

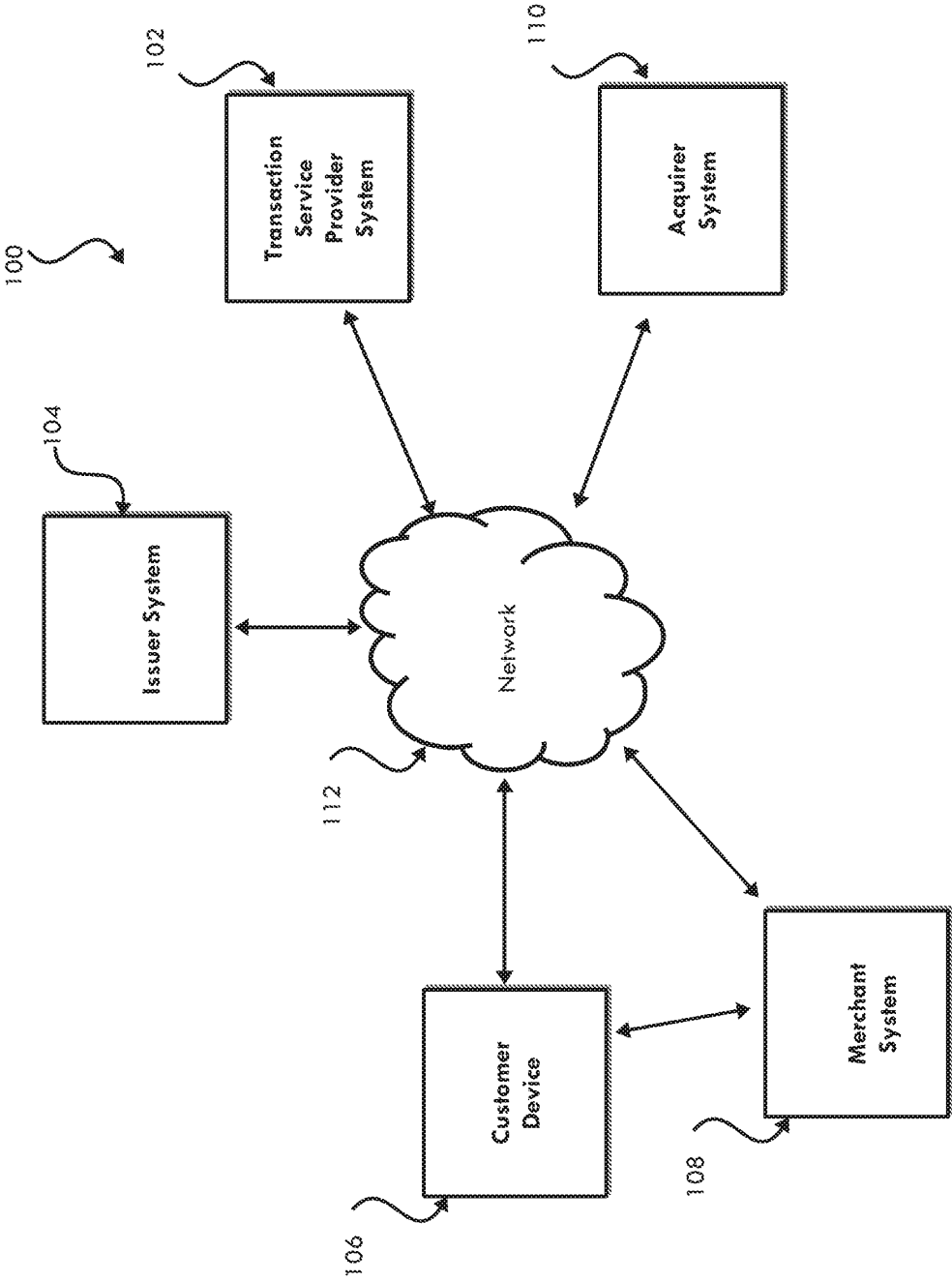


FIG. 1

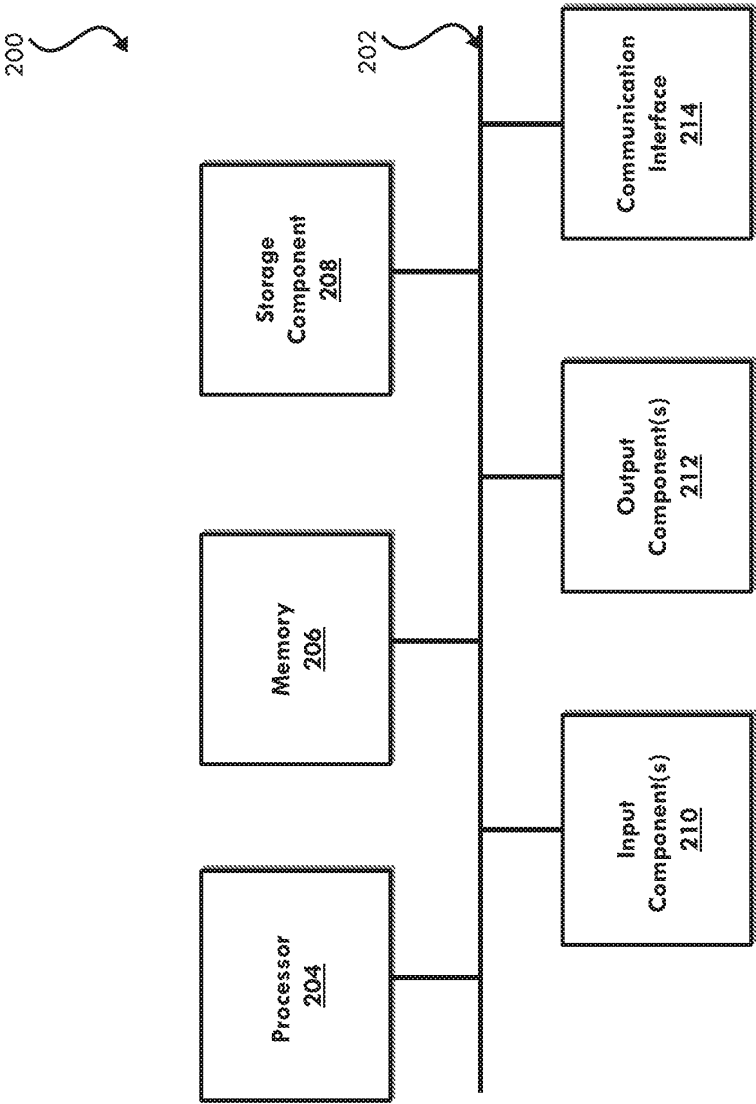


FIG. 2

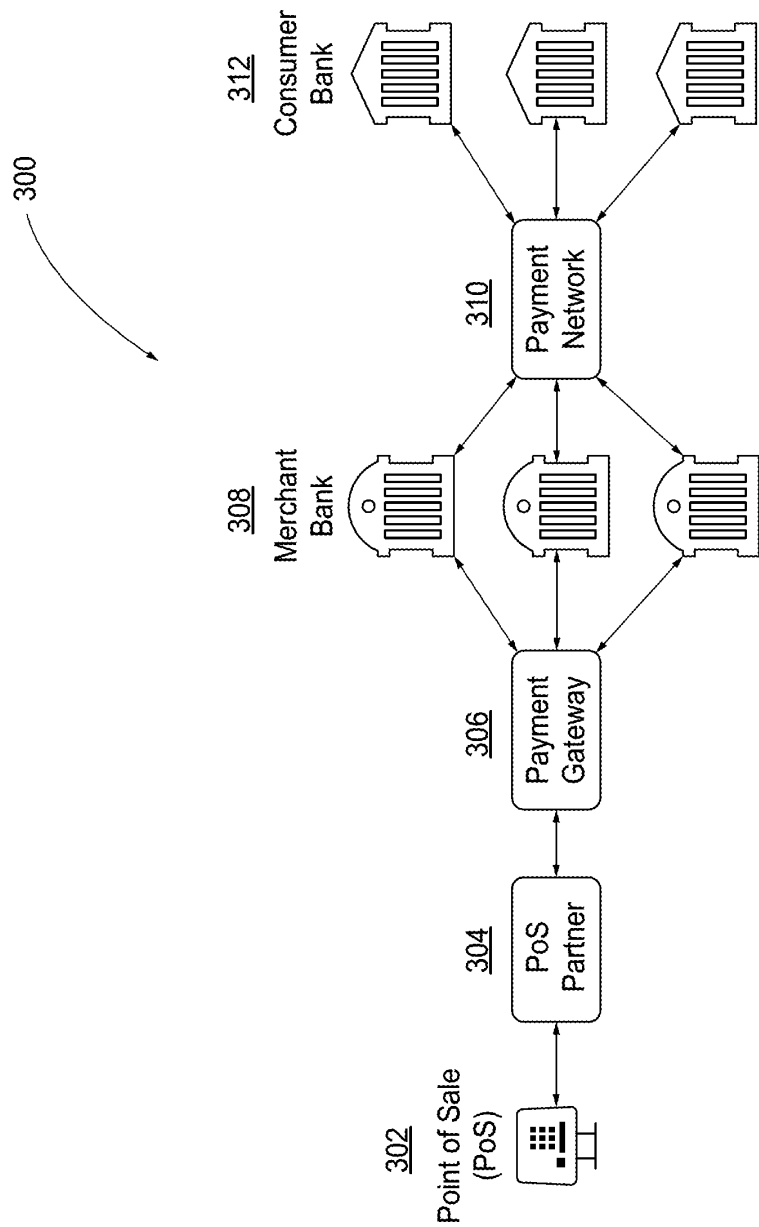


FIG. 3

ISO PIN-BLOCK ENCODINGS (AS DEFINED IN ISO 9564-1)

A PIN-block encoding scheme $ES = (\text{encode}, \text{decode})$ take two inputs a PIN and a PAN, and outputs a PIN Block (PB).

The encoded PB is the XOR of two fields, a plaintext PIN field and an account number field.

Patient PIN field:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	F
	N	P	P	P	P	P/F	P/F	P/F	P/F	F

where each 4-bit field is as follows:

- C – Control field: takes value 0000 (zero) for ISO-0 or 0011 (3) for ISO-3;
- N – PIN length: takes a value from 0100 (4) to 1100 (12);
- P – PIN digit: takes a values from 0000 (zero) to 1001 (9);
- P/F – PIN/Fill digit: designation of these fields is determined by the PIN length field;
- F – Fill digit: takes value 1111 (15) for ISO-0 or a random value between 1010 (10) and 1111 (15) for ISO-3.

Account Number field:

0	0	0	0	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇	A ₈	A ₉	A ₁₀	A ₁₁	A ₁₂
---	---	---	---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------	-----------------	-----------------

where each 4-bit field is as follows:

- 0 – Pad digit: takes value 0000 (zero);
- $A_1 \dots A_{12}$ – Account number: the 12 rightmost digits of the PAN excluding the check digit (values are from 0000 (0) to 1001 (9)).

FIG. 4

5/21

500

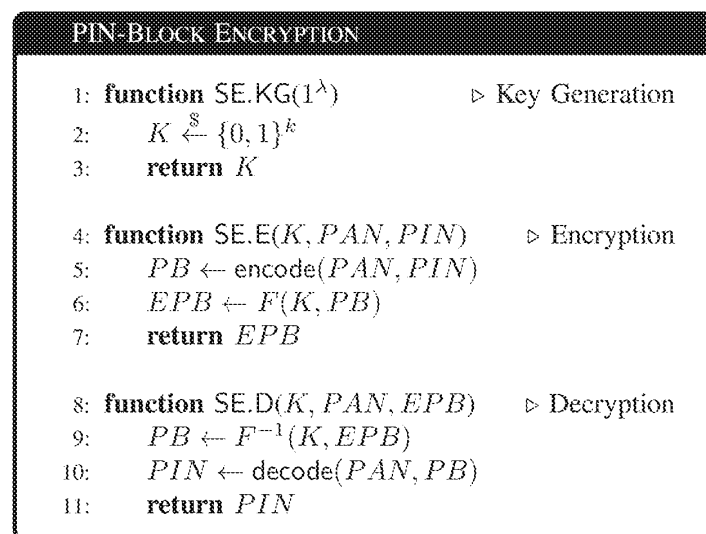


FIG. 5

6/21

600

PRE HRA-CCA GAME ORACLES

```

1: function  $\mathcal{O}_{\text{RKG}}^{\text{UNI}}(i, j)$   $\triangleright$  Unidirectional Re-key
2:   if  $i \in H, j \in C$  then return  $\perp$ 
3:   return  $\text{PRE.RKG}(sk_i, pk_i, pk_j)$ 

4: function  $\mathcal{O}_{\text{RKG}}^{\text{BI}}(i, j)$   $\triangleright$  Bidirectional Re-key
5:   if  $\{i, j\} \not\subseteq H$  or  $\{i, j\} \not\subseteq C$  then return  $\perp$ 
6:   return  $\text{PRE.RKG}(sk_i, pk_i, sk_j, pk_j)$ 

7: function  $\mathcal{O}_{\text{Chall}}(i, m_0, m_1)$   $\triangleright$  Challenge
8:   if  $i \notin H$  and  $|m_0| \neq |m_1|$  then return  $\perp$ 
9:   return  $\text{PRE.E}(pk_i, m_b)$ 

10: function  $\mathcal{O}_{\text{E}}(i, m)$   $\triangleright$  Encryption
11:   return  $\text{PRE.E}(pk_i, m)$ 

12: function  $\mathcal{O}_{\text{RE}}(i, j, c)$   $\triangleright$  Re-encryption
13:   if  $c \notin D^* \cup D$ , or  $c \in D^*$  and  $j \in C$  then
14:     return  $\perp$ 
15:   return  $\text{PRE.RE}(rk_{i \rightarrow j}, c)$ 

16: function  $\mathcal{O}_{\text{D}}(j, c)$   $\triangleright$  Decryption
17:   if  $c \in D^*$  then return  $\perp$ 
18:   return  $\text{PRE.D}(sk_j, c)$ 

```

FIG. 6

SUBSTITUTE SHEET (RULE 26)

7/21

700

PPA-HRCCA GAME ORACLES

```

1: function  $\mathcal{O}_{\text{Enroll}}^{\text{UNI}}(i, j)$  ▷ Unidirectional Enroll
2:   if  $i \in H, j \in C$  then return  $\perp$ 
3:   return PPA.Enroll( $sk_i, pk_i, pk_j$ )

4: function  $\mathcal{O}_{\text{Enroll}}^{\text{BI}}(i, j)$  ▷ Bidirectional Enroll
5:   if  $\{i, j\} \not\subseteq H$  or  $\{i, j\} \not\subseteq C$  then return  $\perp$ 
6:   return PPA.Enroll( $sk_i, pk_i, sk_j, pk_j$ )

7: function  $\mathcal{O}_{\text{Chall}}(i, a, m_0, m_1)$  ▷ Challenge
8:   if  $i \notin H$  and  $|m_0| \neq |m_1|$  then return  $\perp$ 
9:   return PPA.Send( $pk_i, m_b$ )

10: function  $\mathcal{O}_{\text{Send}}(i, m)$  ▷ Send
11:   return PPA.Send( $pk_i, m$ )

12: function  $\mathcal{O}_{\text{Relay}}(i, j, c)$  ▷ Relay
13:   if  $c \notin D^* \cup D$ , or  $c \in D^*$  and  $j \in C$  then
14:     return  $\perp$ 
15:   return PPA.Relay( $rk_{i \rightarrow j}, c$ )

16: function  $\mathcal{O}_{\text{Verify}}(j, c)$  ▷ Verify
17:   if  $c \in D^*$  then return  $\perp$ 
18:   return PPA.Verify( $sk_j, c$ )

```

FIG. 7

8/21

800

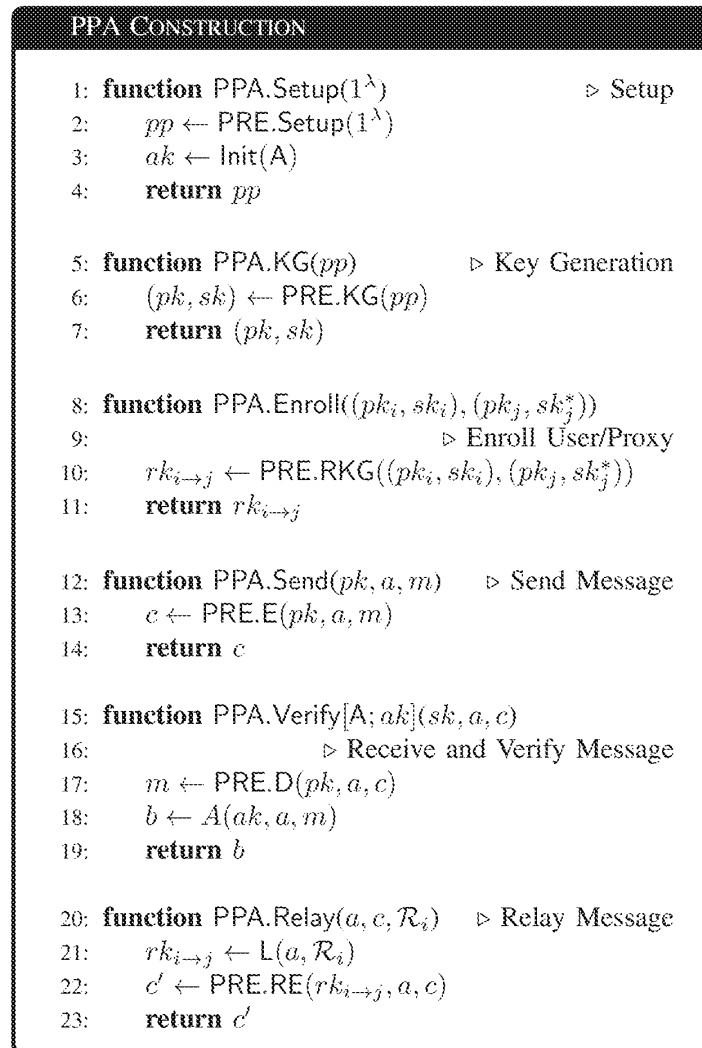


FIG. 8

9/21

900

FHYBRID-PRE CONSTRUCTION

```

1: function PRE.Setup( $1^\lambda$ )                                ▷ Setup
2:   Choose  $\mathbb{G}$  of prime order  $q$  with generator  $g$ 
3:   return  $pp = (g, q)$ 

4: function PRE.KG( $pp$ )                                    ▷ Key Generation
5:    $sk \xleftarrow{\$} \mathbb{Z}_q^*$ ;  $pk \leftarrow g^{sk}$ 
6:   return  $(pk, sk)$ 

7: function PRE.RKG( $(pk_i, sk_i), (pk_j, sk_j)$ )
8:                                     ▷ ReKey Generation
9:    $rk_{i \rightarrow j} \leftarrow sk_i / sk_j$ 
10:  return  $rk_{i \rightarrow j}$ 

11: function PRE.E( $pk, a, m$ )                                ▷ Encryption
12:    $r \xleftarrow{\$} \mathbb{Z}_q^*$ ;  $c_1 \leftarrow g^r$ 
13:    $K = F(pk^r)$ ;  $c_2 \leftarrow \text{SE.E}(K, a, m)$ 
14:   return  $(c_1, c_2)$ 

15: function PRE.D( $sk, a, c$ )                                ▷ Decryption
16:   Parse  $c$  as  $(c_1, c_2)$ 
17:    $K = F(c_1^{sk})$ ;  $m \leftarrow \text{SE.D}(K, a, c)$ 
18:   return  $m$ 

19: function PRE.RE( $rk_{i \rightarrow j}, a, c$ )                    ▷ ReEncryption
20:   Parse  $c$  as  $(c_1, c_2)$ 
21:    $c'_1 \leftarrow c_1^{rk_{i \rightarrow j}}$ 
22:   return  $(c'_1, c_2)$ 

```

FIG. 9

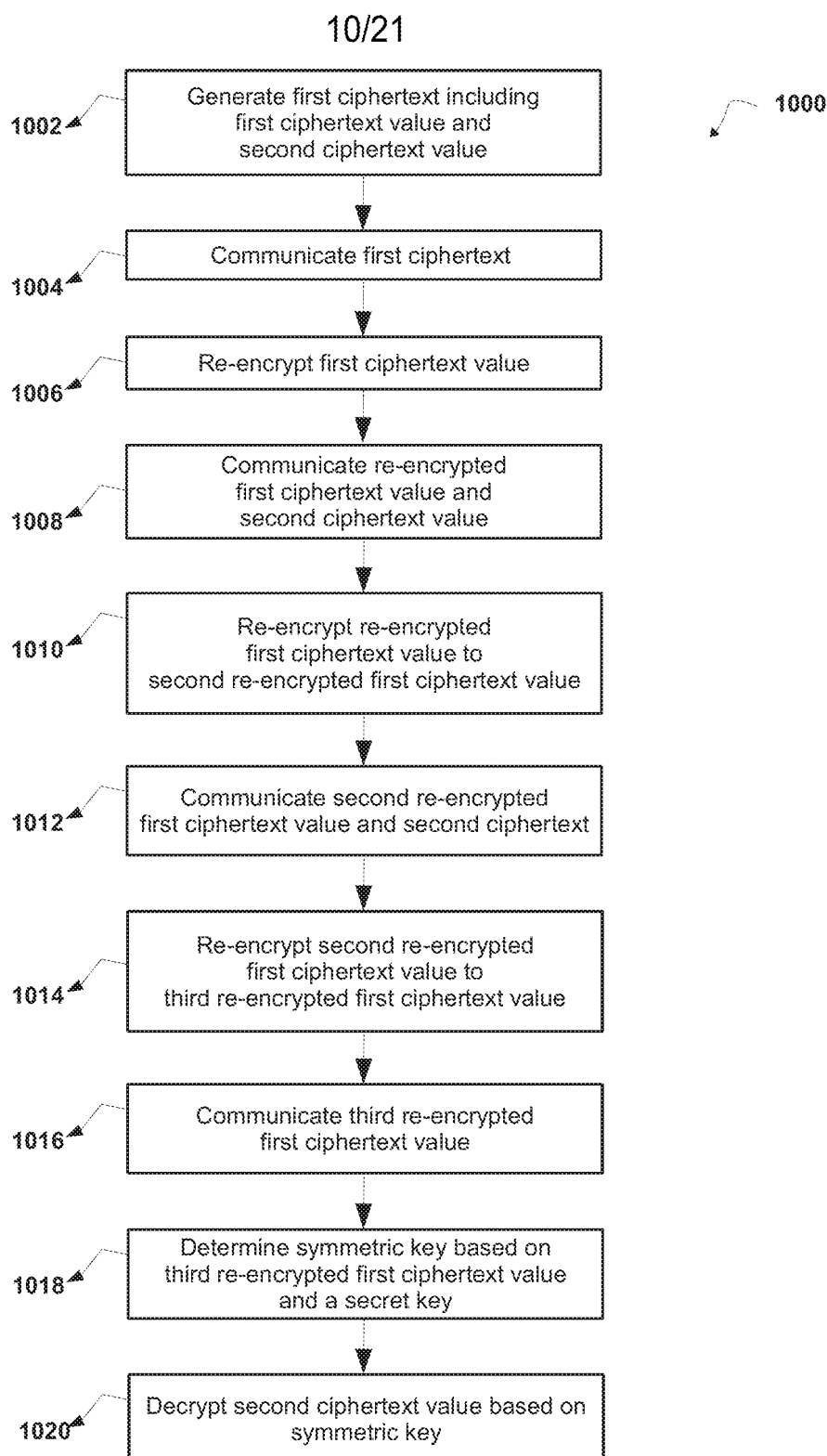


FIG. 10

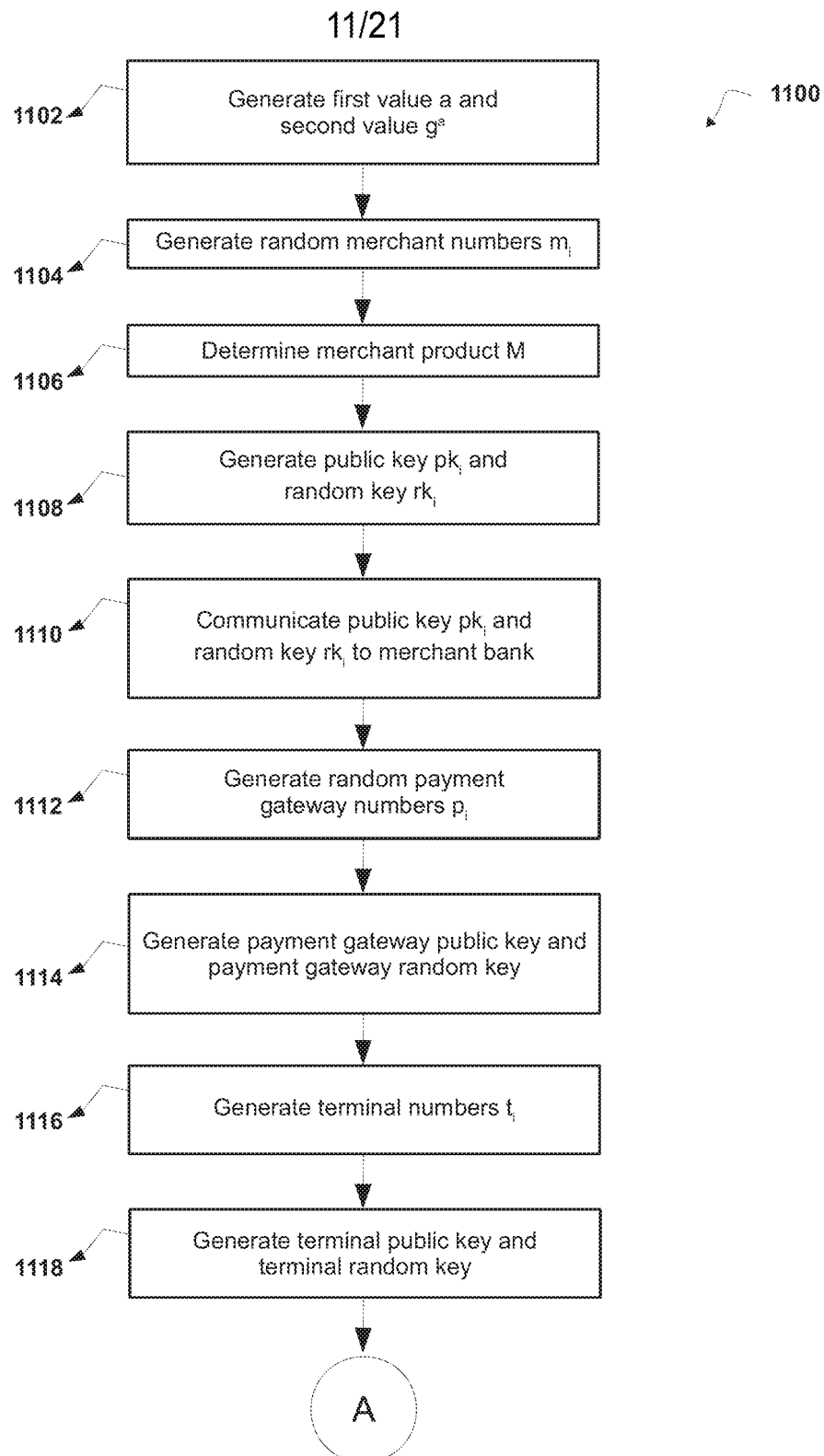


FIG. 11A

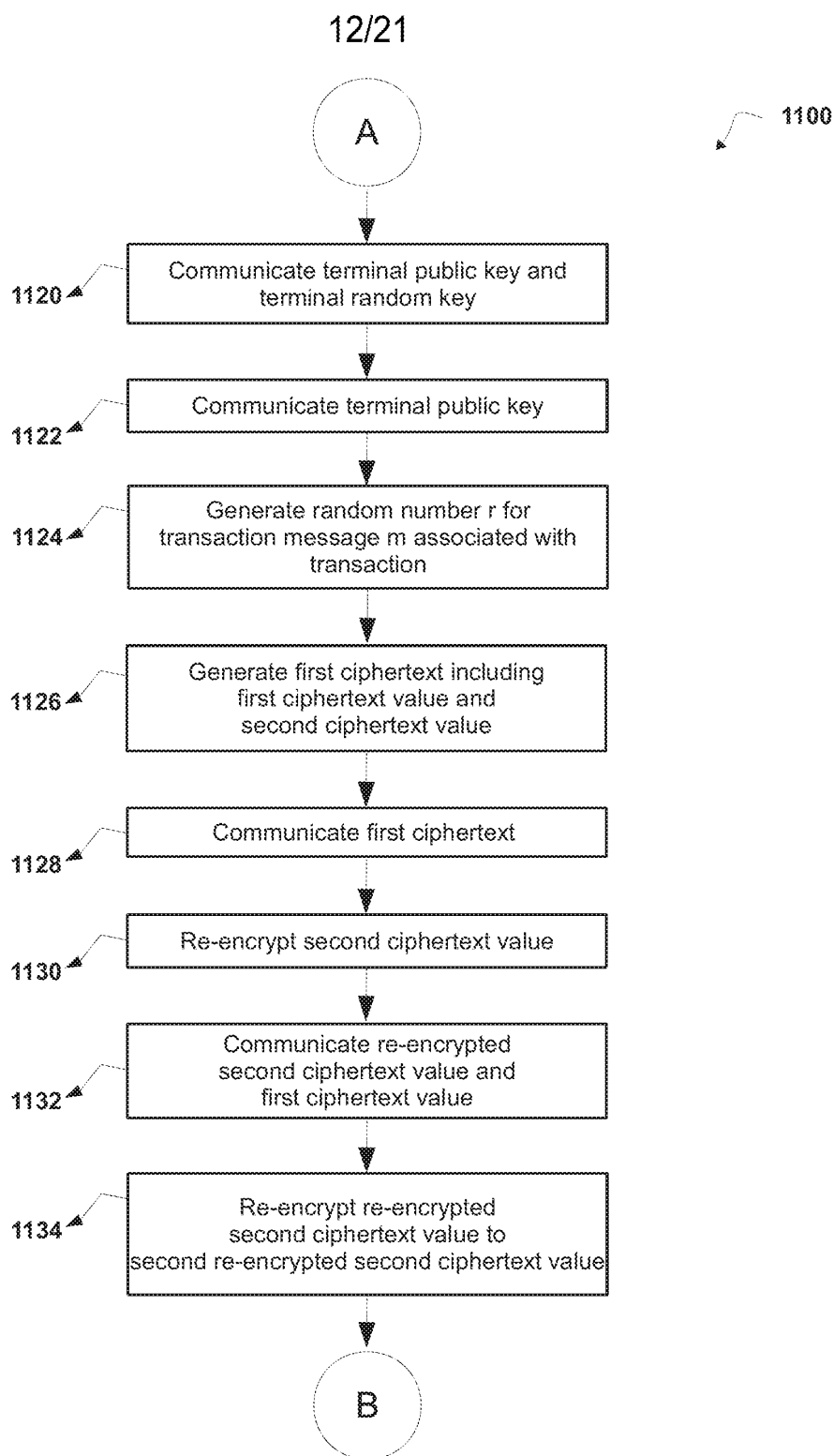


FIG. 11B

13/21

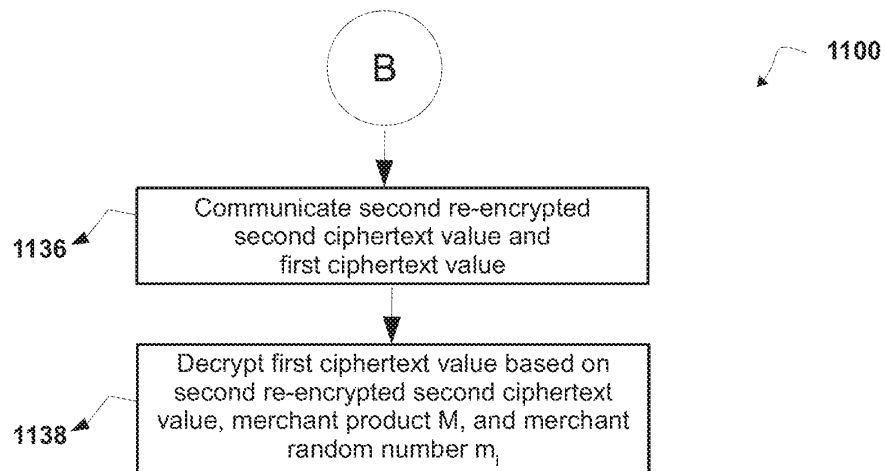


FIG. 11C

14/21

1200

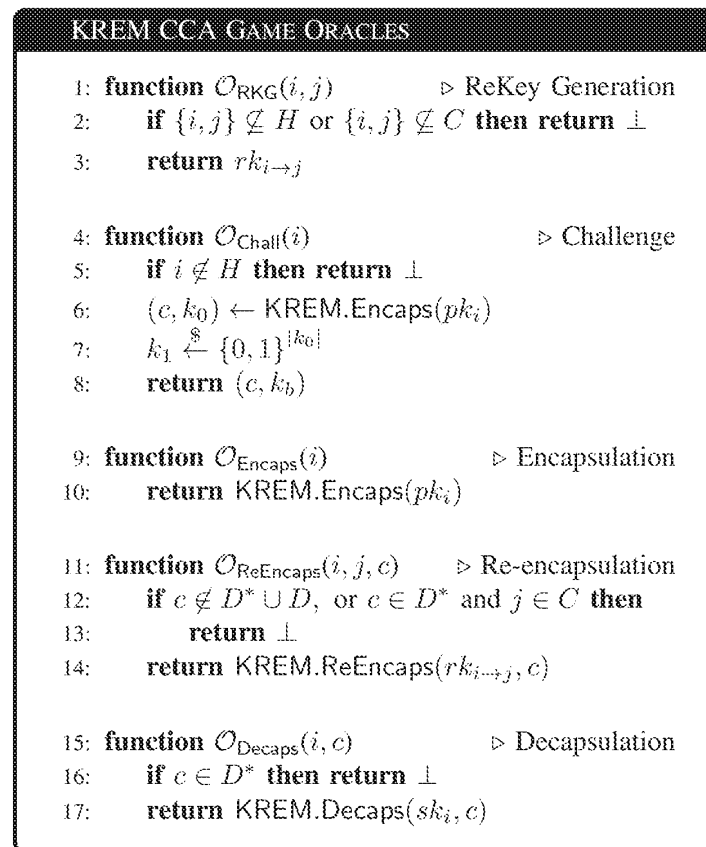


FIG. 12A

15/21

1250

THE ORACLES SIMULATED BY B

```

1: function  $\mathcal{O}_{\text{RKG}}(i, j)$  ▷ ReKey Generation
2:   if  $\{i, j\} \not\subseteq H$  or  $\{i, j\} \not\subseteq C$  then return  $\perp$ 
3:   else if  $\{i, j\} \subseteq C$  then
4:     return  $\text{KREM.RKG}((pk_i, sk_i), (pk_j, sk_j))$ 
5:   else if  $\{i, j\} \subseteq H$  then return  $r_i/r_j$ 

6: function  $\mathcal{O}_{\text{Chall}}(i)$  ▷ Challenge
7:   if  $i \notin H$  then return  $\perp$ 
8:   return  $((g^u)^{1/r_i}, W)$ 

9: function  $\mathcal{O}_{\text{Encaps}}(i)$  ▷ Encapsulation
10:   $r \xleftarrow{\$} \mathbb{Z}_q^*$ ;  $c \leftarrow g^r$ ;  $R[c] \leftarrow r$ 
11:  return  $(c, F(pk_i^*))$ 

12: function  $\mathcal{O}_{\text{ReEncaps}}(i, j, c)$  ▷ Re-encapsulation
13:  if  $c \notin D^* \cup D$ , or  $c \in D^*$  and  $j \in C$  then
14:    return  $\perp$ 
15:  else if  $\{i, j\} \subseteq C$  then
16:    return  $\text{KREM.ReEncaps}(sk_i/sk_j, c)$ 
17:  else if  $\{i, j\} \subseteq H$  then
18:    return  $\text{KREM.ReEncaps}(r_i/r_j, c)$ 
19:  else if  $i \in C, j \in H$  then
20:    return  $c^{sk_i/r_j}$ 
21:  else if  $i \in H, j \in C$  then
22:    for all  $i' \in H$  do
23:      if  $R[c^{r_i/r_{i'}}] \neq \perp$  then
24:         $r \leftarrow R[c^{r_i/r_{i'}}]$ 
25:        return  $(g^v)^{rr_{i'}/sk_i}$ 

26: function  $\mathcal{O}_{\text{Decaps}}(i, c)$  ▷ Decapsulation
27:  if  $c \in D^*$  then return  $\perp$ 
28:  else if  $i \in H$  then
29:    return  $F_v(c^{r_i})$ 
30:  return  $\text{KREM.Decaps}(sk_i, c)$ 

```

FIG. 12B

SUBSTITUTE SHEET (RULE 26)

16/21

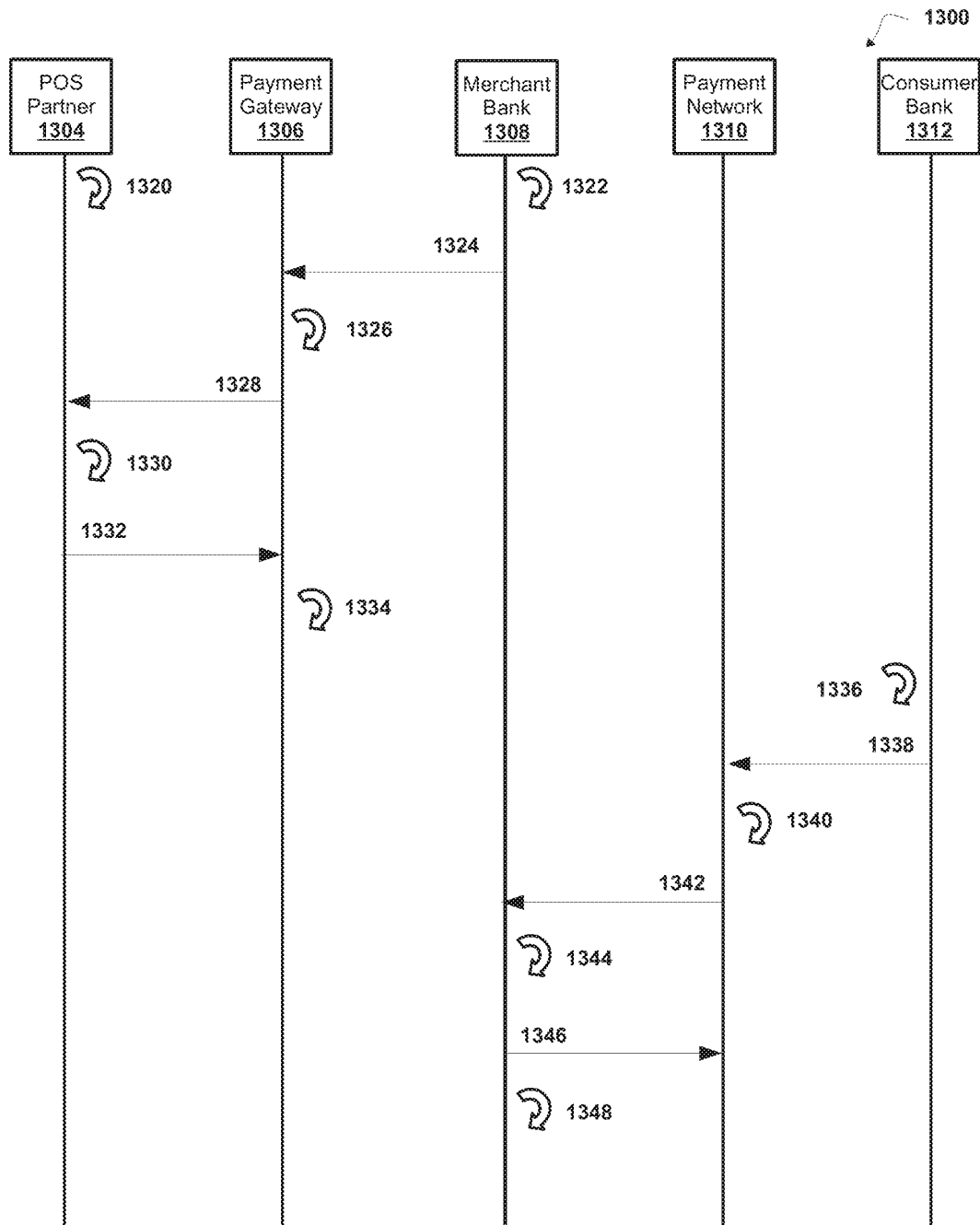


FIG. 13

SUBSTITUTE SHEET (RULE 26)

17/21

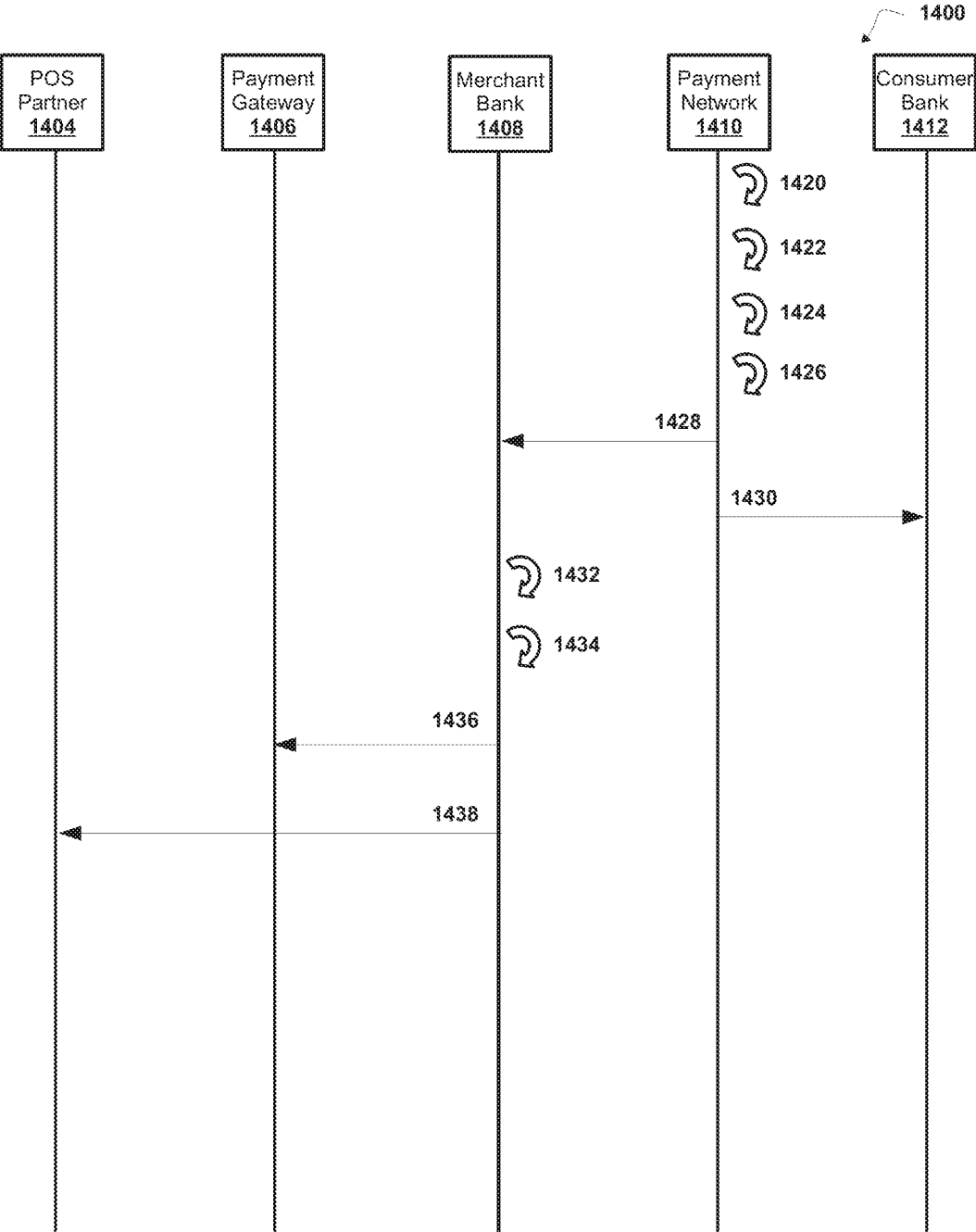


FIG. 14

18/21

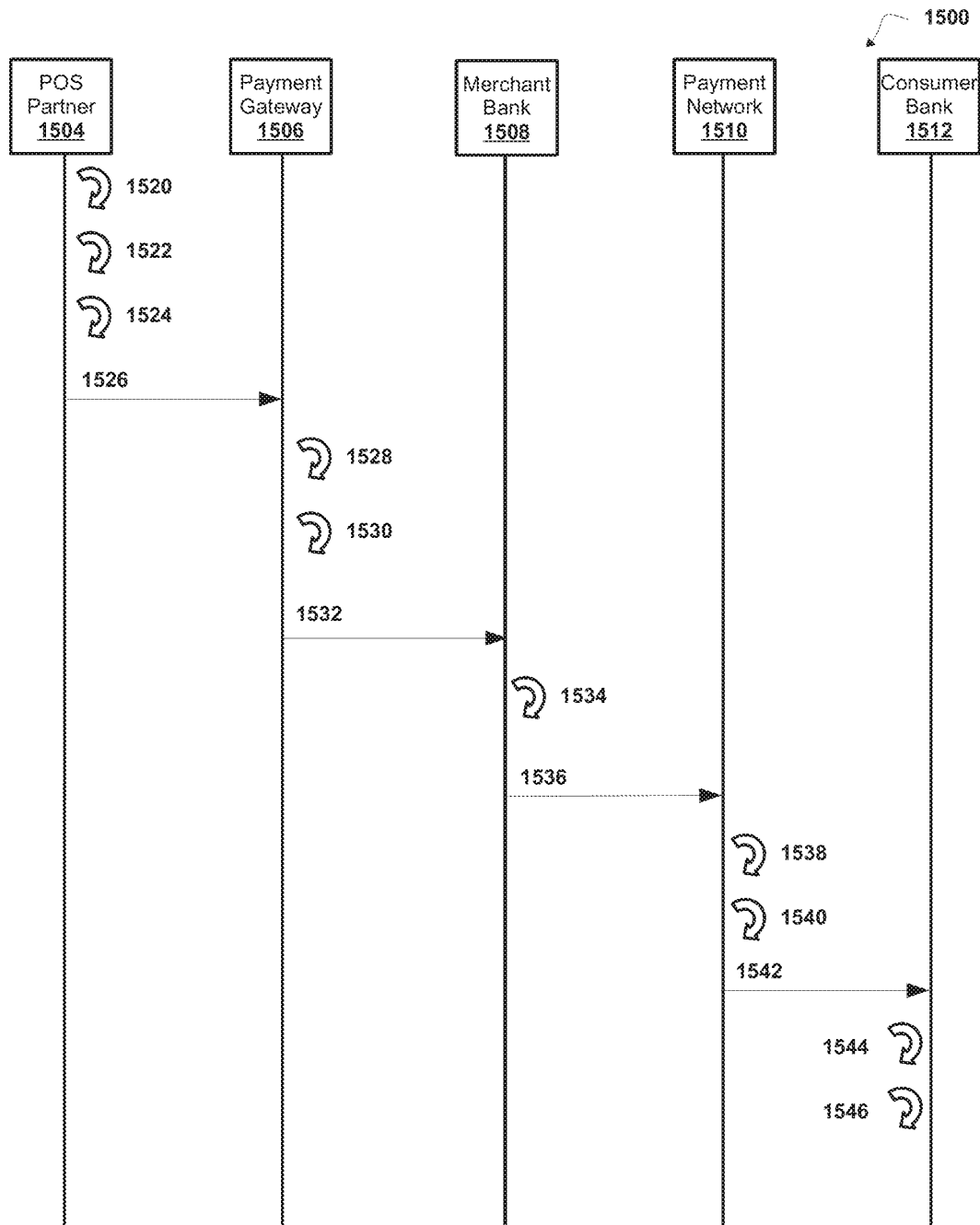


FIG. 15

SUBSTITUTE SHEET (RULE 26)

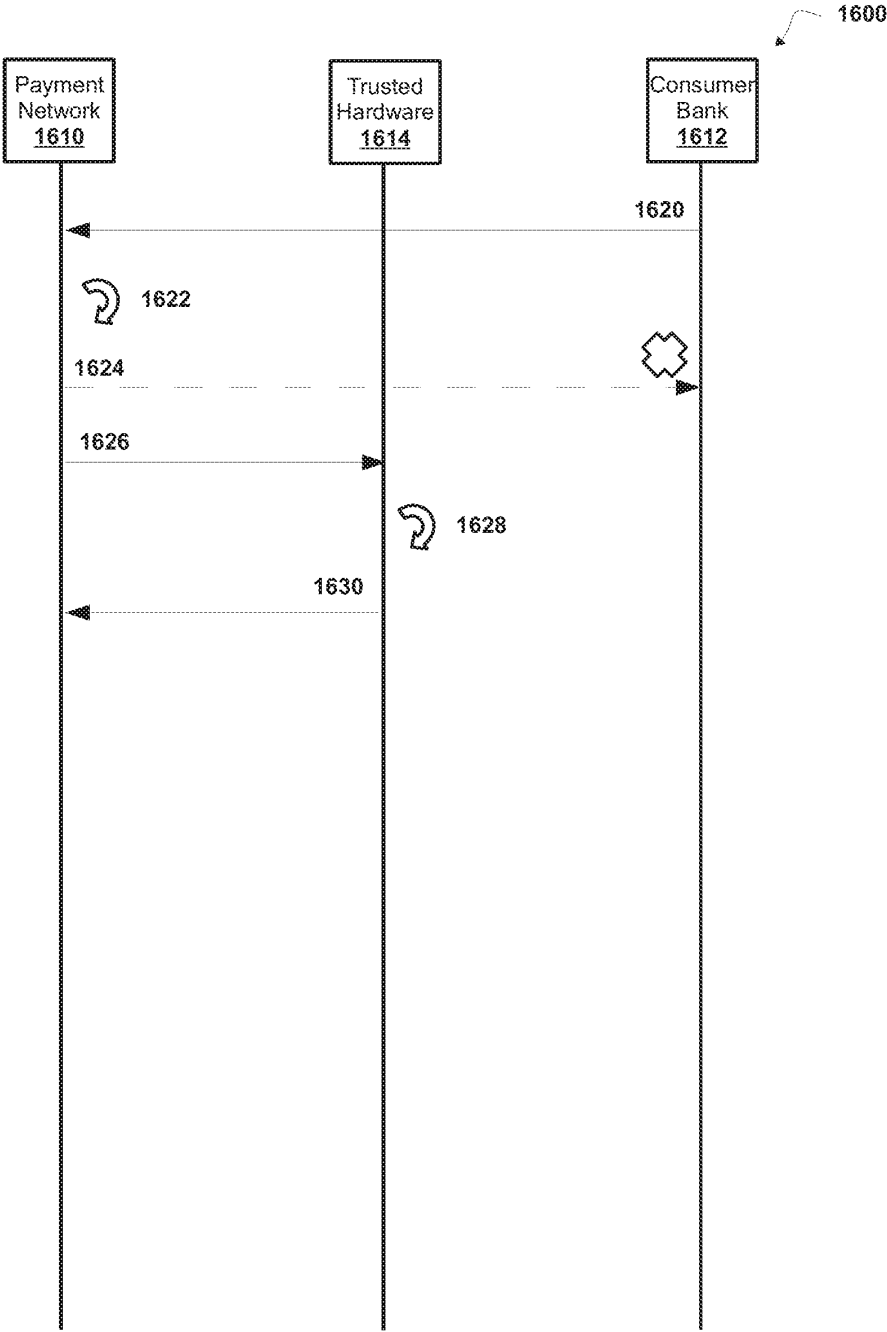


FIG. 16

20/21

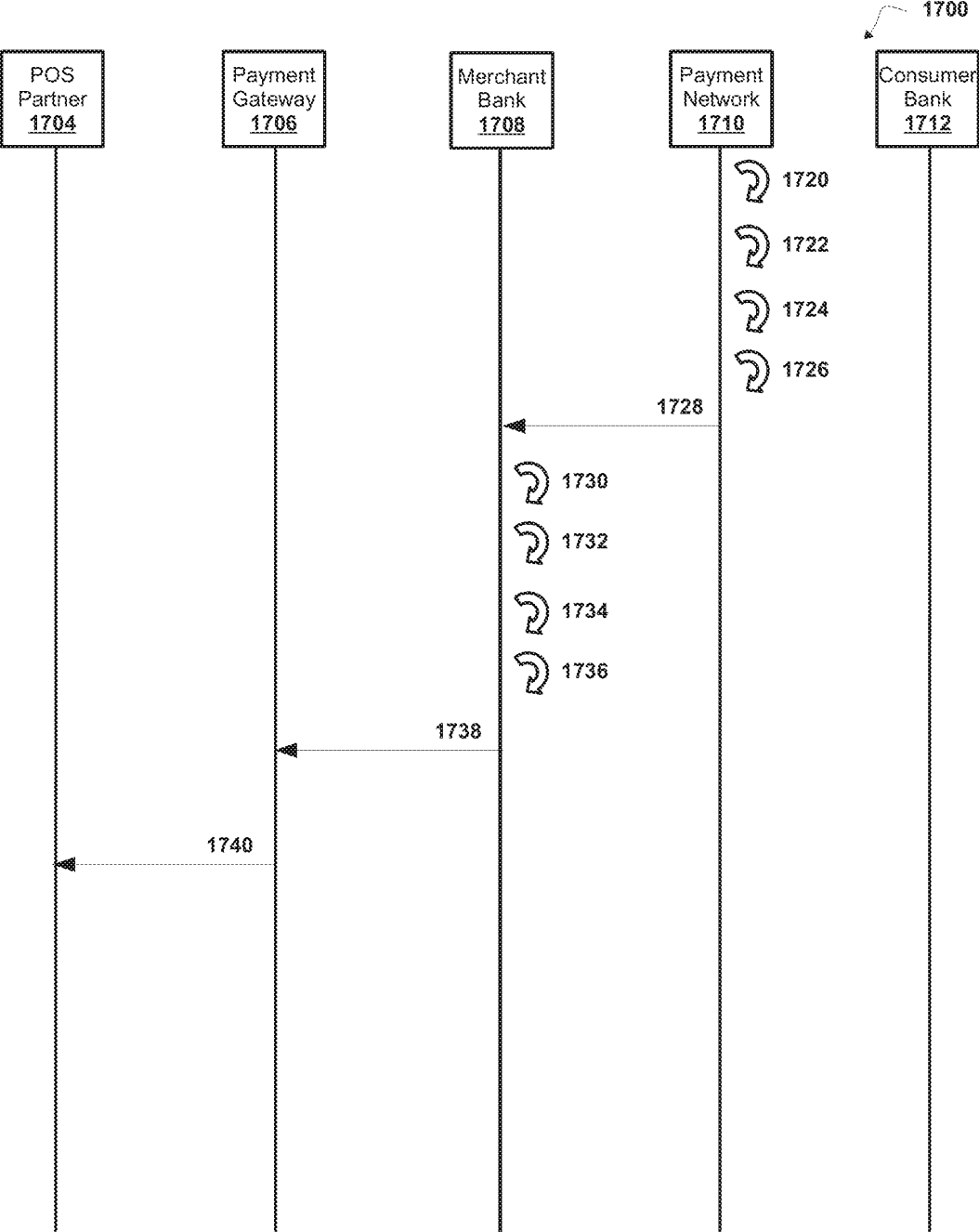


FIG. 17

21/21

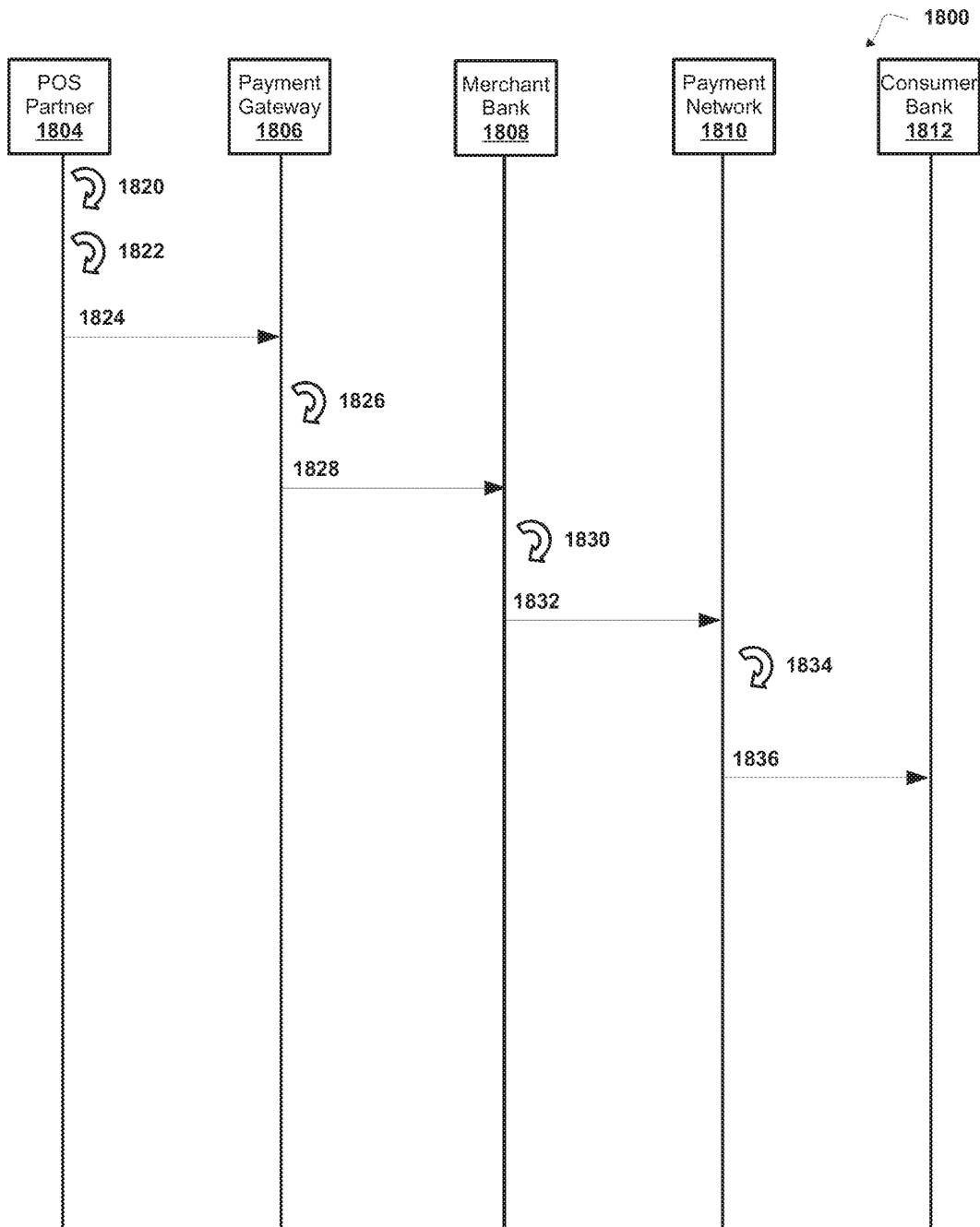


FIG. 18

SUBSTITUTE SHEET (RULE 26)