(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) **World Intellectual Property Organization**
International Bureau

(43) **International Publication Date**
15 October 2009 (15.10.2009)

PCT

(10) **International Publication Number**
WO 2009/126647 A2

(51) **International Patent Classification**: *H04L 29/06* (2006.01)

(21) **International Application Number**: PCT/US2009/039805

(22) **International Filing Date**: 7 April 2009 (07.04.2009)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
61/043,007 7 April 2008 (07.04.2008) US
61/081,756 18 July 2008 (18.07.2008) US

(71) **Applicant** *(for all designated States except US)*: **INTERDIGITAL PATENT HOLDINGS, INC.** [US/US]; 3411 Silverside Road, Concord Plaza, Suite 105, Hagley Building, Wilmington, DE 19810 (US).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **GUCCIONE, Louis, J.** [US/US]; 211 Lincoln Place, East Chester, NY 10709 (US). **DODIS, Yevgeniy** [US/US]; 2 Washington Square Village #12h, New York, NY 10012 (US). **SHAH, Yogendra, C.** [GB/US]; 10 Regency Court, Exton, PA 19341 (US). **CHA, Inhyok** [US/US]; 510 Southridge Circle, Yardley, PA 19067 (US).

(74) **Agent**: **KLINE, Adam, D.**; Volpe And Koenig, P.c., United Plaza, Suite 1600, 30 South 17th Street, Philadelphia, PA 19103 (US).

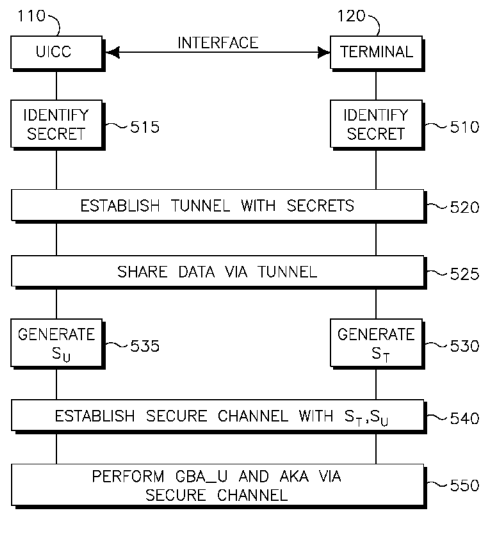
(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) **Title**: SECURE SESSION KEY GENERATION

FIG. 5



(57) **Abstract**: A method and apparatus for securing the interface between a Universal Integrated Circuit Card (UICC) and a Terminal in wireless communications is disclosed. The security of Authentication and Key Agreement (AKA) and application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U) procedures is improved. A secure shared session key is used to encrypt communications between the UICC and the Terminal. The secure shared session key generated using authenticating or non-authenticating procedures.

[0001] SECURE SESSION KEY GENERATION

[0002] FIELD OF INVENTION

[0003] This application is related to wireless communications.

[0004] BACKGROUND

[0005] The Authentication and Key Agreement (AKA) procedure is used for establishing authentication and shared secret keys for a wireless transmit/receive unit (WTRU) in a 3rd Generation Partnership Project (3GPP) communication network. The AKA provides for secure mutual authentication between two parties. In addition, the application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U), which is based on AKA procedures, provides a means to enable application security. However, the AKA and the application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U) procedures do not protect the security of the interface connecting the Universal Integrated Circuit Card (UICC) and Terminal of the WTRU. Critical key related material passes from the UICC to the Terminal during the AKA and GBA_U processes. As a result, the session keys (for example CK/IK and Ks_ext_NAF), are exposed during initial provisioning of the Terminal at the point of sale, when a local key has not yet been established and when an established local key expires.

[0006] Existing protocols that are designed to protect the connection between the UICC and the Terminal cannot be initiated until the AKA and GBA_U processes are complete. As a result, these protocols allow for eavesdropping of the keys. Attempts to secure the link between the Terminal and the UICC, after the AKA and GBA_U process, for other application level processes through interactions with and participation by the wireless network components, do not resolve these deficiencies.

[0007] Therefore, there exists a need for an improved method and apparatus for securing communications between a Terminal and a UICC.

[0008] SUMMARY

[0009] A method and apparatus for securing the interface between a Universal Integrated Circuit Card (UICC) and a Terminal in wireless communications is disclosed. The security of the Authentication and Key Agreement (AKA) and the application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U) procedures is improved. A secure shared session key is used to encrypt communications between the UICC and the Terminal. The secure shared session key generated using authenticating or non-authenticating procedures.

[0010] BRIEF DESCRIPTION OF THE DRAWINGS

[0011] A more detailed understanding may be had from the following description, given by way of example in conjunction with the accompanying drawings wherein:

[0012] Figure 1 shows an example of a wireless transmit/receive unit for performing secure session key generation;

[0013] Figure 2 shows an example of a Terminal configured as a handset for performing wireless communications;

[0014] Figure 3 shows an example of a wireless transmit/receive unit for performing secure session key generation in conjunction with connected device;

[0015] Figure 4 shows an example of a network for performing wireless communications;

[0016] Figure 5 is an example of session key generation for securing communications between the Universal Integrated Circuit Card and the Terminal;

[0017] Figure 6 shows an example of explicit mutual authentication using the AKA procedure;

[0018] Figure 7 shows an example of explicit mutual authentication using one-time authenticated encryption;

[0019] Figure 8 shows an example of explicit mutual authentication using one-time authenticated encryption and replay protection;

[0020] Figure 9 shows an example of implicit mutual authentication;

[0021] Figure 10 shows an example of implicit mutual authentication with replay protection; and

[0022] Figure 11 shows an example of shared secret key establishment without authentication.

[0023] DETAILED DESCRIPTION

[0024] When referred to hereafter, the terminology "wireless transmit/receive unit (WTRU)" includes but is not limited to a user equipment (UE), a mobile station, a fixed or mobile subscriber unit, a pager, a cellular telephone, a personal digital assistant (PDA), a computer, or any other type of user device capable of operating in a wireless environment. When referred to hereafter, the terminology "base station" includes but is not limited to a Node-B, a site controller, an access point (AP), or any other type of interfacing device capable of operating in a wireless environment. The terminology “WTRU” and “base station” are not mutually exclusive.

[0025] Figure 1 is an example block diagram of a wireless transmit/receive unit (WTRU) 100 for performing secure session key generation. The WTRU 100 includes a Universal Integrated Circuit Card (UICC) 110 and a Terminal 120. The UICC communicates with the Terminal via interface 130. The WTRU 100 is shown as including the UICC 110 and the Terminal 120 for illustrative purposes. The UICC 110 or the Terminal 120 may be configured in any manner so long as they are able to communicate as described herein. For example, Figure 3 shows an example wherein the Terminal 120 is located in a connected device.

[0026] Figure 2 is an example block diagram of an expanded view of the Terminal 120 configured as a handset for performing wireless communication. The Terminal 120 includes a processor 210, an antenna 220, a user interface 230, and a display 240.

[0027] Figure 3 is an example block diagram of a wireless transmit/receive unit (WTRU) 100 for performing secure session key generation in conjunction with a connected device 300. The UICC 10 in the WTRU 100 communicates with the Terminal 120 in a connected device 300 via the interface 130. The connected device 300 may be a personal computer (PC), or any other device configured as the Terminal 120. The interface 130 may be a wired or a wireless interface. The method and apparatus recited herein includes any other combination or configuration of a UICC 110 and a Terminal 120. Optionally, the Terminal 120 may include an internal or external UICC reader.

[0028] For example, the connected device 300 may be a laptop computer. The laptop may be connected to the internet via an Ethernet connection. The laptop may also be connected to the WTRU 100 via a Bluetooth interface 130. The UICC 110 in the WTRU 100 may then use the Terminal 120 in the laptop for performing communications requiring a secure connection. Alternatively, the Terminal 120 in the laptop may use the UICC 110 in the WTRU 100 for performing communications requiring a secure connection.

[0029] Figure 4 is an example block diagram of a network 400 for performing wireless communications. The network 400 includes the WTRU 100, a radio access network (RAN) 410, and a core network (CN) 420. The RAN 410 includes a base station 430, and a Radio Network Controller (RNC) 440. The CN 420 includes a Visitor Location Register (VLR) 450 and a Home Location Register (HLR) 460. The network 400 also includes an eavesdropper (EVE) 490. The base station 430 serves as a point of network entry for the RAN 410. The RNC 440 carries out various functions in wireless communication, such as radio resource management, mobility management functions, and encryption functions. The VLR 450 stores information about the WTRU 100, such as a copy of a user service profile and a device location area, which is used for wireless communications. The HLR 460, which stores a master copy of a user service profile, carries out switching functions and manages the wireless communications between the WTRU 100 and the network 400.

[0030] Figure 5 is an example of session key generation for securing the interface 130 between the UICC 110 and the Terminal 120. The Terminal 120 identifies a secret that can be used to encrypt communications with the UICC 110, at 510. Similarly, the UICC identifies a secret that can be used to encrypt communications with the Terminal 120, at 515. Optionally, the identified secrets are a pre-provisioned shared secret. A tunnel is established on the interface 130 using the secrets, at 520, such that a channel between the UICC 110 and the

Terminal 120 is secured with the respective secrets. The tunnel is used to share data for use in deriving a secure shared session key, at 525.

[0031] Next, the Terminal 120 derives a secure shared session key S_T from it's secret, at 530. Similarly, the UICC 110 derives a secure shared session key S_U from it's secret, at 535. Optionally, the UICC 110 and the Terminal 120 also perform mutual authentication, at 530, 535. The secure shared session keys S_T, S_U are used to establish a secure channel between the UICC 110 and the Terminal 120, at 540, such that the confidentiality and integrity of information passing through the secure channel are protected. The UICC 110 and the Terminal 120 then carry out the AKA 300 and GBA_U 400 procedures via the secure channel, at 550.

[0032] In some embodiments, the shared secret K is used to perform a keyed pseudorandom function (PRF) that is capable of accommodating arbitrary-length inputs, such as HMAC with SHA-256, encrypted CBC MAC with AES-128, or the AKA security functions. A PRF using a shared secret K and an input, x, may be denoted as $f_K(x)$. Similarly, the notation $f_K(x,y)$ indicates that the PRF is performed on a concatenation of the arguments shown. A PRF family is a set of related one-way, non-invertible, PRFs, wherein a value of variable bit-length is transformed to a bit sequence of fixed length (i.e., 128 or 256). For example a first PRF in a PRF family may be denoted as $f_K(0, Y, Z)$ and a second PRF in the PRF family may be denoted as $f_K(1, Y, Z)$, such that the PRF having the leading 0 produces a different result than the PRF having the leading 1.

[0033] In some embodiments, the Terminal 120 is configured to generate a random challenge (RAND), an anonymity key (AK), and a sequence number (SQN). Terminal 120 is also configured to compute a message authentication code (MAC), an Expected Response (XRES), an expected sequence number (XSQN), or an authentication value (Tag). Similarly, the UICC 110 is configured to generate a response (RES) or an expected authentication value (XTag). One having ordinary skill in the art would recognize that a RAND, an AK, a SQN, a MAC, and a XRES may be produced in accordance with any of a number of respective functions known in the art. Optionally, the functions may be the key generation functions defined by the 3rd generation partnership project (3GPP). The

Terminal 120 is also configured to send the calculated values to the UICC 110. The Terminal 120 is also configured to receive a response (RES) from the UICC 110 and to compare calculated values with received values for authentication of the UICC 110. Similarly, the UICC 110 is configured to send the values to the Terminal 120, and to compare calculated values with received values for authentication of the UICC 110. The Terminal 120 and UICC 110 are also configured to independently derive shared values, such as shared session keys and anonymity keys. For clarity, values produced at the UICC 110 may be indicated with the subscript U, and values produced at the Terminal 120 may be indicated with the subscript T. For example, AK_U at the UICC 110 has the same value as AK_T at the Terminal 120.

[0034] Figure 6 shows an example of an explicit mutual authentication and session key generation method 600. First, the Terminal 120 generates a RAND and a SQN_T, at 610. The Terminal 120 also computes a MAC, an XRES, an AK_T, and a XSQN, at 620. The MAC is computed based on the shared secret K, the RAND, and the SQN_T. The XRES represents an authentication code and is computed using the shared secret K and the RAND. The AK_T is generated using the shared secret K and the RAND. Optionally, the AK_T is the same size as the SQN_T. The XSQN is computed by performing a bitwise exclusive-or (XOR or $\oplus$) of the SQN and the AK_T.

[0035] Next, the Terminal 120 sends the MAC, the RAND, and the XSQN to the UICC 110 over the interface 130, at 630. The UICC 110 computes an AK_U, a SQN_U, and an expected MAC (XMAC), at 640. The AK_U is calculated using the shared secret K and the received RAND. The SQN_U is calculated by performing a bitwise exclusive-or of the AK_U and the XSQN. The XMAC is calculated using the shared secret K, the RAND, and the SQN_U. Optionally, the function used to calculate the AK_U at the UICC 110 is identical to the function used to calculate the AK_T at the Terminal 120.

[0036] Next the UICC 110 compares the XMAC with MAC, at 650. If the XMAC and the MAC are not equal, the authentication process fails and terminates with a fail condition, at 655. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the Terminal 120 is authenticated, and the UICC 110 computes a RES using the shared secret K and RAND, at 660. The UICC 110 sends the RES to the Terminal 120, at 670, and derives a shared session key S_U, at 680. For example, the shared session keys are derived using the RAND and the shared secret K.

[0037] Finally, the Terminal 120 compares the RES with the XRES, at 690. If the RES and the XRES are not equal, the authentication process fails and terminates with a fail condition, at 691. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the UICC 110 is authenticated, and the Terminal 120 derives a shared session key S_T, at 692. The UICC 110 and the Terminal 120 then use the shared session key S_U, S_T to perform the GBA_U 400 and AKA 300 procedures.

[0038] Figure 7 shows an example of an explicit mutual authentication and session key generation method 700 using one-time authenticated encryption. The Terminal 120 generates a session key S_T and a nonce R, at 705. Optionally, the nonce R is selected using a counter and the counter is incremented. The Terminal 120 computes the encrypted session key e of the session key S_T using the shared secret K, the nonce R, and a tuple E of the nonce R and the encrypted session key e at 710. The tuple E is generated by an encryption process according to the following vector notation:

$$E = (R, e=f_K(0,R) \oplus S_T).$$

Equation (1)

[0039] The Terminal 120 then calculates an authentication value Tag using the shared secret K, the nonce R and the encrypted session key e at 720, according to the following equation:

$$Tag = f_K(0,R,e).$$

Equation (2)

[0040] Next, the Terminal 120 sends the tuple E and the authentication value Tag to the UICC 110 over the interface 130, at 730. The UICC 110 uses the shared secret K and the received tuple E to validate the received authentication value Tag, at 740. This validation may be denoted as:

$$\text{Tag} == f_K(0,R,e).$$
Equation (3)

[0041] If the received authentication value Tag is not validated, the authentication process fails and terminates with a fail condition, at 745. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the Terminal 120 is authenticated and the UICC decrypts the session key S_U, at 750, according to the following equation:

$$S_U = f_K(0,R) \oplus e.$$
Equation (4)

[0042] Next, the UICC 110 computes an expected authentication value (XTag), at 760. This computation may be denoted as:

$$\text{XTag} = f_K(1,R).$$
Equation (5)

[0043] The UICC 110 sends the expected authentication value XTag to the Terminal 120 over the interface 130, at 770. The Terminal 120 uses the shared secret K and the nonce R to validate the received XTag, at 780. This validation may be denoted as:

$$\text{XTag} == f_K(1,R).$$
Equation (6)

[0044] If the XTag is validated the UICC 110 is authenticated, at 790. Otherwise, the authentication process fails and terminates with a fail condition, at 791. Optionally, the authentication process may be restarted after a predetermined interval.

[0045] Figure 8 shows an example of an explicit mutual authentication and session key generation method 800 using one-time authenticated encryption and replay attack protection. The UICC 110 generates a nonce N at 805. Although a nonce is shown in figure 8, any appropriate pre-key negotiation parameter may be used. Optionally, the nonce N is generated using a counter and the counter is incremented. The UICC 110 then sends the nonce N to the Terminal 120 over the interface 130, at 810.

[0046] The Terminal 120 generates a session key S_T and a nonce R, at 820. Optionally, the nonce R is generated using a counter and the counter is incremented. The Terminal 120 computes the encrypted session key e of the session key S_T using the shared secret K and the nonce R per Equation 1, at 830. The Terminal 120 then calculates an authentication value Tag, using the shared secret K, the nonce R, the encrypted session key e, and the nonce N, at 840. This calculation may be denoted as:

$$Tag = f_K(0,R,e, N).$$

Equation (7)

[0047] Next, the Terminal 120 sends the authentication value Tag and a tuple E of the nonce R, and the encrypted session key e to the UICC 110 over the interface 130, at 850. The UICC 110 uses the shared secret K, the received tuple E, and the nonce N, to validate the received authentication value Tag, at 860. This validation may be denoted as:

$$Tag == f_K(0,R,e, N).$$

Equation (8)

[0048] If the received authentication value Tag is not validated, the authentication process fails and terminates with a fail condition, at 865. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the UICC decrypts the session key S_U, per Equation 4, at 870. Next, the UICC 110 computes an expected authentication value XTag per Equation 5, at 880.

[0049] The UICC 110 sends the XTag to the Terminal 120 over the interface 130, at 890. The Terminal 120 uses the nonce R to validate the received XTag per Equation 6, at 892. If the XTag is validated, the UICC 110 is authenticated, at 894. Otherwise, the authentication process fails and terminates with a fail condition, at 896. Optionally, the authentication process may be restarted after a predetermined interval.

[0050] Figure 9 shows an example of implicit mutual authentication and session key generation. The Terminal 120 generates a nonce R, at 900. Optionally, the nonce R is generated using a counter and the counter is incremented. The Terminal 120 then calculates an authentication value Tag using the shared secret K and the nonce R, at 910. This calculation may be denoted as:

$$\text{Tag} = f_K(0, R).$$

Equation (9)

[0051] Next, the Terminal 120 sends nonce R and the authentication value Tag to the UICC 110 over the interface 130, at 920. The UICC 110 uses the shared secret K and the nonce R to validate the received authentication value Tag, at 930. This validation may be denoted as:

$$\text{Tag} == f_K(0,R).$$

Equation (10)

[0052] If the received authentication value Tag is not validated, the authentication process fails and terminates with a fail condition, at 935. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the Terminal 120 is authenticated and the UICC 110 computes session key S_U using the shared secret K and the nonce R, at 940. The session key computation may be denoted as:

$$S_U = f_K(2,R).$$

Equation (11)

[0053] Next, the UICC 110 computes an expected authentication value XTag per Equation 5, at 950. The UICC 110 sends the expected authentication value XTag to the Terminal 120 over the interface 130, at 960. The Terminal 120 uses the nonce R to validate the received expected authentication value XTag per Equation 6, at 970. If the received expected authentication value XTag is not validated the authentication process fails and terminates with a fail condition, at 975. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the UICC 110 is authenticated, and the Terminal 120 computes the session key S_T using the shared secret K and the nonce R, at 980. The session key computation may be denoted as:

$$S_T = f_K(2,R).$$

Equation (12)

[0054] Figure 10 shows an example of implicit mutual authentication and session key generation with replay protection. The UICC 110 generates a nonce N, at 1005. Optionally, the nonce N is generated using a counter and the counter is incremented. The UICC 110 then sends the nonce N to the Terminal 120 over the interface 130, at 1010.

[0055] The Terminal 120 generates a nonce R, at 1020. Optionally, the nonce R is generated using a counter and the counter is incremented. The Terminal 120 then calculates an authentication value Tag using the nonce R and the nonce N, at 1030. This calculation may be denoted as:

$$Tag = f_K(0, R, N).$$

Equation (13)

[0056] Next, the Terminal 120 sends nonce R and the authentication value Tag to the UICC 110 over the interface 130, at 1040. The UICC 110 uses the shared secret K, the nonce R, and the nonce N to validate the received authentication value Tag, at 1050. This validation may be denoted as:

$$Tag == f_K(0, R, N).$$

Equation (14)

[0057] If the received authentication value Tag is not validated, the authentication process fails and terminates with a fail condition, at 1055. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the Terminal 120 is authenticated and the UICC 110 computes the session key S_U using the shared secret K and the nonce R, per Equation 11, at 1060. Next, the UICC 110 computes an expected authentication value XTag, per Equation 5, at 1070. The UICC 110 sends the expected authentication value XTag to the Terminal 120 over the interface 130, at 1080.

[0058] Next, the Terminal 120 uses the nonce R to validate the received expected authentication value XTag per Equation 6, at 1090. If the received expected authentication value XTag is not validated the authentication process fails and terminates with a fail condition, at 1091. Optionally, the authentication process may be restarted after a predetermined interval. Otherwise, the UICC 110 is authenticated and the Terminal 120 computes the session key S_T, using the shared secret K and the nonce R, at 1092. The session key computation may be denoted as:

$$S_T = f_K(2,R).$$

Equation (15)

[0059] Figure 11 shows an example of shared secret key establishment without authentication using a Diffie-Hellman key exchange protocol. First, the UICC 110 and the Terminal 120 agree upon a very large prime number, *p*, and a generator, g, at 1100. The algebraic structure employed is the multiplicative group F_p^*, derived from the field F_p. F_p^* is cyclic and contains the generator g, such that, for any member a of F_p^* an integer n can be found such that $a = g^n \bmod p$. The values p and g are known publically, and represent a public key part of a key pair.

[0060] Next, the Terminal 120 randomly selects a private key, $RAND_i$, such that the private key $RAND_i$ is at least one (1) and is not greater than two (2) less than the very large prime number p, at 1110. The Terminal 120 computes g_{RAND_i} from the private key $RAND_i$, at 1120. This computation may be denoted as:

$$g_{RAND_i} \equiv g^{RAND_i} \bmod p.$$ Equation (16)

[0061] Similarly, the UICC 110 selects a private key, FRESH, such that the private key FRESH is at least one (1) and is not greater than two (2) less than the very large prime number p, at 1130. Then the UICC 110 computes g_{FRESH} from the private key FRESH, at 1140. This computation may be denoted as:

$$g_{FRESH} \equiv g^{FRESH} \bmod p.$$ Equation (17)

[0062] Next, the UICC 110 and the Terminal 120 exchange g_{RAND_i} and g_{FRESH} over the interface 130, at 1150.

[0063] Next, the Terminal 120 computes the shared secret, K, using the private key $RAND_i$ and the received g_{FRESH}, at 1160. This computation may be denoted as:

$$\mathrm{K} \equiv g_{FRESH}^{RAND_i} \bmod p.$$ Equation (18)

[0064] Similarly, the UICC 110 computes the shared secret, K', using the private key FRESH and the received g_{RAND_i}, at 1170. This computation may be denoted as:

$$K' \equiv g_{RAND_i}^{FRESH} \bmod p . \qquad \text{Equation (19)}$$

[0065] The Terminal 120 and the UICC 110 now possess a shared secret, K' = K, which is then used to compute a secure secret session key S, at 1165, 1175. The secure secret session key S is used to perform the GBA_U and AKA procedures by securing the interface 130, at 1180.

[0066] Although features and elements are described above in particular combinations, each feature or element can be used alone without the other features and elements or in various combinations with or without other features and elements. The methods or flow charts provided herein may be implemented in a computer program, software, or firmware incorporated in a computer-readable storage medium for execution by a general purpose computer or a processor. Examples of computer-readable storage mediums include a read only memory (ROM), a random access memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs).

[0067] Suitable processors include, by way of example, a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine.

[0068] Embodiments

1. A method for securing communications between a Universal Integrated Circuit Card (UICC) and a Terminal.

2. A method as in any one of the preceding embodiments, wherein securing communications includes generating a secure shared session key

3. A method as in any one of the preceding embodiments, wherein securing communications includes encrypting communications between the UICC and the Terminal with the secure shared session key.

4. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes deriving the secure shared session key from a shared secret.

5. A method as in any one of the preceding embodiments, wherein deriving the secure shared session key from a shared secret includes generating a shared secret from a secret.

6. A method as in any one of the preceding embodiments, wherein deriving the secure shared session key includes performing a pseudorandom function (PRF) using the shared secret.

7. A method as in any one of the preceding embodiments, wherein encrypting communications includes establishing a secure channel.

8. A method as in any one of the preceding embodiments, further comprising:

performing an application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U) procedure, using the secure channel

9. A method as in any one of the preceding embodiments, further comprising:

performing an Authentication and Key Agreement (AKA) procedure, using the secure channel.

10. A method as in any one of the preceding embodiments, further comprising:

creating a tunnel on an interface between the UICC and the Terminal.

11. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes determining whether a secure shared session key exists between the UICC and the Terminal.

12. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes generating a new secure shared session key on a condition that a secure shared session key does not exist.

13. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes producing a produced key negotiation parameter.

14. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes reporting the produced key negotiation parameter to the UICC.

15. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes receiving a received key negotiation parameter.

16. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes creating the secure shared session key using the produced key negotiation parameter and the received key negotiation parameter.

17. A method as in any one of the preceding embodiments, wherein the creating includes determining whether the produced key negotiation parameter is the same as the received key negotiation parameter.

18. A method as in any one of the preceding embodiments, wherein the creating includes deriving a secure shared session key on a condition that the produced key negotiation parameter is the same as the received key negotiation parameter.

19. A method as in any one of the preceding embodiments, wherein the producing includes selecting a random challenge (RAND) and a sequence number (SQN).

20. A method as in any one of the preceding embodiments, wherein the producing includes calculating an anonymity key (AK).

21. A method as in any one of the preceding embodiments, wherein the producing includes calculating a message authentication code (MAC).

22. A method as in any one of the preceding embodiments, wherein the producing includes calculating an expected response (XRES)

23. A method as in any one of the preceding embodiments, wherein the producing includes calculating an expected sequence (XSQN)

24. A method as in any one of the preceding embodiments, wherein the producing includes combining the RAND, the MAC, and the XSQN to produce the produced key negotiation parameter.

25. A method as in any one of the preceding embodiments, wherein the calculating includes computing the AK using a shared secret and the RAND.

26. A method as in any one of the preceding embodiments, wherein the calculating includes computing the MAC using the shared secret, the RAND, and the SQN.

27. A method as in any one of the preceding embodiments, wherein the calculating includes computing the XRES using the shared secret and the RAND.

28. A method as in any one of the preceding embodiments, wherein the calculating includes computing the XSQN using the SQN and the AK.

29. A method as in any one of the preceding embodiments, wherein the producing includes selecting a nonce.

30. A method as in any one of the preceding embodiments, wherein the producing includes calculating an authentication value (Tag).

31. A method as in any one of the preceding embodiments, wherein the producing includes combining the nonce and the Tag to produce the produced key negotiation parameter.

32. A method as in any one of the preceding embodiments, wherein the producing includes selecting a session key.

33. A method as in any one of the preceding embodiments, wherein the producing includes calculating an encrypted session key.

34. A method as in any one of the preceding embodiments, wherein the producing includes using the encrypted session key to produce the key negotiation parameter.

35. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes receiving a received key negotiation parameter.

36. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes producing a produced key negotiation parameter.

37. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes reporting the produced key negotiation parameter to the Terminal.

38. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes creating the secure shared session key using the received key negotiation parameter and the produced key negotiation parameter.

39. A method as in any one of the preceding embodiments, wherein the creating includes determining whether the produced key negotiation parameter is the same as the received key negotiation parameter.

40. A method as in any one of the preceding embodiments, wherein the creating includes deriving a secure shared session key on a condition that the produced key negotiation parameter is the same as the received key negotiation parameter.

41. A method as in any one of the preceding embodiments, wherein the producing includes extracting a random challenge (RAND) from the received key negotiation parameter.

42. A method as in any one of the preceding embodiments, wherein the producing includes extracting a message authentication code (MAC) from the received key negotiation parameter.

43. A method as in any one of the preceding embodiments, wherein the producing includes extracting an expected sequence (XSQN) from the received key negotiation parameter.

44. A method as in any one of the preceding embodiments, wherein the producing includes calculating an anonymity key (AK).

45. A method as in any one of the preceding embodiments, wherein the producing includes calculating an expected message authentication code (XMAC).

46. A method as in any one of the preceding embodiments, wherein the producing includes calculating a sequence number (SQN).

47. A method as in any one of the preceding embodiments, wherein the producing includes determining whether the XMAC is the same as the MAC.

48. A method as in any one of the preceding embodiments, wherein the producing includes computing a response (RES) using a shared secret and the RAND on a condition that the XMAC is the same as the MAC.

49. A method as in any one of the preceding embodiments, wherein the calculating includes computing the AK using the shared secret and the RAND.

50. A method as in any one of the preceding embodiments, wherein the calculating includes computing the SQN using the XSQN and the AK.

51. A method as in any one of the preceding embodiments, wherein the calculating includes computing the XMAC using the shared secret, the RAND, and the SQN.

52. A method as in any one of the preceding embodiments, wherein the producing includes extracting a nonce and a Tag from the received key negotiation parameter.

53. A method as in any one of the preceding embodiments, wherein the producing includes validating the Tag.

54. A method as in any one of the preceding embodiments, wherein the producing includes deriving a session key and computing an expected authentication value (XTag) on a condition that the Tag is valid.

55. A method as in any one of the preceding embodiments, wherein the producing includes producing the produced key negotiation parameter using the XTag.

56. A method as in any one of the preceding embodiments, wherein the producing includes extracting the encrypted session key from the received key negotiation parameter.

57. A method as in any one of the preceding embodiments, wherein the producing includes deriving a session key includes decrypting the encrypted session key.

58. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes generating a pre-key negotiation parameter.

59. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes reporting the pre-key negotiation parameter to the terminal.

60. A method as in any one of the preceding embodiments, wherein generating a secure shared session key includes receiving a pre-key negotiation parameter from the UICC.

61. A method as in any one of the preceding embodiments, wherein the generating includes performing a Diffie-Hellman key exchange protocol.

62. A wireless transmit/receive unit (WTRU) configured to perform at least part of any one of the preceding embodiments.

63. A base station configured to perform at least part of any one of the preceding embodiments.

64. An integrated circuit configured to perform at least part of any one of the preceding embodiments.

[0069] A processor in association with software may be used to implement a radio frequency transceiver for use in a wireless transmit receive unit (WTRU), user equipment (UE), Terminal, base station, radio network controller (RNC), or any host computer. The WTRU may be used in conjunction with modules, implemented in hardware and/or software, such as a camera, a video camera module, a videophone, a speakerphone, a vibration device, a speaker, a microphone, a television transceiver, a hands free headset, a keyboard, a Bluetooth® module, a frequency modulated (FM) radio unit, a liquid crystal display (LCD) display unit, an organic light-emitting diode (OLED) display unit, a digital music player, a media player, a video game player module, an Internet browser, and/or any wireless local area network (WLAN) or Ultra Wide Band (UWB) module.

CLAIMS

What is claimed is:

1. A method for securing communications between a Universal Integrated Circuit Card (UICC) and a Terminal, the method comprising:

generating a secure shared session key; and

encrypting communications between the UICC and the Terminal with the secure shared session key.

2. The method of claim 1, wherein generating a secure shared session key includes deriving the secure shared session key from a shared secret.

3. The method of claim 2, wherein deriving the secure shared session key from a shared secret includes generating a shared secret from a secret.

4. The method of claim 2, wherein deriving the secure shared session key includes performing a pseudorandom function (PRF) using the shared secret.

5. The method of claim 1, wherein encrypting communications includes establishing a secure channel.

6. The method of claim 5, further comprising:

performing at least one of an application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U) procedure or an Authentication and Key Agreement (AKA) procedure, using the secure channel.

7. The method of claim 1, further comprising:

creating a tunnel on an interface between the UICC and the Terminal.

8. The method of claim 1, wherein generating a secure shared session key includes:

determining whether a secure shared session key exists between the UICC and the Terminal; and

on a condition that a secure shared session key does not exist, generating a new secure shared session key.

9. The method of claim 1, wherein generating a secure shared session key includes:

producing a produced key negotiation parameter;

reporting the produced key negotiation parameter to the UICC;

receiving a received key negotiation parameter;

creating the secure shared session key using the produced key negotiation parameter and the received key negotiation parameter.

10. The method of claim 9, wherein the creating includes:

determining whether the produced key negotiation parameter is the same as the received key negotiation parameter; and

on a condition that the produced key negotiation parameter is the same as the received key negotiation parameter, deriving a secure shared session key.

11. The method of claim 9, wherein the producing includes:

selecting a random challenge (RAND) and a sequence number (SQN);

calculating an anonymity key (AK), a message authentication code (MAC), an expected response (XRES), and an expected sequence (XSQN); and

combining the RAND, the MAC, and the XSQN to produce the produced key negotiation parameter.

12. The method of claim 11, wherein the calculating includes:

computing the AK using a shared secret and the RAND;

computing the MAC using the shared secret, the RAND, and the SQN;

computing the XRES using the shared secret and the RAND; and

computing the XSQN using the SQN and the AK.

13. The method of claim 9, wherein the producing includes:

selecting a nonce;

calculating an authentication value (Tag); and

combining the nonce and the Tag to produce the produced key negotiation parameter.

14. The method of claim 9, wherein the producing includes:

selecting a session key;

calculating an encrypted session key; and

using the encrypted session key to produce the key negotiation parameter.

15. The method of claim 1, wherein generating a secure shared session key includes:

receiving a received key negotiation parameter;

producing a produced key negotiation parameter;

reporting the produced key negotiation parameter to the Terminal; and

creating the secure shared session key using the received key negotiation parameter and the produced key negotiation parameter.

16. The method of claim 15, wherein the creating includes:

determining whether the produced key negotiation parameter is the same as the received key negotiation parameter; and

on a condition that the produced key negotiation parameter is the same as the received key negotiation parameter, deriving a secure shared session key.

17. The method of claim 15, wherein the producing includes:

extracting a random challenge (RAND), a message authentication code (MAC), and an expected sequence (XSQN) from the received key negotiation parameter;

calculating an anonymity key (AK), an expected message authentication code (XMAC), and a sequence number (SQN);

determining whether the XMAC is the same as the MAC; and

on a condition that the XMAC is the same as the MAC, computing a response (RES) using a shared secret and the RAND.

18. The method of claim 17, wherein the calculating includes:

computing the AK using the shared secret and the RAND;

computing the SQN using the XSQN and the AK; and

computing the XMAC using the shared secret, the RAND, and the SQN.

19. The method of claim 15, wherein the producing includes:

extracting a nonce and a Tag from the received key negotiation parameter;

validating the Tag;

on a condition that the Tag is valid, deriving a session key and computing an expected authentication value (XTag); and

producing the produced key negotiation parameter using the XTag.

20. The method of claim 19, wherein the producing includes extracting the encrypted session key from the received key negotiation parameter; and deriving a session key includes decrypting the encrypted session key.

21. The method of claim 1, wherein generating a secure shared session key includes:

generating a pre-key negotiation parameter; and

reporting the pre-key negotiation parameter to the terminal.

22. The method of claim 1, wherein generating a secure shared session key includes:

receiving a pre-key negotiation parameter from the UICC.

23. The method of claim 1, wherein the generating includes performing a Diffie-Hellman key exchange protocol.

24. A wireless transmit/receive unit (WTRU), the WTRU comprising:

a Universal Integrated Circuit Card (UICC) configured to:

generate a secure shared session key,

encrypt communications with the secure shared session key,

transmit the encrypted communications, and

decrypt received encrypted communications using the secure shared session key; and

a Terminal configured to

generate the secure shared session key,

encrypt communications with the secure shared session key,

transmit the encrypted communications, and

decrypt received encrypted communications using the secure shared session key.

25. The WTRU of claim 24, wherein the UICC is configured to generate the secure shared session key by deriving the secure shared session key from a shared secret, and the Terminal is configured to generate the secure shared session key by deriving the secure shared session key from the shared secret.

26. The WTRU of claim 25, wherein the UICC is configured to derive the secure shared session key from the shared secret by generating the shared secret from a first secret, and the Terminal is configured to derive the secure shared session key from the shared secret by generating the shared secret from a second secret.

27. The WTRU of claim 25, wherein the UICC is configured to derive the secure shared session key by performing a pseudorandom function (PRF) using the shared secret, and the Terminal is configured to derive the secure

shared session key by performing the pseudorandom function (PRF) using the shared secret.

28. The WTRU of claim 24, wherein the UICC is configured to establish a secure channel with the Terminal, and the Terminal is configured to establish a secure channel with the UICC.

29. The WTRU of claim 28, wherein the Terminal is configured use the secure channel to perform at least one of an application level generic bootstrapping architecture (GBA) with UICC-based enhancements (GBA_U) procedure or an Authentication and Key Agreement (AKA) procedure.

30. The WTRU of claim 24, wherein the Terminal is configured to:

produce a produced key negotiation parameter;

report the produced key negotiation parameter to the UICC;

receive a received key negotiation parameter from the UICC; and

generate the secure shared session key using the produced key negotiation parameter and the received key negotiation parameter.

31. The WTRU of claim 30, wherein the Terminal is configured to:

determine whether the produced key negotiation parameter is the same as the received key negotiation parameter; and

generate the secure shared session key, on a condition that the produced key negotiation parameter is the same as the received key negotiation parameter.

32. The WTRU of claim 30, wherein the Terminal is configured to:

select a random challenge (RAND) and a sequence number (SQN);

calculate an anonymity key (AK), a message authentication code (MAC), an expected response (XRES), and an expected sequence (XSQN); and

produce the produced key negotiation parameter using the RAND, the MAC, and the XSQN.

33. The WTRU of claim 32, wherein the Terminal is configured to:
calculate the AK using a shared secret and the RAND;
calculate the MAC using the shared secret, the RAND, and the SQN;
calculate the XRES using the shared secret and the RAND; and
calculate the XSQN using the SQN and the AK.

34. The WTRU of claim 10, wherein the Terminal is configured to:
select a nonce;
calculate an authentication value (Tag); and
produce the produced key negotiation parameter using the nonce and the Tag.

35. The WTRU of claim 30, wherein the Terminal is configured to:
select a session key;
calculate an encrypted session key; and
produce the produced key negotiation parameter using the encrypted session key.

36. The WTRU of claim 24, wherein the UICC is configured to:
receive a received key negotiation parameter from the Terminal;
produce a produced key negotiation parameter;
report the produced key negotiation parameter to the Terminal; and
generate the secure shared session key using the received key negotiation parameter and the produced key negotiation parameter.

37. The WTRU of claim 36, wherein the UICC is configured to:
determine whether the produced key negotiation parameter is the same as the received key negotiation parameter; and
generate the secure shared session key, on a condition that the produced key negotiation parameter is the same as the received key negotiation parameter.

38. The WTRU of claim 36, wherein the UICC is configured to:

extract a random challenge (RAND), a message authentication code (MAC), and an expected sequence (XSQN) from the received key negotiation parameter;

calculate an anonymity key (AK), an expected message authentication code (XMAC), and a sequence number (SQN);

determine whether the XMAC is the same as the MAC;

compute a response (RES) using a shared secret and the RAND, on a condition that the XMAC is the same as the MAC; and

produce the produced key negotiation parameter using the RES.

39. The WTRU of claim 38, wherein the UICC is configured to:

calculate the AK using the shared secret and the RAND;

calculate the SQN using the XSQN and the AK; and

calculate the XMAC using the shared secret, the RAND, and the SQN.

40. The WTRU of claim 36, wherein the UICC is configured to:

extract a nonce and a Tag from the received key negotiation parameter;

validate the Tag;

compute an expected authentication value (XTag); and

produce the produced key negotiation parameter using the Xtag.

41. The WTRU of claim 40, wherein the UICC is configured to:

extract an encrypted session key from the received key negotiation parameter;

decrypt the encrypted session key; and

generate the secure shared session key using the decrypted session key.

42. The WTRU of claim 24, wherein the UICC is configured to:

generate a pre-key negotiation parameter; and report the pre-key negotiation parameter to the terminal.

43. The WTRU of claim 24, wherein the Terminal is configured to: receive a pre-key negotiation parameter from the UICC.

44. The WTRU of claim 24, wherein the UICC is configured to perform a Diffie-Hellman key exchange protocol, and the Terminal is configured to perform a Diffie-Hellman key exchange protocol.

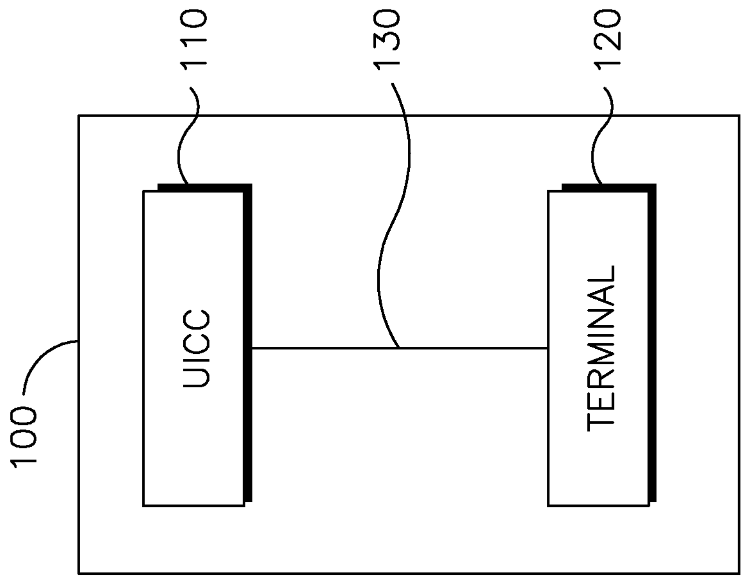

100
110
UICC
130
120
TERMINAL

FIG. 1

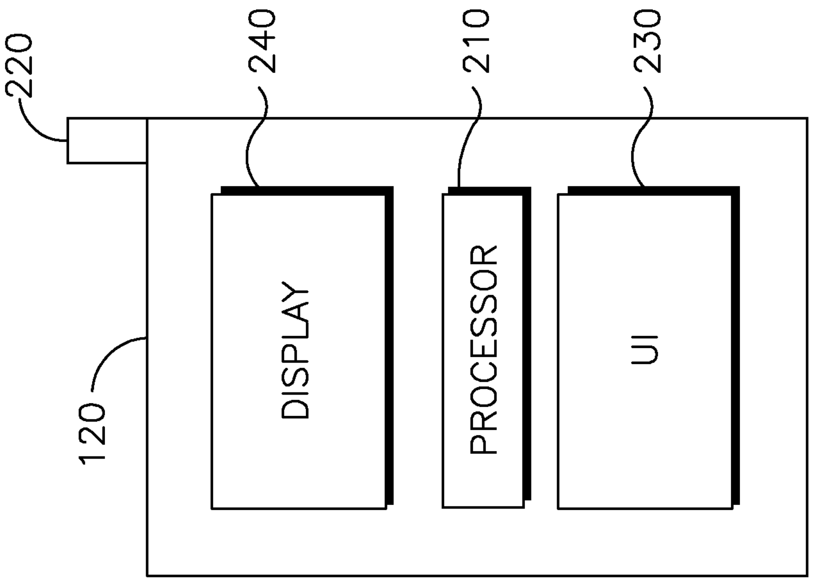

220
120
240
DISPLAY
210
PROCESSOR
230
UI

FIG. 2

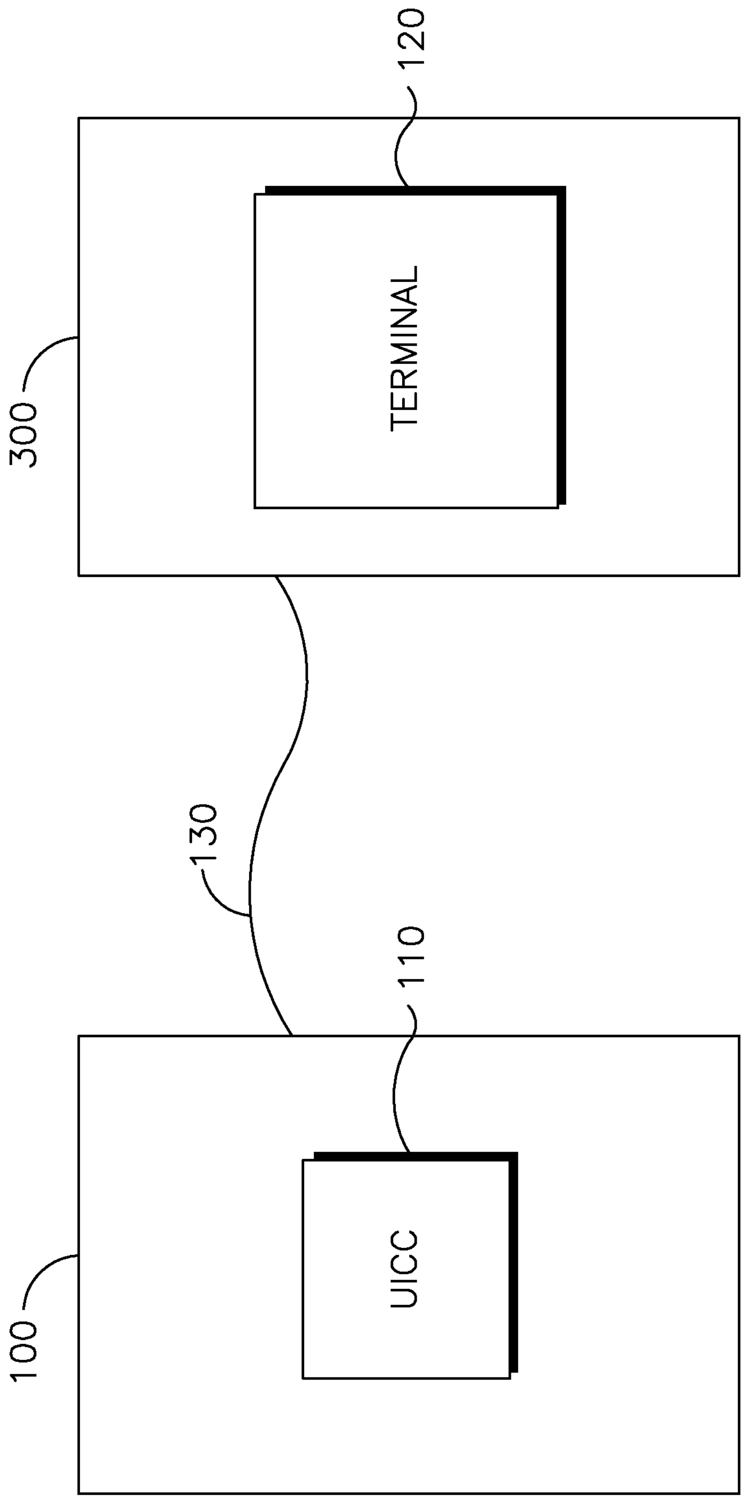


FIG. 3

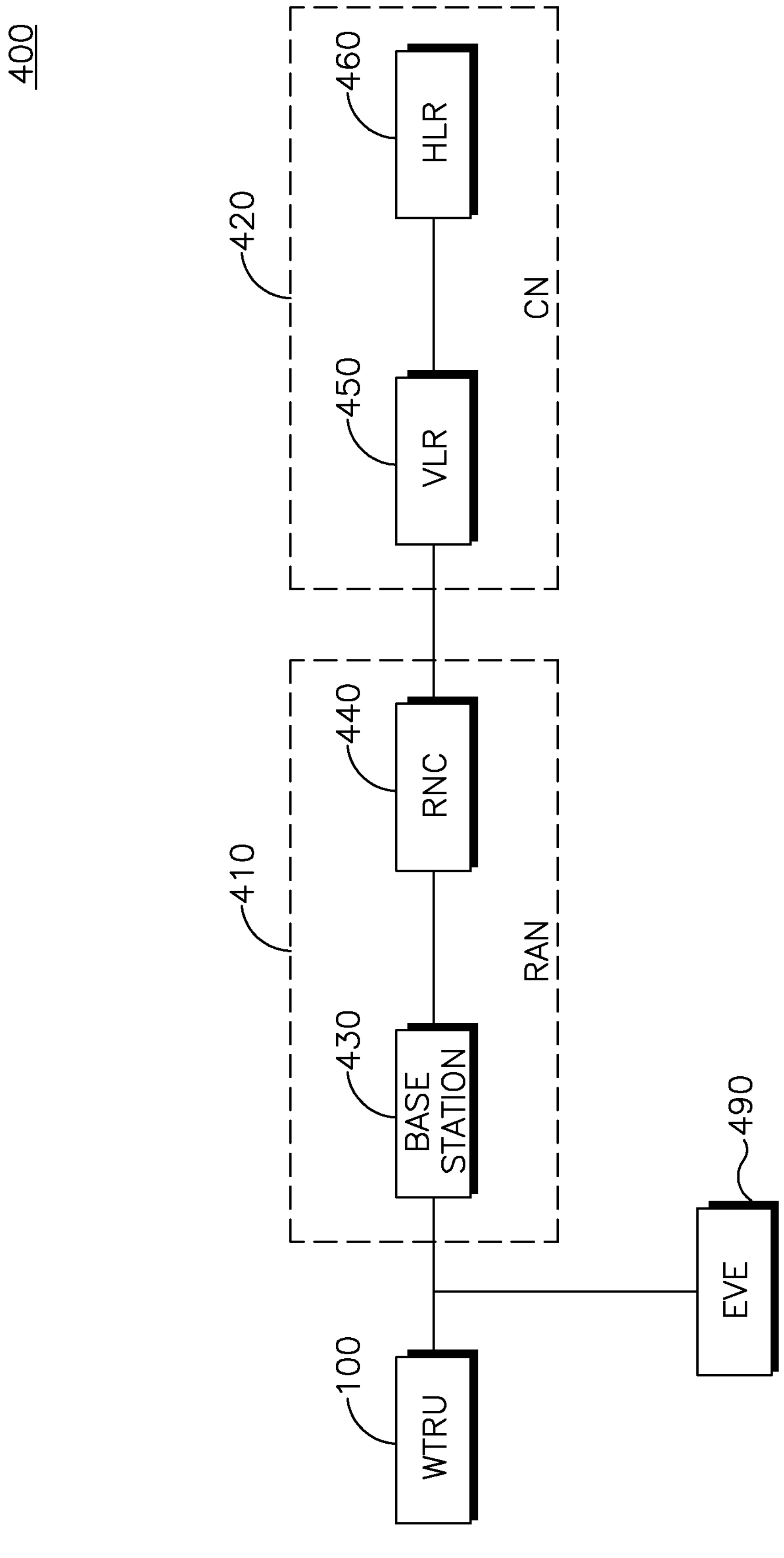

400
WTRU
100
BASE STATION
430
RNC
440
RAN
410
VLR
450
HLR
460
CN
420
EVE
490

FIG. 4

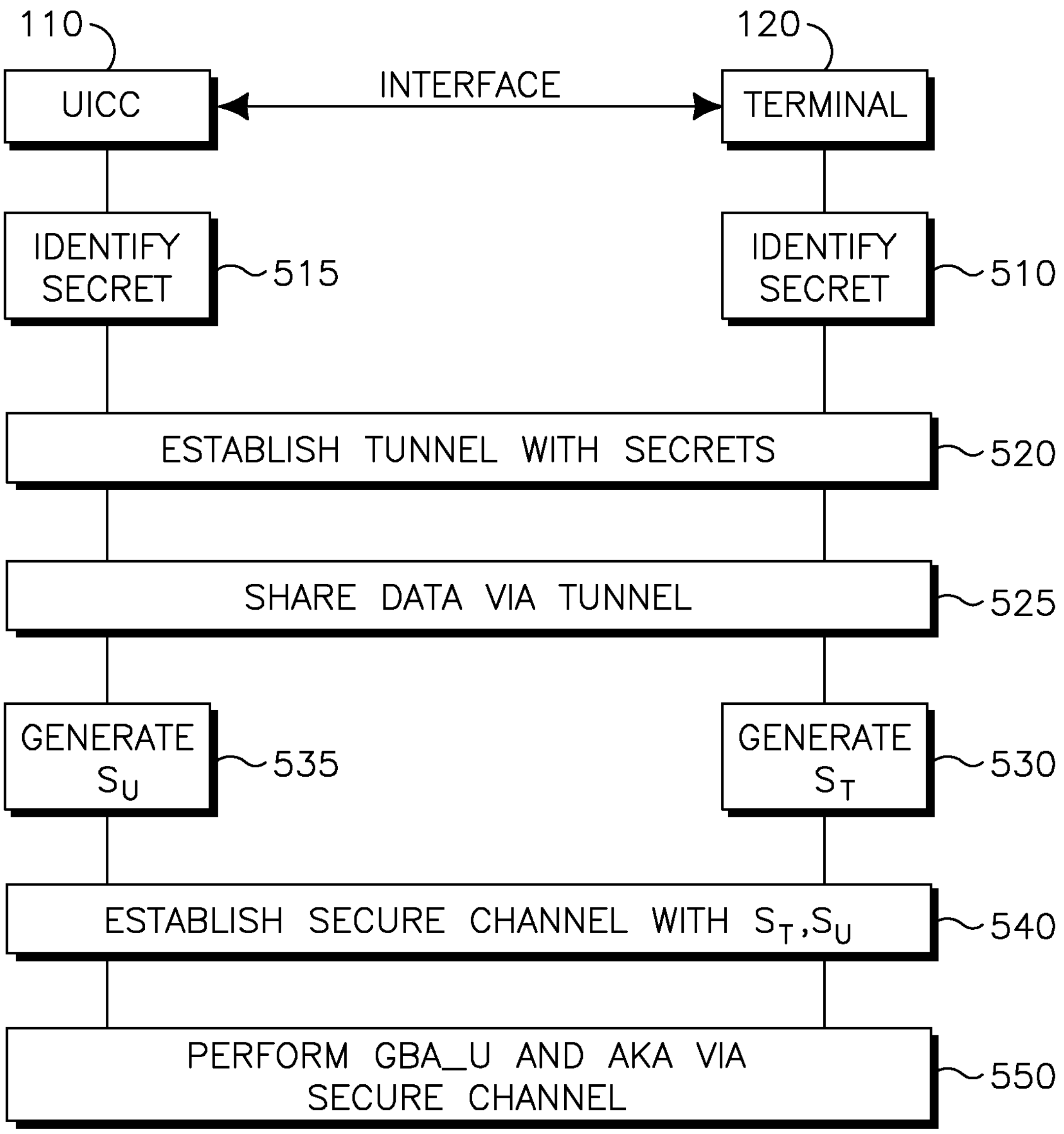

110
UICC
INTERFACE
120
TERMINAL
IDENTIFY SECRET
515
IDENTIFY SECRET
510
ESTABLISH TUNNEL WITH SECRETS
520
SHARE DATA VIA TUNNEL
525
GENERATE S_U
535
GENERATE S_T
530
ESTABLISH SECURE CHANNEL WITH S_T,S_U
540
PERFORM GBA_U AND AKA VIA SECURE CHANNEL
550

FIG. 5

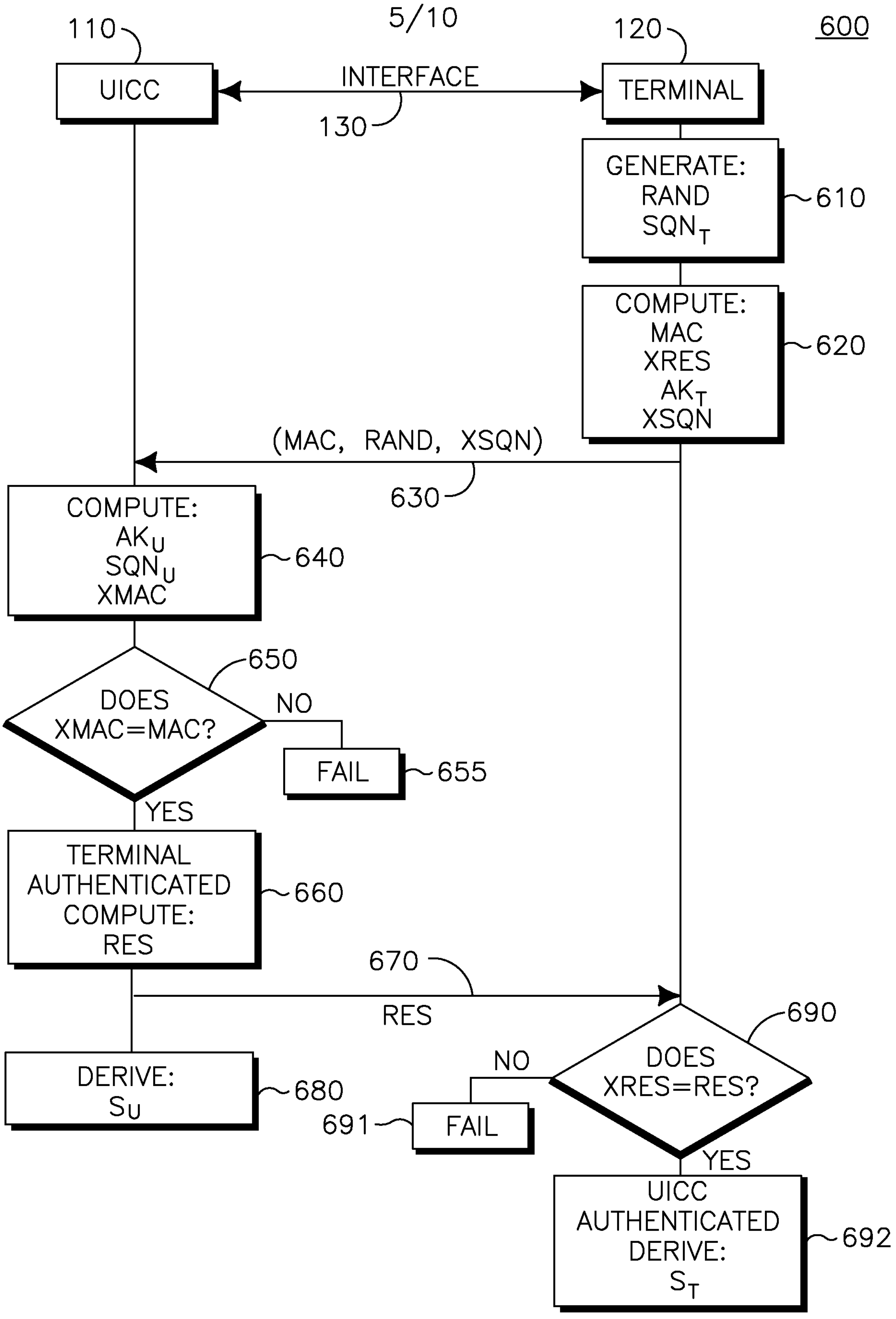

5/10
600
110
UICC
INTERFACE
130
120
TERMINAL
GENERATE: RAND SQN_T
610
COMPUTE: MAC XRES AK_T XSQN
620
(MAC, RAND, XSQN)
630
COMPUTE: AK_U SQN_U XMAC
640
650
DOES XMAC=MAC?
NO
FAIL
655
YES
TERMINAL AUTHENTICATED COMPUTE: RES
660
670
RES
DERIVE: S_U
680
690
DOES XRES=RES?
NO
FAIL
691
YES
UICC AUTHENTICATED DERIVE: S_T
692

FIG. 6

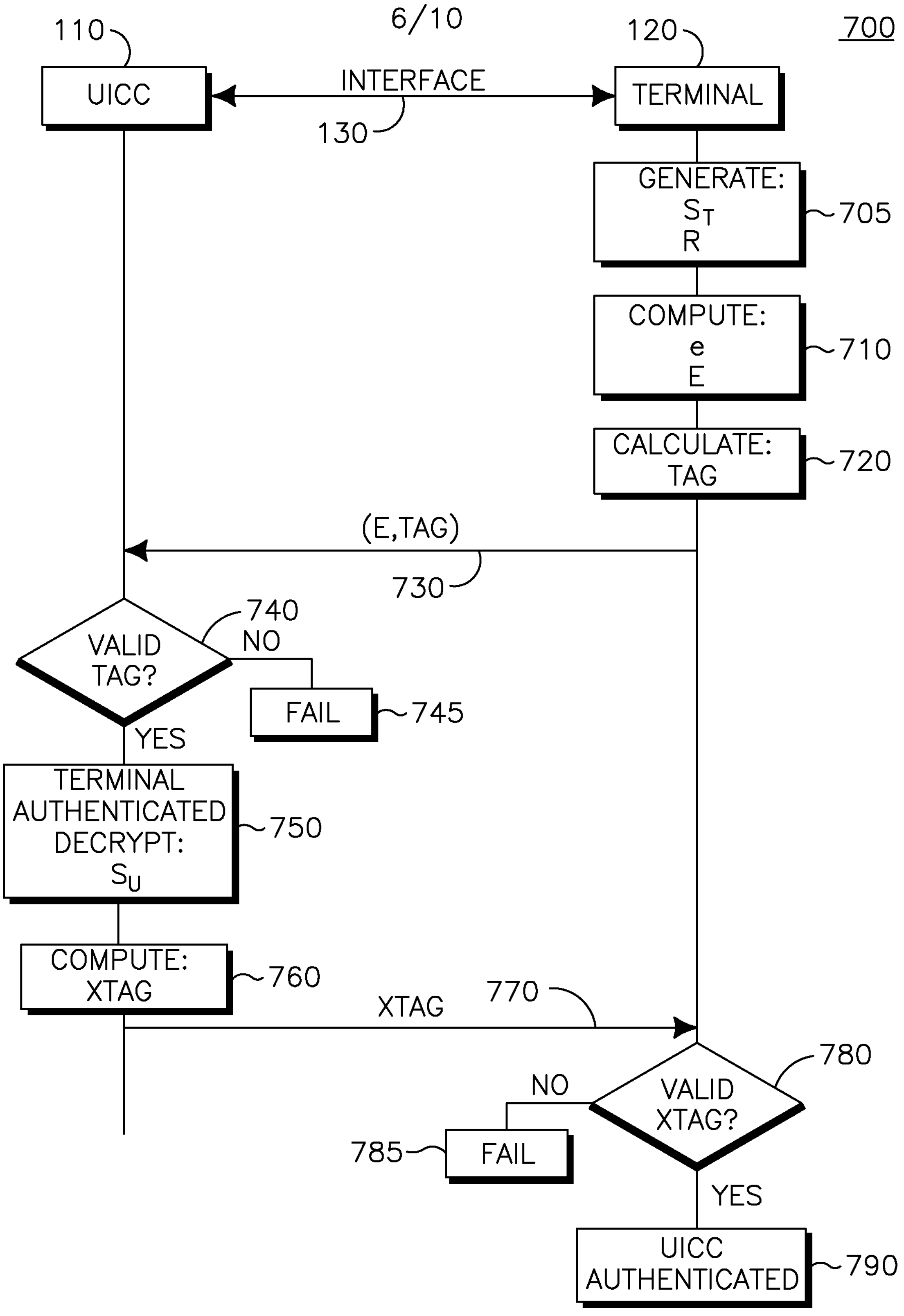

6/10
700
110
UICC
INTERFACE
130
120
TERMINAL
GENERATE: S_T R
705
COMPUTE: e E
710
CALCULATE: TAG
720
(E,TAG)
730
740
VALID TAG?
NO
FAIL
745
YES
TERMINAL AUTHENTICATED DECRYPT: S_U
750
COMPUTE: XTAG
760
XTAG
770
780
VALID XTAG?
NO
FAIL
785
YES
UICC AUTHENTICATED
790

FIG. 7

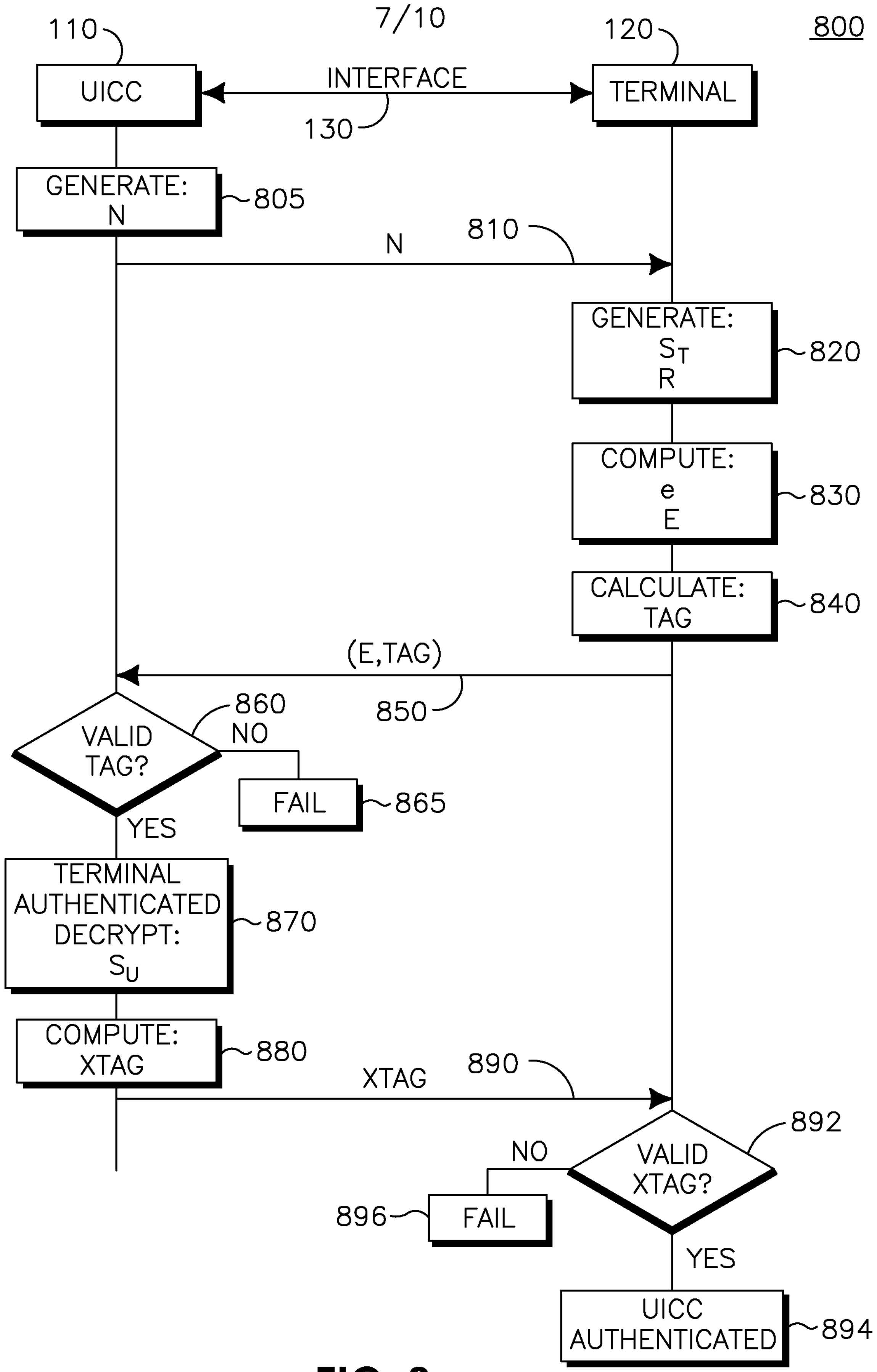

7/10
800
110
UICC
INTERFACE
130
120
TERMINAL
GENERATE: N
805
N
810
GENERATE: S_T R
820
COMPUTE: e E
830
CALCULATE: TAG
840
(E,TAG)
850
VALID TAG?
860
NO
FAIL
865
YES
TERMINAL AUTHENTICATED DECRYPT: S_U
870
COMPUTE: XTAG
880
XTAG
890
VALID XTAG?
892
NO
FAIL
896
YES
UICC AUTHENTICATED
894

FIG. 8

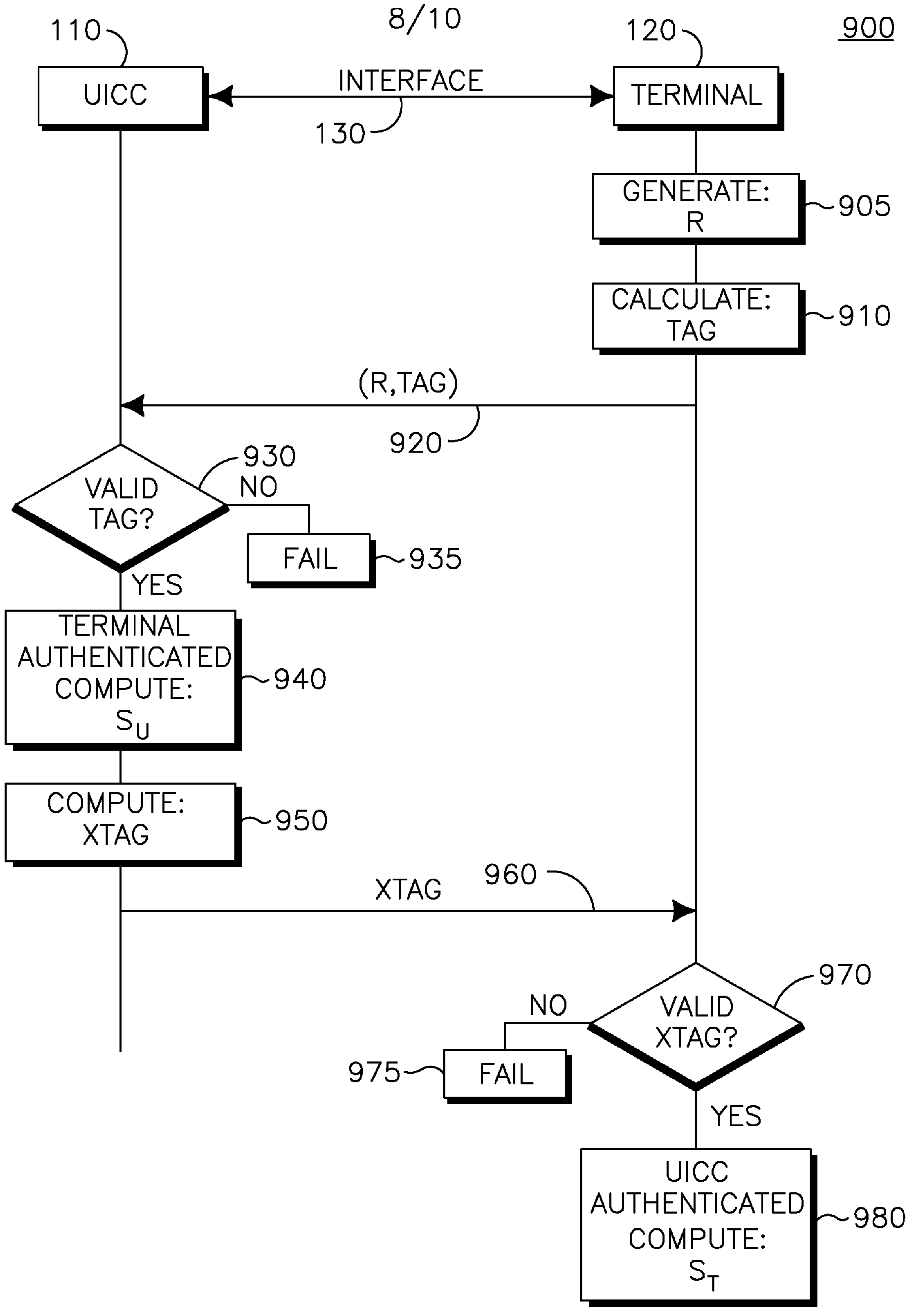

8/10
900
110
UICC
INTERFACE
130
120
TERMINAL
GENERATE: R
905
CALCULATE: TAG
910
(R,TAG)
920
VALID TAG?
930
NO
FAIL
935
YES
TERMINAL AUTHENTICATED COMPUTE: S_U
940
COMPUTE: XTAG
950
XTAG
960
VALID XTAG?
970
NO
FAIL
975
YES
UICC AUTHENTICATED COMPUTE: S_T
980

FIG. 9

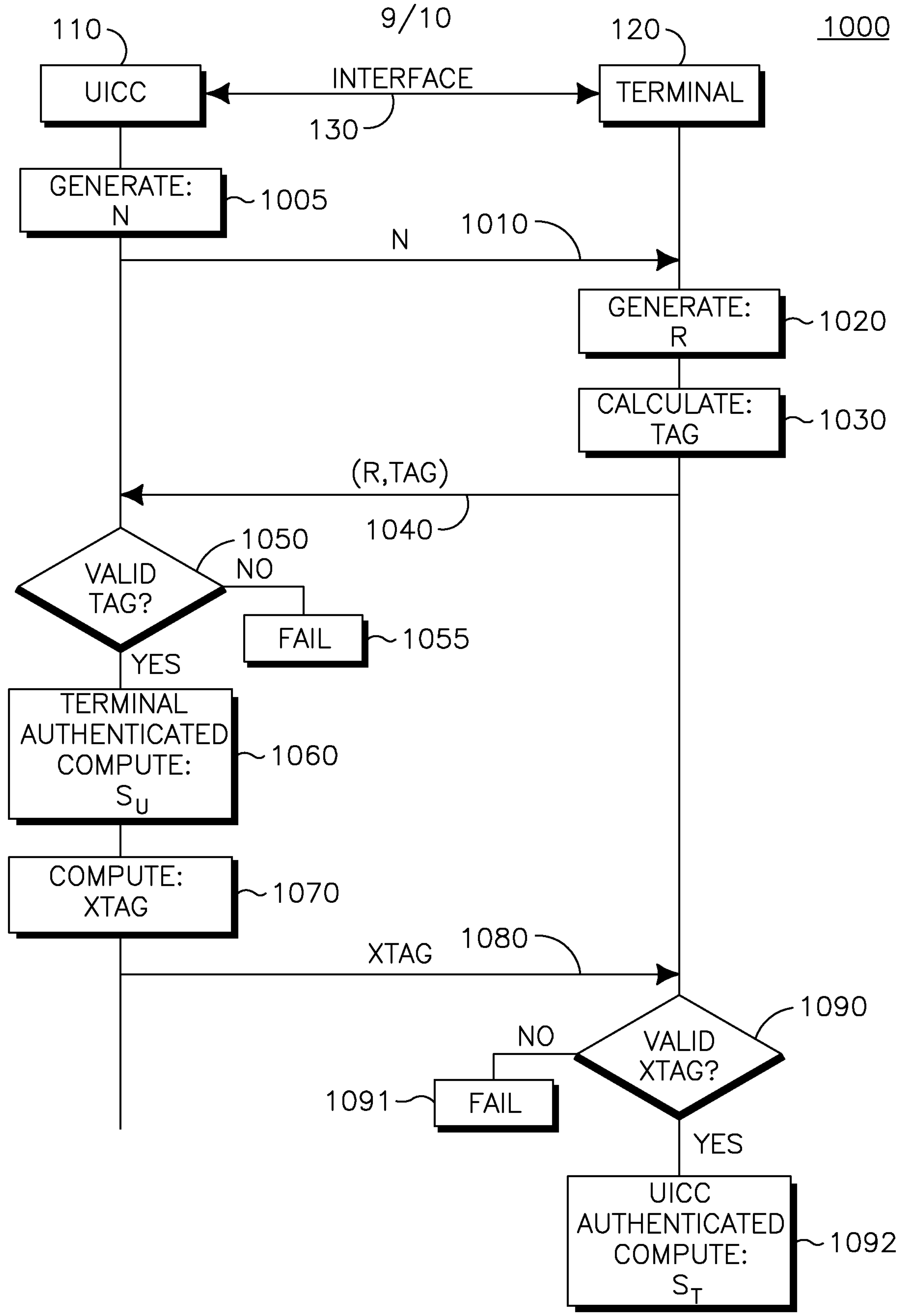

9/10
1000
110
UICC
INTERFACE
130
120
TERMINAL
GENERATE: N
1005
N
1010
GENERATE: R
1020
CALCULATE: TAG
1030
(R,TAG)
1040
VALID TAG?
1050
NO
FAIL
1055
YES
TERMINAL AUTHENTICATED COMPUTE: S_U
1060
COMPUTE: XTAG
1070
XTAG
1080
VALID XTAG?
1090
NO
FAIL
1091
YES
UICC AUTHENTICATED COMPUTE: S_T
1092

FIG. 10

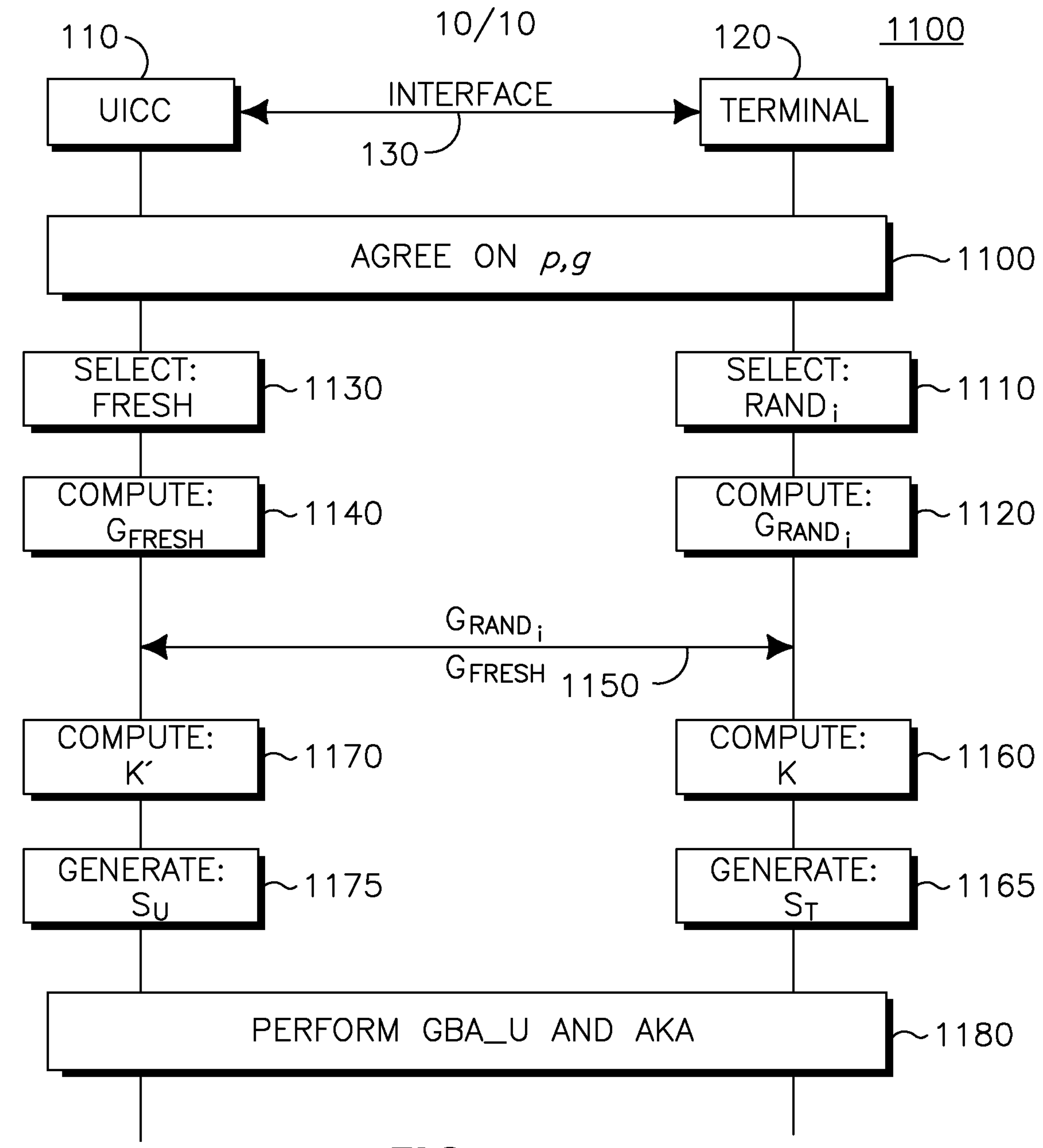

10/10
1100
110
UICC
120
TERMINAL
INTERFACE
130
AGREE ON p,g
1100
SELECT: FRESH
1130
SELECT: RAND_i
1110
COMPUTE: G^{FRESH}
1140
COMPUTE: G^{RAND_i}
1120
G^{RAND_i}
G^{FRESH}
1150
COMPUTE: K′
1170
COMPUTE: K
1160
GENERATE: S_U
1175
GENERATE: S_T
1165
PERFORM GBA_U AND AKA
1180

FIG. 11