓚ →</td> <td>18 ~ 3 ⓐ 13 %</td> </tr> <tr> <td>25 Ⓢ ▲</td> <td>24 Ⓢ ○ ☼ 35 Ⓢ 36 ?</td> </tr> <tr> <td>33 Ⓚ △</td> <td>29 ⊗ □ ♠ 19 Ⓚ ▲ ● 34 Ⓢ ☼</td> </tr> <tr> <td>4 Ⓢ ☆</td> <td>26 Ⓢ ★ + 9 Ⓢ ▲ ● 10 Ⓢ ○ ☼</td> </tr> </tbody> </table>	G4 721	G4 723	G4 725	G4 727	19 Ⓚ ▲ ●	24 Ⓢ ○ ☼	36 ? ↑ β	19 Ⓚ ▲ ●	L	G4 713	30 Ⓚ →	18 ~ 3 ⓐ 13 %	25 Ⓢ ▲	24 Ⓢ ○ ☼ 35 Ⓢ 36 ?	33 Ⓚ △	29 ⊗ □ ♠ 19 Ⓚ ▲ ● 34 Ⓢ ☼	4 Ⓢ ☆	26 Ⓢ ★ + 9 Ⓢ ▲ ● 10 Ⓢ ○ ☼
G4 721	G4 723	G4 725	G4 727																
19 Ⓚ ▲ ●	24 Ⓢ ○ ☼	36 ? ↑ β	19 Ⓚ ▲ ●																
L	G4 713																		
30 Ⓚ →	18 ~ 3 ⓐ 13 %																		
25 Ⓢ ▲	24 Ⓢ ○ ☼ 35 Ⓢ 36 ?																		
33 Ⓚ △	29 ⊗ □ ♠ 19 Ⓚ ▲ ● 34 Ⓢ ☼																		
4 Ⓢ ☆	26 Ⓢ ★ + 9 Ⓢ ▲ ● 10 Ⓢ ○ ☼																		
Check Failed [X] [OK] G4 731																			
Sender Passcode 123 G4 702 [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 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.																			
Encrypted Message Sent / Received secret G4 708 secret G4 706																			
Decrypted Message Receiver Passcode: 123 Decrypt																			

36/43

Fig. 16B GATE_4_Encryption

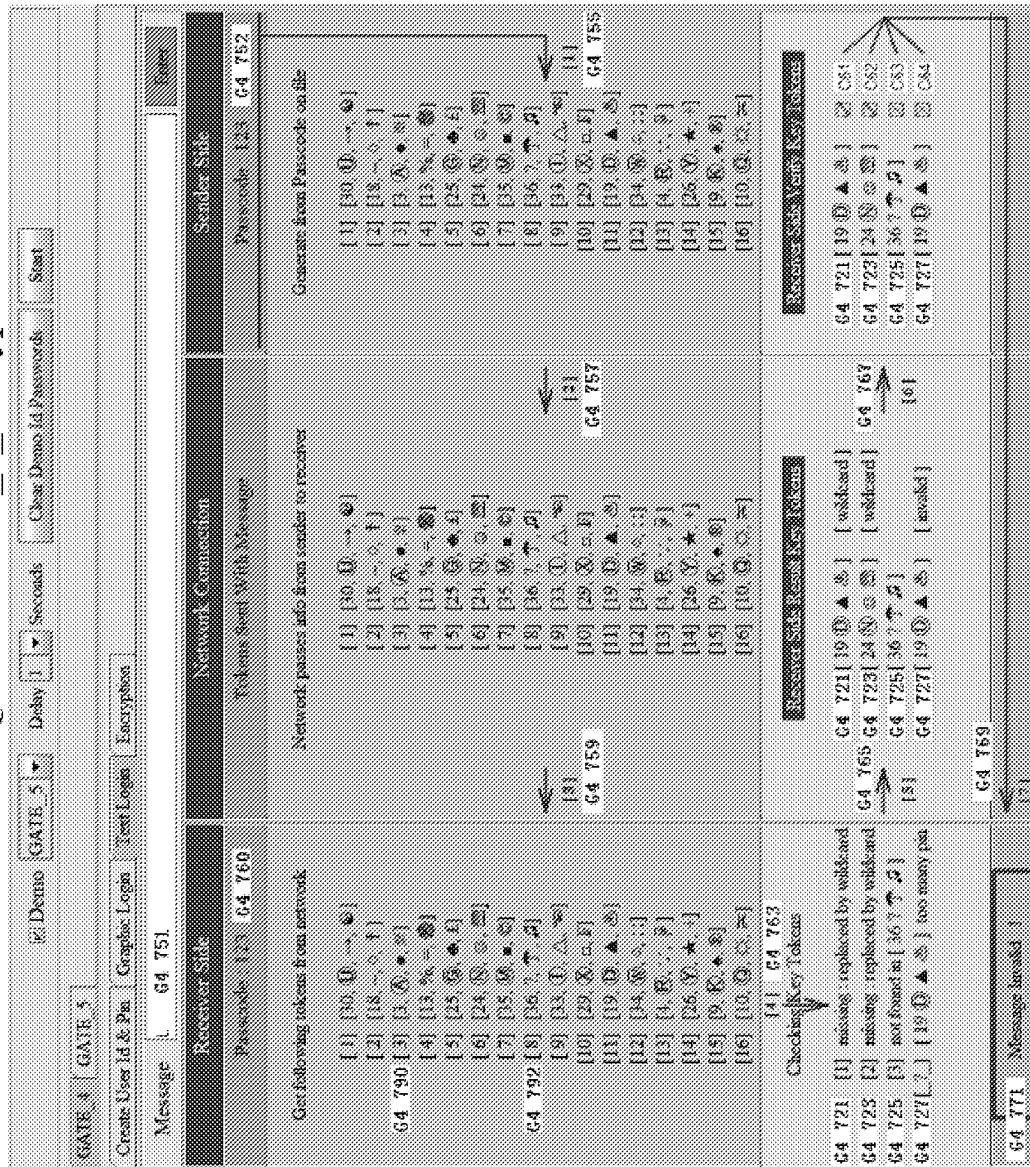


Fig. 17 GATE_4_Encryption

GATE_4	GATE_5	[E] Demo	GATE_5 ▼	Delay I ▾ Seconds	Clear Demo Id Passwords	Start
Create User Id & Pin	Graphic Login	Test Logon	Encryption			
Plain Message			secret G# 700			
[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.						
Key Tokens ↑ Current 35 [H] 32 : 3 [O] : Δ ⊗			CLEAR			
Sender Passcode 123			G# 702			
[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 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.						
Encrypted Message Sent Received						
Decrypted Message						
Receiver Passcode 567			G# 701			

Fig. 18 GATE_5_Encryption

[illegible]

Fig. 19A GATE_5_Encryption

GATE 4	GATE 5	Demo	GATE 5 ▾	Delay 1 ▾ Seconds	Clear D
Create User Id & Pn		Graphic Login		Text Login	
Plan Message		FYEO GS 700			

[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 its attached table of 16 tokens.

Receiver passcode is used in the authentication process. Receiver passcodes 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.

Sender Passcode	GS 702	Check Successful ✓	GS 730	Decrypt	GS 703
		OK			

Plan Message	FYEO GS 700	Current	GS 704	GS 720	GS 722	GS 724	GS 726

Key Tokens	GS 720	GS 722	GS 724	GS 726
	S 1 Q B	K 2 C °C	N Y 3 +	Check GS 726

Encrypted Message Sent / Received	FYEO GS 708	Receiver Passcode	GS 706

40/43

Fig. 19B GATE_5_Encryption

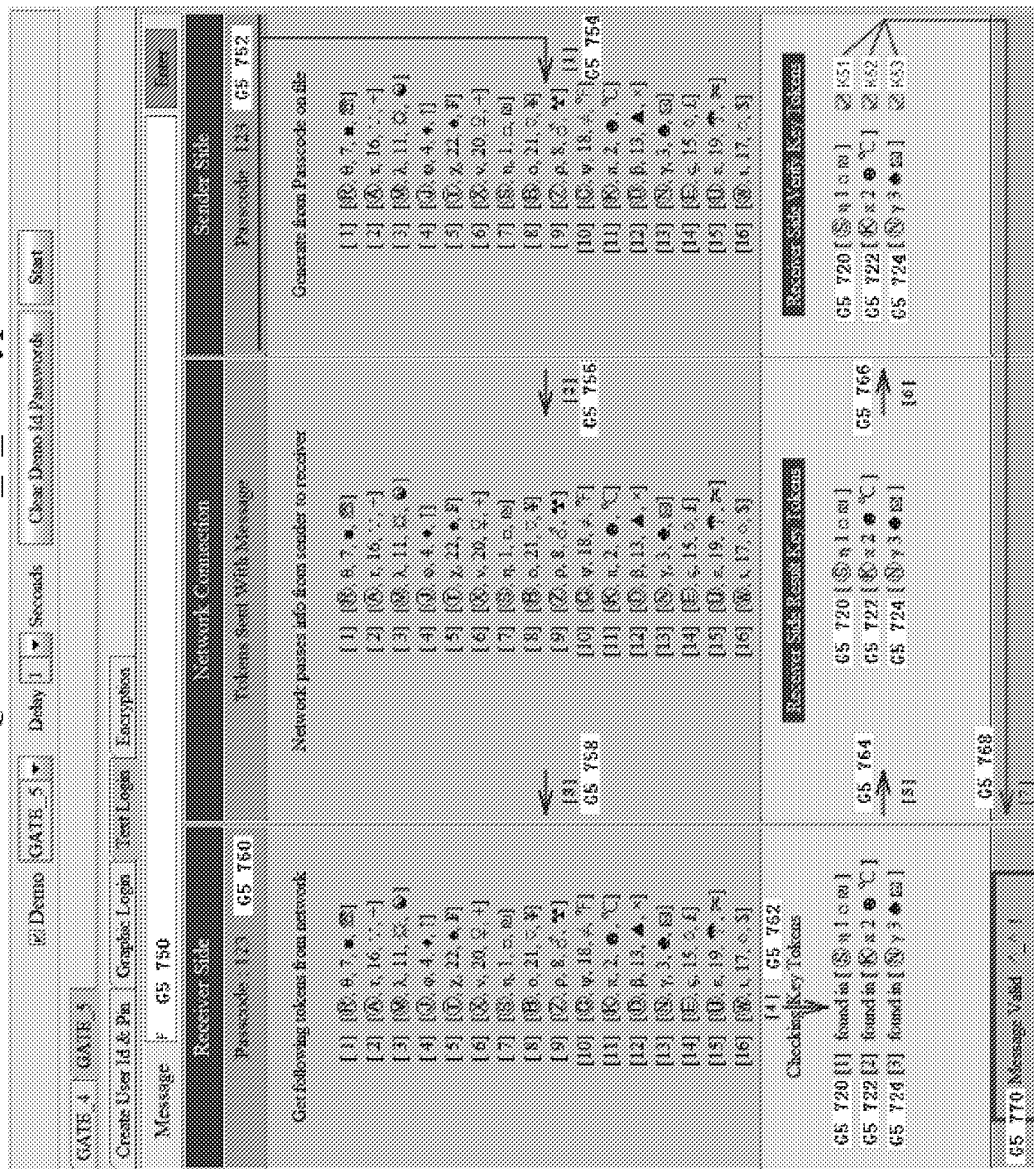


Fig. 20A GATE_5_Encryption

[illegible]

42/43

Fig. 20B GATE_5_Encryption

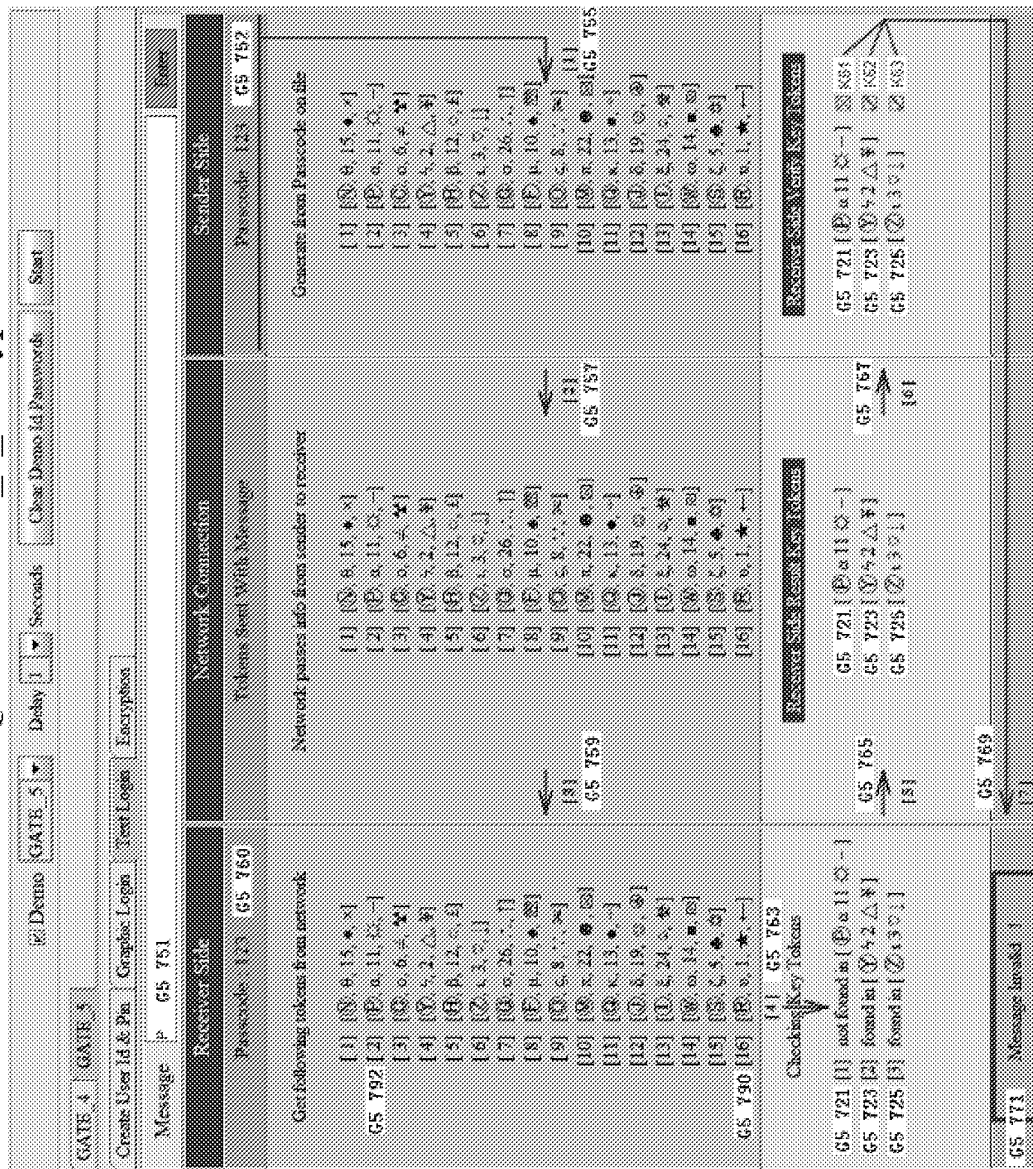


Fig. 21 GATE_5_Encryption

GATE 4 GATE 5		Demo GATE 5 Delay I Seconds Clear Demo Id Passwords Start																									
Create User Id & Pw Graphic Login Test Login Encryption																											
Plain Message FREQ GS 700		Sender Passcode : 23 GS 702 Decrypt																									
[X] 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.		[S] 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 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 <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>W</td><td>9</td><td>X</td><td>T</td></tr> <tr> <td>O</td><td>E</td><td>C</td><td>A</td></tr> </table> Current <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td> </td><td> </td><td> </td><td> </td></tr> <tr> <td> </td><td> </td><td> </td><td> </td></tr> </table>		W	9	X	T	O	E	C	A									<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td> </td><td> </td><td> </td><td> </td></tr> <tr> <td> </td><td> </td><td> </td><td> </td></tr> </table>									
W	9	X	T																								
O	E	C	A																								
Encrypted Message Sent / Received		GS 705 Decrypt																									