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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 December 2005 (08.12.2005)

PCT

(10) International Publication Number
WO 2005/117334 A1

(51) International Patent Classification[7]: H04L 9/18, 9/32, 9/20

(21) International Application Number: PCT/CA2005/000817

(22) International Filing Date: 31 May 2005 (31.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2,469,426 31 May 2004 (31.05.2004) CA
2,470,697 10 June 2004 (10.06.2004) CA

(71) Applicant *(for all designated States except US)*: **NATIONAL RESEARCH COUNCIL OF CANADA** [CA/CA]; 1200 Montreal Road, BLDG. M-58, Ottawa, Ontario K1A 0R6 (CA).

(72) Inventors; and

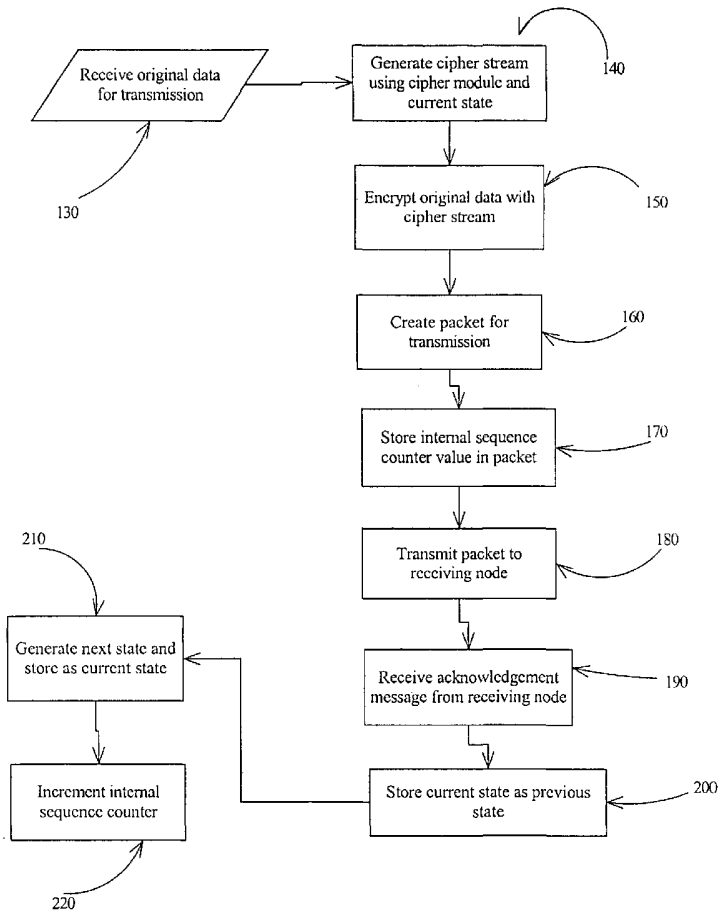
(75) Inventors/Applicants *(for US only)*: **SRINIVASAN, Kannan** [CA/CA]; 70 Towerview Place, Apt. 2014, Sydney, Nova Scotia B1S 3B9 (CA). **MICHELL, Stephen** [CA/CA]; 1969 Rosebella Avenue, Ottawa, Ontario K1T 1G6 (CA).

(74) Agent: **CASSAN MACLEAN**; Suite 401, 80 Aberdeen Street, Ottawa, Ontario K1S 5R5 (CA).

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

[Continued on next page]

(54) Title: STATE BASED SECURE TRANSMISSION FOR A WIRELESS SYSTEM



(57) Abstract: Methods, devices, and systems for ensuring state synchronization between two communicating nodes which use a state based stream cipher. Two nodes in a communications network set up two one way links between them, one from a first node to a second node and one from the second node to the first node. Each link has its own resources at each node and each link has its own counter at each node. When communicating, the transmitting node encrypts data using a state based stream cipher and a key shared by both nodes along with an encryption state. The transmitting node transmits the encrypted data to the receiving node along with a counter value corresponding to the link used for the transmission. The receiving node, after receiving the encrypted data, confirms the received counter value with its own counter for that link. After confirmation, the receiving link decrypts the encrypted data using the state based stream cipher, the shared key, and the encryption state. The two nodes are state synchronized such that each node, when using the state based stream cipher and the same key, produce the same cipher stream. In the event synchronization is lost, on possible resynchronization procedure involving a 4 part handshaking process between the two nodes and which involves an exchange of nonces (also referred to as markers) may be used.

WO 2005/117334 A1

(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, HU, IE, IS, IT, LT, LU, MC, NL, 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:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

STATE BASED SECURE TRANSMISSION FOR A WIRELESS SYSTEM

Field of Invention

[0001] The present invention relates to transmission methods for wireless communications. More specifically, the present invention relates to methods and devices for secure communications between two nodes in a network using state based stream ciphers.

Description of the Related Prior Art

[0002] The recent revolution in communications has given rise to an explosion of developments in wireless communications. The ubiquity of cellular telephones and wireless networks is merely one offshoot of this growth. With the boom in wireless communications, there has been a corresponding need for secure wireless communications. To this end, a multitude of protocols, standards, and encryption methods have been formulated. However, most of these protocols, standards, and methods have shortcomings that have, at one point or another, been glaringly revealed.

[0003] Wireless devices need a robust security protocol that does not require excessive resources. Excessive resources requirement may lead to performance degradation of the overall network and may lead to depletion of energy that may be limited in some devices such as battery-operated wireless sensors. This security protocol must be robust against active attacks and must include countermeasures to such active attacks while also providing message integrity to valid encrypted messages. For example, the current widely used wireless standard IEEE 802.11 does not prevent an intruder from listening to message exchanges, fraudulently participate

in message exchanges, disrupt ongoing communication and easily break encryption codes (wired equivalent privacy (WEP)).

[0004] WEP is a symmetric encryption scheme in which a WEP key is known or shared between two communicating nodes. WEP uses the well-known RC4 algorithm to do per packet encryption. RC4 algorithm is a stream cipher scheme, [see Mantin I., Analysis of the Stream Cipher RC4. Weizmann Institute of Science, Nov. 2001., hereinafter Mantin 2001] in which the data is encrypted by XORing data with the cipher stream generated by RC4 from an RC4 seed. WEP concatenates the WEP key (40 or 104 bits) and the initialization vector (IV) (24 bits), as the RC4 seed. For every new RC4 seed, RC4 reinitializes its states using key-scheduling algorithm (RC4-KSA). After RC4-KSA, RC4 generates the cipher stream using pseudo random generation algorithm (RC4-PRGA). Since the IV is sent in every packet, WEP carries out RC4-KSA and RC4-PRGA for every packet.

[0005] WEP is identified as being weak in a number of areas which make its continued use as a security mechanism for wireless untenable. These weaknesses are summarized here. Walker [Walker J., 802.11 Security Series - Part II: The Temporal Key Integrity Protocol (TKIP), Intel Corporation, 2002 hereinafter Walker 2002] identify weak key issues and shows that the forgery attacks, replay attacks and bit flipping attacks let active attackers spoof networks, make invalid packets seem valid and can derive the shared key from such attacks. These attacks make WEP easy to crack and so a new proposal was needed to provide enhanced security. IEEE 802.11 task group i (802.11 TGi) is working on a new standard (802.11i) [see Draft Amendment to STANDARD FOR Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: MAC Security Enhancements, IEEE Std. 802.11i/D7.0, Oct. 2003 hereinafter Draft 2003] for security in 802.11 networks. There are two proposals from 802.11 TGi: one for legacy devices and the other for future devices. WPA is a subset of 802.11i and claims to be future compatible with 802.11i. WPA for legacy devices is called WPA 1.0 and WPA for future devices is called WPA 2.0.

[0006] It became apparent as IEEE 802.11 was being standardized that WEP was not going to provide the wired equivalent privacy desired. The following specific defects within WEP were noted:

IV Collision: Since, WEP uses concatenation of WEP key and IV as RC4 seed, the generated cipher stream will be the same for same IVs if the WEP key does not change. This means that if an attacker can predict the unencrypted data (plain text) for a given IV, then he could determine what the corresponding cipher stream is. Once the same IV is repeated, he could make use of the previously identified cipher stream to predict the plain text, corresponding to the repeated IV.

Weak Key: Due to the nature of RC4-KSA and RC4-PRGA algorithms used within RC4, there exist some weak keys, in which specific pattern in first few bytes of the RC4 seed will result in a corresponding pattern in the first few bytes of the RC4 cipher stream [Walker 2002]. This property can be used to derive the RC4 seed and hence the WEP key after monitoring the IVs and their corresponding cipher streams.

Bit Flipping Attack: WEP uses a CRC-32 algorithm to compute integrity check value (ICV) on the plain text data. This ICV is concatenated with plaintext data and then is encrypted with cipher stream. Known weaknesses in CRC-32 algorithm make it easy to modify data and perform corresponding changes to ICV such that this modification gets unnoticed. An intruder can perform such modifications to previously sent packets and send them again. The receiving node will accept that packet as ICV will succeed and will forward that packet to the final destination, permitting the attacker to interfere with higher-level protocols or generate and analyze known error responses from the destination.

Forgery Attack: WEP has no protection against redirection of encrypted packets. This can happen when receiver address (RA) or destination address (DA) field within each frame is changed. Changing one of these fields will result in delivery of the frame to a wrong destination. Changing source address (SA) or transmitter address (TA) field is also possible and may result in redirection of the responses.

Replay Attack: Any intruder that intercepts any WEP encrypted frames can resend the same frame later. The access point will forward such frames, as frame replays succeed decryption.

[0007] In order to repair the defects identified above, IEEE 802.11i task group developed two proposals [Draft 2003] for security for 802.11 devices. IEEE 802.11i's first proposal for 802.11 legacy devices (Wi-Fi Protected Access WPA 1.0) encapsulates WEP functionalities by temporal key integrity protocol (TKIP). IEEE 802.11i's second part (WPA 2.0) uses Advanced Encryption Standard (AES) and requires change in hardware. WPA 1.0 is a subset of the former and WPA 2.0 is a subset of the latter.

[0008] WPA 1.0 provides message integrity code (MIC) for data integrity to protect data from forgery and bit-flipping attacks. Michael is the MIC algorithm used in WPA 1.0 that adds an 8 octet field called MIC key to every MAC service data unit (MSDU). Michael is still vulnerable to replay attacks. TKIP provides extended initialization vector (IV), adding 4 octets of additional overhead to every MAC payload data unit (MPDU), to protect Michael from replay attacks.

[0009] WPA can work with 802.1x and extensible authentication protocol (EAP) to provide per-node session keys. This mode of operation requires hard-to-maintain RADIUS server in any 802.11 networks. WPA 1.0 has another mode called pre- shared key (PSK) that is being used in many SOHO user nodes in which all the nodes carry static base key. This mode has significant security issues as well [see Moskowitz R., Simple Secrets/Simple Security, ICSA Labs, 2003 hereinafter Moskowitz 2003]. It also suffers from attacks from any insider who has the same key.

[00010] WPA, as WEP, still reinitializes RC4 states by running key scheduling algorithm (RC4-KSA) and then carrying out the pseudo random number generation algorithm (RC4-PRGA) to generate encryption cipher stream, for every packet. Thus, WPA due to its complexity, extensive processing cost and high overheads (8 octet MIC key + 4 octet extended IV) is unsuitable for battery-operated devices and SOHO users.

[00011] It is well known that WEP has been broken, and that tools exist on the Internet to decipher messages sent using WEP after reading as few as 20,000 messages. This decryption relies upon an improper use of RC4 [Mantin 2001]. Mantin identifies three significant RC4 weaknesses which, when RC4 is used improperly, make it trivial to attack. The most significant weakness is at the start of a cipher stream, which would let an attacker determine a key in 10^6 packets. WEP aggravates this problem by using a weak initialization vector and directly concatenating the IV to the shared key. WPA 1.0 still suffers from this RC4 weakness but uses extended IV, without direct concatenation to the key, to make it harder for an attacker to figure out the actual key.

[00012] WPA 1.0 still suffers from the basic RC4 weakness but uses extended IV, without direct concatenation to the key, to make it harder for an attacker to figure out the actual key. This is not to say that RC4 is by nature susceptible to easy attacks. A single RC4 pseudo random sequence forms a strong encryption stream.

[00013] All of the public encryption schemes for IEEE802.11 use in-the-clear control, and management messages, such as acknowledgements, association request messages, authentication and deauthentication messages. This use can compromise the

integrity of encrypted data, or can open the BSS to other forms of attack, such as replay attacks, modified message replay attacks and fake management messages when WEP is used. WPA 1.0 uses encrypted message integrity fields (MIC key) to identify fake messages, but at a cost of increased overhead and reduced performance.

[00014] Insiders are a significant threat to security systems. Shared key systems like WEP and WPA-PSK give insiders open access to all messages exchanged and permit active attacks.

[00015] For IEEE802.11 nodes powered by a limited power supply (eg. batteries), more complex encryption equates to increased energy consumption, processing times, and decreased node lifetime. RC4-KSA carried out on every packet adds an extra processing cycle in WEP and WPA 1.0, while WPA's extra fields (TSC and MIC Key) and AES encryptions add even more processing.

[00016] Running RC4-KSA at packets of 250 octets doubles the cost of encryption for WEP, and increases it manifolds for packets smaller than 250 bytes. The extra overheads in WPA 1.0 for the Message Integrity Check (MIC) and WPA 2.0 for AES make them even costlier.

[00017] Since the average network has the mean packet size less than 200 octets [Walker 2003] and sensor networks even smaller packets, these protocols can be quite expensive.

[00018] Based on the above, there is therefore a need for a more secure method of communicating between two nodes in a network. Ideally, such a method would provide security without requiring an onerous hardware retooling. Furthermore, it would be ideal to provide a method that uses well-known and well-understood underlying encryption and decryption methods.

Summary of the Invention

[00019] In order to overcome the deficiencies of the prior art there is provided methods, devices, and systems for ensuring state synchronization between two communicating nodes which use a state based stream cipher. Two nodes in a communications network set up two one way links between them, one from a first node to a second node and one from the second node to the first node. Each link has its own resources at each node and each link has its own counter at each node. When communicating, the transmitting node encrypts data using a state based stream cipher

and a key shared by both nodes along with an encryption state. The transmitting node transmits the encrypted data to the receiving node along with a counter value corresponding to the link used for the transmission. The receiving node, after receiving the encrypted data, confirms the received counter value with its own counter for that link. After confirmation, the receiving link decrypts the encrypted data using the state based stream cipher, the shared key, and the encryption state. The two nodes are state synchronized such that each node, when using the state based stream cipher and the same key, produce the same cipher stream. In the event synchronization is lost, on possible resynchronization procedure involving a 4 part handshaking process between the two nodes and which involves an exchange of nonces (also referred to as markers) may be used.

[00020] In accordance with one aspect of the invention there is provided a method of communicating between a pair of nodes in a communications network, the method comprising:

- using a state based stream cipher to encrypt and decrypt data between said nodes

- maintaining one set of uplink states for an uplink connection between said pair of nodes and one set of downlink states for a downlink connection between said pair of nodes, said states being used for encryption and decryption of data with said state based stream cipher

- maintaining at least one common marker value between said pair of nodes, said marker value being used for authentication and synchronization of states between said pair of nodes

- maintaining separate keys for said uplink and downlink connections between said communicating nodes, said separate keys being used for said encryption and decryption of data with said state based stream cipher

- maintaining a separate counter value for each one of said downlink and uplink connections between said pair of nodes, said sequence counter value being used to ensure that a first state used to encrypt data at a first node is the same state as a second state used to decrypt data at a second node.

[00021] In accordance with another aspect of the invention, there is provided In a communications network having a first node and a second node, each of which is using a state based stream cipher to encrypt and decrypt data transmitted between said

first node and said second node, a method of processing transmitted data received from said first node by said second node, the method comprising:

- receiving at said second node transmitted data from said first node, said transmitted data including encrypted data encrypted by said first node using said state based stream cipher using a first transmitting current state and a common key, said common key being common to said first node and said second node, a copy of said common key being stored at said first node and said second node, said transmitted data including a sequence counter value, said sequence counter value being equal to a first internal sequence counter value stored at said first node;

- comparing at said second node said sequence counter value with an internal previous sequence counter value stored in said second node

- in the event said sequence counter value is one more than said internal previous sequence counter value, decrypting said encrypted data at said second node using a receiving current state stored at said second node and said copy of said common key stored in said second node

- in the event said encrypted data is successfully decrypted by said second node, sending an acknowledgement message from said second node to said first node

- incrementing said internal previous sequence counter at said second node

- storing said receiving current state as a previous state at said second node

- determining a receiving next state at said second node

- storing said receiving next state as said receiving current state at said second node

[00022] In another embodiment of the invention, there is provided a method of communicating between a first node and a second node in a communications network, the method comprising:

- at said first node, encrypting data for transmission to said second node using a state based stream cipher with a transmitting current state and a key shared by both first and second nodes,

- transmitting encrypted data from said first node to said second node, said encrypted data being transmitted with a sequence counter value from said first node

- at said second node, receiving said encrypted data and said sequence counter value

- ensuring a receiving current state in said second node is the same as said transmitting current state using said sequence counter value

- in the event said receiving current state is the same as said transmitting current state, decrypting said encrypted data using said state based stream cipher with said receiving current state and said key.

[00023] In yet another embodiment, the present invention provides a first node for use in a communications network, said first node communicating with a second node using a state based stream cipher, said first node generating a first single cipher stream for encrypting data for transmission to said second node, said first node also generating a second single cipher stream for decrypting data received from said second node, said first and second nodes being state synchronized with each other to enable encryption and decryption of data.

[00024] Further features and advantages of the invention will be apparent from the detailed description which follows together with the accompanying drawings.

Brief Description of the Drawings

[00025] A better understanding of the invention will be obtained by considering the detailed description below, with reference to the following drawings in which:

Figure 1 is a diagram illustrating how stream ciphers are used;

Figure 2 illustrates the encryption process used by a transmitting node;

Figure 3 illustrates the decryption process used by a receiving node;

Figure 4 illustrates the fields in a packet according to one aspect of the invention;

Figure 5 is a block diagram illustrating two nodes A and B communicating by means of a method according to the invention;

Figure 6 is a flowchart illustrating the steps executed by a transmitting node when communicating with a receiving node;

Figure 7 is a flowchart illustrating the steps executed by a receiving node when communicating with a transmitting node;

Figure 8 is a block diagram illustrating the data flow for an RC4 implementation of the invention in a transmitting node;

Figure 9 is a block diagram illustrating the data flow for an RC4 implementation of the invention in a receiving node;

Figure 10 is a dataflow/message flow diagram illustrating the direction in which administrative messages are exchanged between a transmitting node and a receiving node during the authentication process; and

Figure 11 is a dataflow/message flow diagram illustrating the direction in which administrative messages are exchanged between a transmitting node and a receiving node during the resynchronization process.

Description of the Preferred Embodiment

[00026] Referring to Figure 1, a diagram explaining how stream ciphers are used is illustrated. A transmitting node 10 generates, using a stream cipher, ciphertext or encoded data which is transmitted to a receiving node 20 using well-known transmission techniques. At the receiving node 20, the encoded data is decoded, again using a stream cipher, to retrieve the original data.

[00027] Referring to Figure 2, the encryption process used by the transmitting node 10 is illustrated. The original data 30 is received as a stream of bytes 30A, 30B, 30C, ... 30H. The transmitting node 10, using a key 40 and a starting state 45, generates a cipher stream 50, comprised of cipher bytes 50A, 50B, 50C, ... 50H, using a cipher module 60A. Each one of the original bytes 30A ... 30H is then paired with a cipher byte 50A ... 50H. A logical XOR operation is then performed on each pair of original byte 30A ... 30H and cipher byte 50A ... 50H to result in an encoded byte or encrypted byte 70A ... 70H, all of which make up the encrypted data or encoded data 70. The encoded data 70 is then transmitted, along with other data, by the transmitting node 10 to receiving node 20.

[00028] Referring to Figure 3, the decryption process used by the receiving node 20 is illustrated. The encoded data 70 is received by the receiving node 20 as the same stream of bytes 70A ...70H produced by the transmitting node 10. Using a copy of the same key 40 as used by the transmitting node 10 and the same starting state 45, the receiving node 60 uses a similar cipher module 60B to generate the same cipher stream 50 comprised of cipher bytes 50A ... 50H. Each of the cipher bytes 50A ... 50H is then paired with an encoded byte 70A ... 70H and an XOR operation is performed on each resulting cipher byte/encoded byte pair. The result of the XOR operation is the original data 30 composed of bytes 30A ... 30H.

[00029] The cipher modules 60A, 60B must both produce the same cipher stream and the same cipher bytes for the scheme to function properly. Thus, for each encoded byte 70A produced by the XOR operation between original byte 30A and cipher byte 50A, the cipher module 60B must produce the same cipher byte 50A. The two cipher modules 60A, 60B must therefore not only produce the exact same cipher stream but must also produce it in a stepwise synchronized manner such that each cipher byte is properly paired with either the corresponding original or the corresponding encoded byte. If cipher module 60B produces the same cipher stream as cipher module 60A but the two streams are not stepwise synchronous (i.e. encoded byte 70A is paired with cipher byte 50B, encoded byte 70B is paired with cipher byte 50C, etc., etc.), then the XOR operation at the receiving node 20 will not produce the desired original data 30.

[00030] As noted above, one stream cipher scheme which may be used by the cipher modules 60A, 60B is the RC4 stream cipher scheme. While other stream cipher schemes may be used, the RC4 scheme will be used as the exemplary example for the rest of this document. For the RC4 scheme, the key 40 would be the base key used to generate the cipher stream 50. In the RC4 scheme, the base key is used as the seed for the key scheduling algorithm (RC4-KSA) which initializes the RC4 state and which is input to the pseudorandom number generator (RC4-PRGA) in cipher modules 60A, 60B. To ensure that the cipher modules 60A, 60B produce the same cipher stream for the same encoded data, the two cipher modules must be state synchronized. This means that, using the base key Y, cipher module 60A in state x will produce cipher byte X_A and cipher module 60B, again using base key Y, in state x will also produce byte X_B where X_A X_B. The starting state 45 is the state which the cipher modules use in conjunction with the base key to generate the cipher stream. Each iteration of the process (RC4-PRGA) that produces a single cipher byte changes the state of the cipher module. Thus, if cipher module 60A starts at state s_0 and runs through 8 iterations, then the end state is s_7. The starting state for the next iteration (to produce the next cipher byte) is therefore s_7. This means that cipher module 60B, to successfully decode the encoded data, must also start at state s_0 and end at state s_7. The end state can thus be stored at each node for use when the next data, either encoded or original, needs to be processed. But it should be noted that for the next data, s_7 is directly fed as an input to the pseudorandom generator, thereby bypassing the key scheduling algorithm. Thus, the key scheduling algorithm which initializes

RC4 state to s_0 based on a given base key is used only whenever the base key is changed. Otherwise, only the pseudorandom number generator is used in encoding subsequent data using the updated RC4 state. The base key is therefore irrelevant after the initialization of the RC4 state to s_0. This leads to the need for strong synchronization of the RC4 states between the nodes for subsequent data.

[00031] Since base key 40 is already shared by both the transmitting node 10 and the receiving node 20, both nodes only need to ensure that they be state synchronized. This is done by using a counter value which is, again, shared between the two nodes 10, 20. Both cipher modules 60A, 60B work similarly such that, by using the same base key and the same initialization or starting state they will produce the exact same cipher stream in the same manner. Thus, cipher module 60A, using base key K and initialization state S, will produce cipher bytes B_1, B_2, B_3, ... B_x after which the module 60A will be in state x. Cipher module 60B, using the same base key K and initialization state S, will also produce the same cipher bytes B_1, B_2, B_3, ... B_x. after which module 60B will be in state x. As long as each cipher module has the same starting state, along with the same initialization variables, then if both cipher modules produce x cipher bytes, they will both be in the same state and are therefore state synchronized. The counter value in the transmitting node 10, in a sense, is a packet counter but, in some implementations, it can be used to keep track of the number of bytes encoded and thereby the number of cipher bytes produced. Similarly, the counter value in the receiving node 20 keeps track of the number of packets decoded. The counter value in the receiving node 20 may also be used, in some implementations, to keep track of the number of cipher bytes produced at the receiving node 20. As long as the counter values at both the receiving and the transmitting nodes 20, 10 are the same or are within the expected values, the two cipher modules 60A, 60B (and hence the transmitting and receiving nodes 10, 20) are state synchronized. As noted above, other implementations may use the counter values to keep track of the number of groups of cipher bytes produced instead of the actual number of cipher bytes produced or encoded/original bytes processed.

[00032] The transmitting node 10 therefore sends, in a packet switching system, the packet 80 of Figure 4 to the receiving node 20. The packet 80 of Figure 4 has the following fields : a MAC (media access control) header field 80A, a counter field 80B, an encrypted data field 80C, and an integrity check field 80D (also encrypted). The contents and function of the MAC header field 80A is well-known in

the art. The counter field 80B contains a sequence counter value or a counter value equal to an internal sequence counter value in the transmitting node. The encrypted data field 80C contains the encrypted data while the integrity check field 80D contains an integrity check value which may be used to ensure that the contents of the packet has been properly received. Other fields may be added to the packet as the implementation requires.

[00033] From the above, it should be clear that each of the transmitting node 10 and the receiving node 20 has a separate counter which produces a counter value for tracking the encoding and decoding of data. The transmitting node 10 transmits its packet, with its counter value, to the receiving node 20. The receiving node 20, based on its own internal counter expects a certain specific value as the counter value in the packet. If the counter value in the packet is the value expected by receiving node 20, then the encoded data can be decoded by the receiving node 20. If the counter value in the packet is what is expected by the receiving node 20, this means that the cipher modules 60A, 60B are state synchronized. It should be noted that the counters in the nodes are incremented after the respective data is processed. Thus, the transmitting node 10 increments its internal sequence counter after the node processes a data packet for transmission to the receiving node 20. Similarly, the receiving node 20 increments its internal counter value only after the node has decoded a packet which has been received.

[00034] The above scheme describes a one way link between the transmitting node 10 and the receiving node 20. Referring to Fig 5, two nodes A and B (denoted by reference numbers 90 and 100 respectively) are communicating using two instances of the above scheme. For simplicity, the link where data is transmitted from node A 90 to node B 100 will be defined as the uplink link 105 while the link where data is transmitted from node B 100 to node A 90 will be defined as the downlink link 107. As can be seen from the Figure, node A 90 has an uplink internal counter 110A and a downlink internal counter 120A. Similarly, the node B 100 has an uplink internal counter 110B and a downlink internal counter 120B. As can be imagined, each of the nodes A and B is both a transmitting node and a receiving node. For the uplink link, the node A 90 is the transmitting node while the node B 100 is the receiving node. For the downlink link, the node B 100 is the transmitting node and the node A 90 is the receiving node. Thus, if data is being transmitted from node A to node B, node A encrypts or encodes the data and transmits this encoded data to node

B using the uplink link. Node B, once it receives and decodes the data, then acknowledges receipt of the data by sending an acknowledgement message to node A by way of the downlink link. For clarity, it should be clear that each node has resources set aside for each link. Thus, the uplink link uses counter 110 A in node A and counter 110 B in node B. Similarly, the base key and initialization states used for the uplink and downlink links are different. For convenience, the base keys for the two links may be determined at the same time and may be changed at the same time.

[00035] Once each link has been initialized (by initializing each end of the link with the proper base key and initialization state), data can be transmitted going in either direction. For a transmitting node, the steps executed when communicating with a receiving node is illustrated in the flowchart of Fig 6. Referring to Fig 6, the initial step is step 130 of receiving the original data which is to be encoded for transmission to the receiving node. Step 140 generates the cipher stream using the current state and the base key with the cipher module which uses a state based stream cipher. Step 150 encrypts the original data using the cipher stream generated in step 140. Step 160 creates the packet for transmission. In step 170, the internal sequence counter value is stored in the packet as the counter value. The packet is then transmitted to the receiving node in step 180. An acknowledgement message from the receiving node is received in step 190, the message denoting that the packet has been properly received and decoded. With the receipt of the acknowledgement message, the transmitting node can ensure that the next cipher stream will be properly generated. In step 200, the current state is saved as a previous state. Step 210 generates the next state and stores it as the current state or the transmitting current state. Step 220 increments the internal sequence counter and the transmitting node is now ready to receive more original data. It should be noted that the internal sequence counter that is incremented in step 220 is the internal sequence counter for which the node is the transmitting node. As noted above, each node has two internal sequence counters – one for which the node is the transmitting node and one for which the node is the receiving node. These counters work and are incremented independently of one another.

[00036] For the receiving node, the steps it undertakes to receive and process the packet are outlined in the flowchart of Fig 7. The process begins with step 230, that of receiving the packet from the transmitting node. In step 240, the sequence counter value is extracted from the packet. Step 250 compares the extracted sequence

counter value with an expected value. In one embodiment, the extracted sequence counter value is compared with the internal sequence counter value at the receiving node. In the same embodiment, the extracted counter value is expected to be same as the internal counter value if the internal counter were to be used as a packet counter. In other embodiments which use the counter value as a byte or a byte group counter, the extracted counter value is expected to be greater than the internal counter. If the extracted counter value does not have the expected value, then step 260 is that of recognizing if state synchronization has been lost and whether a resynchronization has to be requested or if the packet merely has to be dropped. If the extracted counter value is as is expected, step 270 generates a cipher stream using the stored current state and the stored base key. The encoded data in the packet is decoded in step 280 using the cipher stream generated in step 270. With the successful decoding/decryption of the encoded data, an acknowledgement message is sent to the transmitting node in step 290. In step 300, the internal sequence counter is incremented. This internal sequence counter is the counter for which the node is the receiving node. In step 310, the current state is saved as the previous state while in step 320, the next state is generated. Step 330 stores the next state as the current state.

[00037] Referring to Figs 8 and 9, block diagrams of the data flow for an RC4 implementation of the above invention are illustrated. Fig 8 illustrates the data flow for the encryption process while Fig 9 illustrates the data flow for the decryption process.

[00038] In fig 8, the plaintext or original data 340 is received and it passes through an integrity algorithm module 350 that computes the integrity check value (ICV) 360. ICV 360 is combined with the plaintext 340 and both are streamed into the XOR module 370 to be XOR'd with the cipher stream 380. The cipher stream 380 is the output of the cipher module 60A in which a pseudorandom generator module 390, using a current encryption state 400 (transmission current state) and a base key (not shown), produces the cipher stream 380. The design and construction of module 390 is well-known in the art of encryption. The module 390 also calculates the next state 410 which is, after the cipher stream 380 is produced, stored as the current state in storage 420. Once the XOR operation is done, the encrypted text or encoded data 425 results. This encoded data 425 is combined with the internal sequence counter value 430 as part of the payload for packet 440. The internal

sequence counter value 430 is produced and incremented when necessary by the transmitting internal sequence counter module 450.

[00039] In Fig 9, the packet 440 has been received by the receiving node. The sequence counter value 460 extracted from the packet 440 is sent to the cipher module 60B while the encrypted text of encoded data 470 is sent to the XOR module 480. In the cipher module 60B, the cipher stream 490 is produced by a pseudorandom generator module 500 (similar to the module 3 in Fig 8) by using the encryption state 510 (receiving current state) and the base key (not shown). The encryption state 510 results from selector module 520 which contains the receiving internal sequence counter module 530. This module 530 is similar to the module 450 in Fig 8. The counter value 460 is compared with the result of module 530 and, if the counter value 460 is same as the result of module 530 or if the counter value 460 is in line with an expected value or range of values, then the encryption state 510 is sent to the module 500 to produce the cipher stream 490. As with module 390 in Fig 8, module 500 in fig 9 also produces the next state 540 which is stored as the current state in storage 550 when necessary (i.e. when the packet has been validated). Once the cipher stream 490 has been produced, this is XOR'd with the encoded data 470 by the XOR module 480 to result in the original data or plaintext 560 and in the integrity check value 570. The plaintext is, again, passed through an integrity algorithm module 580. The result of the module 580 is compared with ICV 570 for validation by way of validation module 590.

[00040] For clarity, it should be noted that, for each uplink/downlink pair of links, there are 5 pairs of states that are used by the links. These are :

Initial states (or initializations states) : IRC_U, IRC_D

Previous states : PRC_U, PRC_D

Current states : CRC_U, CRC_D

Next States : NRC_U, NRC_D

Synchronization States : SRC_U, SRC_D

It should also be noted that the subscript letter denotes which link the states relate to. As an example, PRC_U refers to the previous state for the uplink link while PRC_D refers to the previous state for the downlink link. Furthermore, it should be noted that:

- IRC are (collectively) the RC4 states after performing RC4-KSA and RC4-PRGA for I-Offset number of bytes for every base key in an RC4 implementation. A node may start encrypting data packets with a new base key only after calculating the RC4 states IRC_U and IRC_D corresponding to the uplink base key and the downlink base key respectively.
- PRC are the RC4 states corresponding to previously successfully transmitted or received and acknowledged encrypted packet. PRC_U and PRC_D are updated independently.
- CRC are the RC4 states with which encryption and decryption of the subsequent packet takes place for a given base key. CRC_U and CRC_D are updated independently.
- NRC are the RC4 states corresponding to I-Offset for the next base key pair. NRC_U and NRC_D are updated independently. These states can be kept ready by a receiving node when it is anticipating a key change and these are simply IRC states corresponding to the next anticipated base key and should not be confused with CRC states.
- SRC are the RC4 states corresponding to S-offset (an offset different from I-Offset and is also shared) for a given base key pair and are used in the resynchronization protocol. These can be calculated the same way IRC states are calculated.

CRC states are continuously maintained for a pair of nodes: IRC, PRC, NRC and SRC states are logical states and may be generated as needed or maintained continuously, an option of interest to resource-limited devices.

[00041] Offsets are also used in the different procedures for authentication (I-Offset) and resynchronization (S-Offset). Offsets are used to indicate how far down the cipher stream a node should start encrypting and decrypting messages for a given base key pair. This is referred to as running down the cipher stream. Running down the cipher stream for a specified number of bytes (equal to I-Offset) will happen only whenever a key rollover or replacement takes place. The purpose of I-Offset is to discard the predetermined number of bytes from the start of a stream (carried out only once for any base key), thereby strengthening the stream cipher. Thus, if the I-Offset is 10 data groups (e.g. 10 bytes) then encryption/decryption of the data stream only begins 10 data groups into the data stream. There are two types of Offsets: Initial Offset (I-$Offset_{U/D}$) and Sync Offset (S-$Offset_{U/D}$). Sync Offset is used during resynchronization mechanism to encrypt and decrypt resynchronization frames. Initial Offset is used for

encryption and decryption of all the other frames exchanged between the nodes. Both Initial Offset and Sync Offset are non-zero values, which are distinct from each other. The Sync offset is generally less than the Initial Offset.

[00042] To initiate the system, the transmitting node (or initiating node) initiates by following the authentication procedure. Referring to Fig 10, a dataflow/message flow diagram illustrates the direction in which administrative messages are exchanged between the node A (transmitting node) 600 and node B (receiving node) 610 for the authentication process.

[00043] The node A 610 first finds a base key for an uplink link (the link for transmissions from node A to node B) and a base key for a downlink link (the link for transmissions from node B to node A). These two base keys can form a base key pair that each node either generates or selects from a list. Node A also generates the initial state for both the uplink and downlink links. An offset (I-offset), or a predetermined point in the data stream where encryption or decryption is to begin, is also selected by Node A. The node A then encrypts a nonce or marker (a number or alphanumeric string generated by the transmitting node) with its integrity check value (ICV) using the uplink base key and the uplink initial state. A first authentication packet 620 is sent to node B 610. This packet 620 contains the encrypted nonce, its encrypted ICV, an initial sequence value (e.g. 1), and an administrative identification. Thus the first authentication message AUTH1 may be Auth(INIT, 1,(nonce,ICV)@ IRC_U) where INIT is the administrative identifier (identifying the message as part of an authentication/initialization sequence), 1 is the initial sequence value (denoting the message as the first in the sequence), and (nonce, ICV) @ IRC_U denotes the encrypted nonce and its ICV, both being encrypted using the initial state for the uplink link and the uplink base key.

[00044] When node B 610 receives the first authentication message 620, it selects/calculates the same base keys (or base key pair), the same initial states for the two links, and decodes the nonce and its ICV from the authentication message 620. If the nonce and its ICV match (i.e. the nonce is validated), then node B 610 generates a second authentication message 630 (AUTH2) and sends this to node A 600. If the nonce is not validated, then node B 610 ignores the message AUTH1. The second authentication message 630 contains two encrypted nonces – the first nonce or marker from node A 600 and a second nonce or marker generated by node B 610. These nonces (and a corresponding ICV for both nonces) are encrypted by node B 610 using

the initial state for the downlink link and the base key for the downlink link. The second authentication message AUTH2 may therefore be Auth(INIT,2,(nonce, new_nonce, ICV)@ IRC_D). The sequence value is 2 to denote the second message in the initialization sequence while (nonce, new_nonce, ICV)@ IRC_D denotes that the first marker/nonce from node A 600, the new nonce/second marker generated by node B 610, and the associated ICV are encrypted using the initial state for the downlink link and the base key for that link.

[00045] When node A 600 receives AUTH2, the two nonces are decrypted and validated. The final authentication message 640 (AUTH3) is then sent to node B 610. The node A 600 extracts and reencrypts the new nonce and its ICV for inclusion in AUTH3. This payload is encrypted using a state derived from the uplink initial state. AUTH3 thus has the format Auth(INIT,3, (new_nonce, ICV) @ IRC_U+X). Since this message is the third in the initialization sequence, then 3 is the sequence value. The encryption state is X states away from the uplink initialization state. X is the number of bytes previously encrypted for AUTH1. Once this message is sent, the internal sequence counters, for both uplink and downlink links in both nodes, are initialized to zero. At this point, both nodes can now communicate with one another as their states are synchronized with a shared nonce/marker being the new nonce generated by node B 610. The shared states are, for the uplink link, the current state is the state arrived at from the initial state for that link (IRC_U) after encrypting the data from AUTH1 and AUTH3. For the downlink link, the shared current state is the state arrived at from the initial state for that link (IRC_D) after encrypting the data in AUTH2. The base keys for the uplink and the downlink links are, as noted above, shared by both nodes. It should also be noted that the authentication messages are management messages used for administrative purposes by the nodes. The administrative identifier field in each management message identifies the type of message being sent.

[00046] In the event state synchronization is lost between two communicating nodes, resynchronization can be accomplished using a four-part resynchronization sequence. Fig 11 illustrates the message flow between two nodes seeking resynchronization. For clarity, the link from node A (transmitting node) to node B (receiving node) will be termed the uplink link while the reverse will be termed the downlink link.

[00047] A need for resynchronization is usually detected when the receiving node receives a packet it cannot decode. When the receiving node (node B) detects

this condition, it declares the receiving current state as invalid and sends a resynchronization request message 650 (RESYNC1) to the transmitting node (node A) at S-Offset using the state SRC_D. The resynchronization messages are one type of management message similar to the authentication message but with a different function. At this point, the two nodes have a shared nonce negotiated between the two when authentication was first initialized. This nonce or marker (an authentication parameter) thus becomes the payload of RESYNC1 and is encrypted using a specific predetermined state (SRC_D) with the base key for that downlink link. RESYNC1 has the format Auth(SYNC,1,(nonce,ICV))@ SRC_D. The SYNC field signals to the transmitting node (node A) receiving RESYNC1 that the payload must be decrypted using the base key for that link and the predetermined state SRC_D. The 1 denotes that the message is the first in a resynchronization sequence. It should be noted that the resynchronization process has associated with it a specific predetermined offset (S-Offset). All encryption and decryption thus begins at the specific offset.

[00048] Once the transmitting node (node A) validates RESYNC1 by way of the payload nonce and its ICV, it selects a new current state derived from its own current state for the uplink link. This is done by running down the current state either a set number of states or a random number of states or by simply using the present current state and redesignating this state as the new current state. Either way, the new current state can be arrived at by running down the current state. Using the new state, the transmitting node creates a new resynchronization message RESYNC2 660 with a payload of the encrypted nonce and its ICV, both being encrypted using the new current state. Also, in RESYNC2 is the internal sequence counter value for the uplink link so that the receiving node may use the value once resynchronization is achieved. RESYNC2 may thus have the following format – Auth(SYNC,2, SSC_U, (nonce,ICV)@CRC_U) with SSC_U being the sequence counter value for the uplink link.

[00049] When the receiving node receives RESYNC2, it acknowledges the message even if it is, as yet, incapable of decoding the payload. The receiving node, since it knows the last current state, uses this as the starting point in searching for CRC_U, the proper current state. The receiving node runs down the states from the last current state and attempts to decode the payload of RESYNC2. When the decryption succeeds, then the state which succeeds must be the same state as CRC_U. This state is then saved as the proper current state for the uplink link in the receiving node and the internal sequence counter in the receiving node for that link is set to SSC_U -1. An

implementation dependent limit of how many states to try in decoding the RESYNC2 payload may be implemented. If the limit is reached, then the receiving node may try to restart the resynchronization process or it may deem itself deauthenticated. If CRC_U is found by the receiving node, this node creates new message RESYNC3 670 with, as its payload, not only the nonce but a newly generated nonce. Both nonces are encrypted using the current state for the downlink link as the synchronization was only lost for the uplink link. Thus, RESYNC3 will have the format Auth(SYNC,3, SSC_D,(nonce, new_nonce, ICV)@ CRC_D) where SSC_D is the sequence counter value for the downlink link and (nonce, new_nonce, ICV)@ CRC_D means the two nonces and their ICV are encrypted using the downlink link current state.

[00050] With node A's receipt of RESYNC3, it validates the new_nonce and encrypts this new nonce using its next current state for the uplink link since receipt of RESYNC3 means that node B has correctly found CRC_U. Node A then sends RESYNC4 380 to node B. RESYNC4 thus has the format Auth(SYNC,4, SSC_U,(new_nonce, ICV)@ CRC_U+1) where (new_nonce, ICV)@ CRC_U+1 denotes encrypting the new nonce and its ICV using the next current state after CRC_U. Both nodes now have a new nonce to use and their states are synchronized again.

[00051] If node A cannot decode RESYNC3, then synchronization has been lost in the downlink link as well. Node A can then run down the states from the last current state for the downlink link until the RESYNC3 payload can be decoded. Once this occurs, this means that node A has found CRC_D and this will be the current state for the downlink link.

[00052] To run down the states when searching for a state which will decode a specific payload, the decoding node first tries to decode one byte from the packet using a given state. It should be noted that this byte is the first byte of the nonce that the transmitter and the receiver shared during the authentication procedure and so the receiver knows what this first byte should be. If the decoding of that single byte is unsuccessful, then the next state is tried. This process continues until the first byte is successfully decoded. If the first byte decodes properly then an attempt at decoding the first two bytes is tried. Again, if there is no success, the next states are tried until the two bytes are decoded. Once decode, then the first three bytes are attempted. The process continues, continuously adding bytes to the portion of the packet being decoded until a state successfully decodes the payload. This state that fully decodes is the desired state. Of course, other processes for resynchronization may also be used.

It should be noted that the above mentioned resynchronization may not be even required if the internal counter maintained by the transmitter and the receiver actually counts the number of bytes exchanged rather than the number of packets.

[00053] The base keys used by the uplink and downlink are, as noted above, shared between the two nodes. One method to share the base keys without transmitting the keys is to maintain base key lists at each end of the links, one for each link. Once a key change is required, the node merely selects the next key in the base key list. Similarly, each node may generate the base keys using predetermined algorithms such that each node executing the algorithm will produce the same base key. These base keys can then be stored or generated as needed. Other methods for either distributing or sharing base keys between nodes, preferably without transmitting the keys, may be used.

[00054] The above scheme may use base key pairs of two full 64 bit or 128 bit keys which are used as RC4 seeds for the communication between two nodes. Other key sizes work, with the provision that the smallest size must be computationally hard to break (i.e. approximately 64 bits). These keys may be selected from a Base Key List, which may be common to all nodes within the same BSS or common only to a communicating node pair. The Base Key Pair may also be per session based and may be agreed upon between the two communicating nodes. It is preferable that a strong key generation and distribution process be used and note that the way that keys are selected or generated must preserve uniqueness of each base key across the BSS (basic service set)/IBSS (independent basic service set).

[00055] The base key list may not necessarily be a list. Techniques such as splitting the base key into a common (static) primary portion and generated secondary portion can also be used for an implicit Base Key List. The generated, secondary portion can be generated using parts of the timestamp of a beacon, and/or hardware addresses of the communicating nodes, with the proviso that each session of the same communicating pair and different pairs generate different secondary key portions. Alternatives to a Base Key List would involve communicating pairs generating session based unique key pairs, using protocols such as the Extensible Authentication Protocol.

[00056] The above system may also use expiring keys – base keys may expire after a set time period defined as a key duration. Key Duration indicates how often a base key has to be changed or when a base key changes. The invention may be adjusted to rely a timestamp from a beacon to give a common notion of time within that network

and hence can be used with Key Duration to have common knowledge of when to change a key between any pair of nodes. Change of base key may be just as easy as selecting the next key from the Base Key List. Note that nodes never exchange information to indicate when key change occurs or to identify the next key pair selected. This information was exchanged before authentication by methods known in the art.

[00057] A node B may send deauthenticate messages to node A if node B has authentication for node A and needs to terminate direct communication with node A. These messages could be used to release resources associated with the communication between nodes B and A. Node B terminates communication with A by sending a deauthentication message with the format Deauth(Reason,1, SSC_U,(nonce,ICV)@ $CRC4_U$)). Node A decrypts the packet and responds by sending a second deauthentication message with the format Deauth(Reason,2, SSC_D,(nonce,ICV)@ $CRC4_D$)). It should be note that neither node B nor node A releases resources until the deauthenticate messages are exchanged. Deauthentication can also occur implicitly when node B has been silent for an implementation dependent amount of time (possibly infinite) and node B did not notify node A that it would be asleep, or when a resynchronization is attempted and fails after an implementation dependent number of attempts.

[00058] The invention may also be adjusted to provide support for broadcasting and multicasting. The lack of acknowledgment in broadcast and multicast messages makes maintenance of synchronization harder, but not impossible. Nodes which cease communication for a while require resynchronization, and running forward on the encryption stream can be costly. Once a node has acquired an encryption stream, however, there is no difficulty in maintaining the state. While other schemes or approaches may be used, one approach which can be used is as follows: Each node which is sending packets via the broadcast approach has a shared private base key which remains in force for some extended period of time (hours or days or indefinite). This primary key is augmented by the concatenation of a secondary key fragment derived from the beacon. The derived key fragment changes frequently on a BSS or IBSS defined period, usually in the order of seconds. When secondary key hop occurs, the transmitter node selects a new secondary key based upon the nkey change period as defined by the beacon, runs down the encryption stream for I-offset bytes and begins transmission from there. Receiving nodes also note the change in key from the beacon and use the key-hop process for receivers described above to receive packets and to

detect key change. While (the primary,secondary) key pair are in use, transmitter nodes and receiver nodes use the state-based encryption to send and receive messages.

[00059] Using the above, nodes which have been in low power or that have missed some packets will not be able to synchronize with the encryption stream until the next secondary key change. Since this is likely only a few seconds, such nodes can use preknowledge of state change to get ready for decryption and begin trying to decrypt packets transmitted after the key change. The first few may fail if they were generated before the key change, but the node entering the process will quickly find the start of the decryption stream and be able to successfully decrypt packets. Similarly, nodes authenticating and associating will wait until the next secondary key change to begin decrypting messages from each node with which they authenticate.

[00060] The invention can also work with IBSS. The 802.11 standard also supports ad hoc networking in which nodes are allowed to send packets to each other directly. This type of network is called an IBSS network. In this mode, the beacon generation is distributed and so all participating nodes are allowed to send beacons, one at a time. Although beacon generation is distributed, all nodes within an IBSS share a common notion of time.

[00061] The other major implication that IBSS has is that each node communicates with a number of neighboring nodes, and must maintain RC4 states for each node with which it forms a communicating pair. This means that selection of keys from a Base Key List and Key Duration also must be negotiated and maintained for each communicating pair as well.

[00062] Some IBSS networks have a dramatic simplification which can ease this process. If the data being exchanged does not have readily identifiable portions, such as TCP/IP headers or fields of fixed patterns, then an IBSS could use a single Base Key Pair, the same Base Key List (if any), the same I-Offset, and the same Key Duration. We note that the RC4 state will differ between different communicating pairs very soon after key hop because of the different sizes of packets and communication rates, so only insiders which know the Base Key Pair, Key Duration, and number of octets exchanged between communicating nodes of interest can decrypt packets. We also note that this simplification does not eliminate the need to maintain states PRC4 and CRC4 for all nodes which are communicating nodes with us.

[00063] Other schemes for key generation are needed for an IBSS with patterned data, since reused cipher streams permit data recovery when some of the data is known. Alternative schemes include:

- indexing into the Base Key List using the communicating pair's MAC addresses and some parameter based upon beacon timestamp to select a unique pair of keys,
- maintaining a single shared key portion of appropriate length, say 96 bits, and generate a unique portion (of 32 bits) based upon a changing criteria, such as a relevant portion of the beacon timestamp or a number which increments with the beacon at some frequency small enough to guarantee that each association created by communicating pairs generates a unique key,
- negotiating base keys or session-oriented base keys and other SBKH parameters, using 802.1x, EAP, or other techniques.

[00064] Once the first BaseKey has been allocated, key hop may occur by the normal process of selecting the "next" key from the Base Key List (if a key list exists), or by repeating the key generating process with new key fragments.

[00065] Embodiments of the invention may be implemented in any conventional computer programming language. For example, preferred embodiments may be implemented in a procedural programming language (e.g. "C") or an object oriented language (e.g. "C++"). Alternative embodiments of the invention may be implemented as pre-programmed hardware elements, other related components, or as a combination of hardware and software components.

[00066] Embodiments can be implemented as a computer program product for use with a computer system. Such implementation may include a series of computer instructions fixed either on a tangible medium, such as a computer readable medium (*e.g.*, a diskette, CD-ROM, ROM, or fixed disk) or transmittable to a computer system, via a modem or other interface device, such as a communications adapter connected to a network over a medium. The medium may be either a tangible medium (*e.g.*, optical or electrical communications lines) or a medium implemented with wireless techniques (*e.g.*, microwave, infrared or other transmission techniques). The series of computer instructions embodies all or part of the functionality previously described herein. Those skilled in the art should appreciate that such computer instructions can be written in a number of programming languages for use with many computer architectures or operating systems. Furthermore, such

instructions may be stored in any memory device, such as semiconductor, magnetic, optical or other memory devices, and may be transmitted using any communications technology, such as optical, infrared, microwave, or other transmission technologies. It is expected that such a computer program product may be distributed as a removable medium with accompanying printed or electronic documentation (*e.g.*, shrink wrapped software), preloaded with a computer system (*e.g.*, on system ROM or fixed disk), or distributed from a server over the network (*e.g.*, the Internet or World Wide Web). Of course, some embodiments of the invention may be implemented as a combination of both software (*e.g.*, a computer program product) and hardware. Still other embodiments of the invention may be implemented as entirely hardware, or entirely software (*e.g.*, a computer program product).

[00067] Although various exemplary embodiments of the invention have been disclosed, it should be apparent to those skilled in the art that various changes and modifications can be made which will achieve some of the advantages of the invention without departing from the true scope of the invention.

[00068] A person understanding this invention may now conceive of alternative structures and embodiments or variations of the above all of which are intended to fall within the scope of the invention as defined in the claims that follow.

We Claim:

1. A method of communicating between a first node and a second node in a communications network, the method comprising:

- at said first node, encrypting data for transmission to said second node using a state based stream cipher with a transmitting current state and a key shared by both first and second nodes,

- transmitting encrypted data from said first node to said second node, said encrypted data being transmitted with a sequence counter value from said first node

- at said second node, receiving said encrypted data and said sequence counter value

- ensuring a receiving current state in said second node is the same as said transmitting current state using said sequence counter value

- in the event said receiving current state is the same as said transmitting current state, decrypting said encrypted data using said state based stream cipher with said receiving current state and said key.

2. A method according to claim 1 wherein said step of ensuring said receiving current state in said second node is the same as said transmitting current state is accomplished by executing the following steps:

- comparing at said second node said sequence counter value with an internal previous sequence counter value stored at said second node
- in the even said sequence counter value is one more than said internal previous sequence counter value, determining that said receiving current state is the same as said transmitting current state
- in the event said encrypted data is successfully decrypted, sending an acknowledgement message from said second node to said first node and incrementing said internal previous sequence counter value.

3. A method according to claim 2 wherein said first node increments a counter value in said first node when said first node receives said acknowledgement message.

4. A method according to claim 1 wherein said key is changed after a predetermined time period.

5. A method according to claim 1 wherein said key is used for transmissions from said first node to said second node with a different key being used for transmissions from said second node to said first node.

6. A method according to claim 4 wherein said first node and said second node authenticate one another by exchanging authentication messages, at least one of said authentication messages causing at least one of said first or second nodes to determine a new key, an initial uplink state, or an initial downlink state, said authentication messages containing at least one marker value encrypted by said state based stream cipher, said transmitting current state and said receiving current state being both derived from a single state selected from the group comprising said initial uplink state and said initial downlink state.

7. A method according to claim 1 wherein both first and second nodes start encrypting and decrypting data at a predetermined offset from a start of a cipher stream.

8. In a communications network having a first node and a second node, each of which is using a state based stream cipher to encrypt and decrypt data transmitted between said first node and said second node, a method of processing transmitted data received from said first node by said second node, the method comprising:

- receiving at said second node transmitted data from said first node, said transmitted data including encrypted data encrypted by said first node using said state based stream cipher using a first transmitting current state and a common key, said common key being common to said first node and said second node, a copy of said common key being stored at said first node and said second node, said transmitted data including a sequence counter value, said sequence counter value being equal to a first internal sequence counter value stored at said first node;

- comparing at said second node said sequence counter value with an internal previous sequence counter value stored in said second node;

- in the event said sequence counter value is one more than said internal previous sequence counter value, decrypting said encrypted data at said second node using a receiving current state stored at said second node and said copy of said common key stored in said second node;

- in the event said encrypted data is successfully decrypted by said second node, sending an acknowledgement message from said second node to said first node;

- incrementing said internal previous sequence counter at said second node;

- storing said receiving current state as a previous state at said second node;

- determining a receiving next state at said second node; and

- storing said receiving next state as said receiving current state at said second node.

9. A method according to claim 8 wherein said first node processes data for transmission to said second node by executing the following steps:

- encrypting data for transmission to said second node using said state based stream cipher with said first transmitting current state and said common key to result in said encrypted data;

- transmitting said encrypted data with said sequence counter value to said second node;

- receiving said acknowledgement message from said second node in the event said encrypted data was successfully decrypted by said receiving node;

- storing said transmitting current state as a transmitting previous state at said first node;

- determining a transmitting next state at said first node;

- storing said transmitting next state as the transmitting current state;

- incrementing said first internal sequence counter value.

10. A method according to claim 8 wherein said common key is changed after a predetermined time period.

11. A method according to claim 8 wherein said common key is used for transmissions from said first node to said second node with a different key being used

for data transmissions from said second node to said first node, said different key also being common to said first and second node.

12. A method according to claim 11 wherein said first node authenticates said second node by executing the following steps:

- determining said common key and a corresponding different key, and an initial uplink state and an initial downlink state, said initial uplink state being used for transmissions from said first node to said second node and said initial downlink state being used for transmissions from said second node to said first node;

- creating a first authentication message for transmission to said second node, said first authentication message containing a first sequence number value and a first marker value encrypted using said state based stream cipher with said common key and said initial uplink state;

- transmitting said first authentication message to said second node;

- receiving a second authentication message from said second node, said second authentication message containing a second sequence number value, said first marker value, and a second marker value, said first and second marker values being encrypted using said state based stream cipher with said different key corresponding to said common key and said initial downlink state;

- decrypting said first and second marker values from said authentication message using said state based stream cipher with said different key and said initial downlink state;

- encrypting said second marker value using said state based stream cipher with said common key and a predetermined state derived from said initial uplink state to result in an encrypted second marker value;

- transmitting a third authentication message to said second node, said third authentication message containing said encrypted second marker value and a third sequence number value,

wherein said first sequence number value is less than said second sequence number value and said second sequence number value is less than said third sequence number value.

13. A method according to claim 12 wherein said second node responds to said authentication by said first node by executing the following steps:

- receiving said first authentication message from said first node;

- determining said common key, said corresponding different key, and said initial uplink state and said initial downlink state;

- decrypting said first marker value from said first authentication message using said state based stream cipher with said common key and said initial uplink state;

- generating said second marker value;

- encrypting said first marker value and said second marker value using said state based stream cipher with said different key and said initial downlink state;

-creating said second authentication message;

- transmitting said second authentication message to said first node;

- receiving said third authentication message from said first node.

14. A method according to claim 9 wherein said steps of encrypting and decrypting data begins at a predetermined offset of x data units in a data stream, x being a natural number greater than 0.

15. A method according to claim 12 wherein said first transmitting state and said receiving current state are both derived from said initial uplink state.

16. A method according to claim 13 wherein said first internal sequence counter value is initialized after said second node is authenticated by said first node.

17. A method of communicating between a pair of nodes in a communications network, the method comprising:

- using a state based stream cipher to encrypt and decrypt data between said nodes;

- maintaining one set of uplink states for an uplink connection between said pair of nodes and one set of downlink states for a downlink connection between said pair of nodes, said states being used for encryption and decryption of data with said state based stream cipher;

- maintaining at least one common marker value between said pair of nodes, said marker value being used for authentication and synchronization of states between said pair of nodes;

- maintaining separate keys for said uplink and downlink connections between said communicating nodes, said separate keys being used for said encryption and decryption of data with said state based stream cipher ;

- maintaining a separate counter value for each one of said downlink and uplink connections between said pair of nodes, said sequence counter value being used to ensure that a first state used to encrypt data at a first node is the same state as a second state used to decrypt data at a second node.

18. A method according to claim 17 wherein said state based stream cipher is RC4.

19. A method according to claim 17 wherein said separate keys are periodically changed.

20. A method according to claim 17 wherein each one of said separate keys are independently generated at each one of said pair of nodes by a predetermined method.

21. A method according to claim 17 wherein each one of said states is independently generated at each one of said pair of nodes.

22. A method according to claim 1 wherein subsequent data encrypted for transmission to said second node is encrypted using a single cipher stream.

23. A method according to claim 8 wherein subsequent encrypted data is encrypted by said first node using a single cipher stream.

24. A method according to claim 17 wherein said keys are used to initialize a module used for generating a cipher stream used for encryption and decryption of data.

25. A first node for use in a communications network, said first node communicating with a second node using a state based stream cipher, said first node generating a first single cipher stream for encrypting data for transmission to said second node, said first node also generating a second single cipher stream for

decrypting data received from said second node, said first and second nodes being state synchronized with each other to enable encryption and decryption of data.

26. A first node according to claim 25 wherein said first single cipher stream is initialized whenever a shared based key changes, said shared based key being changed on a periodic and non-packet based basis.

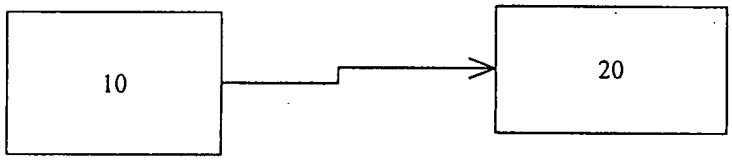


FIGURE 1

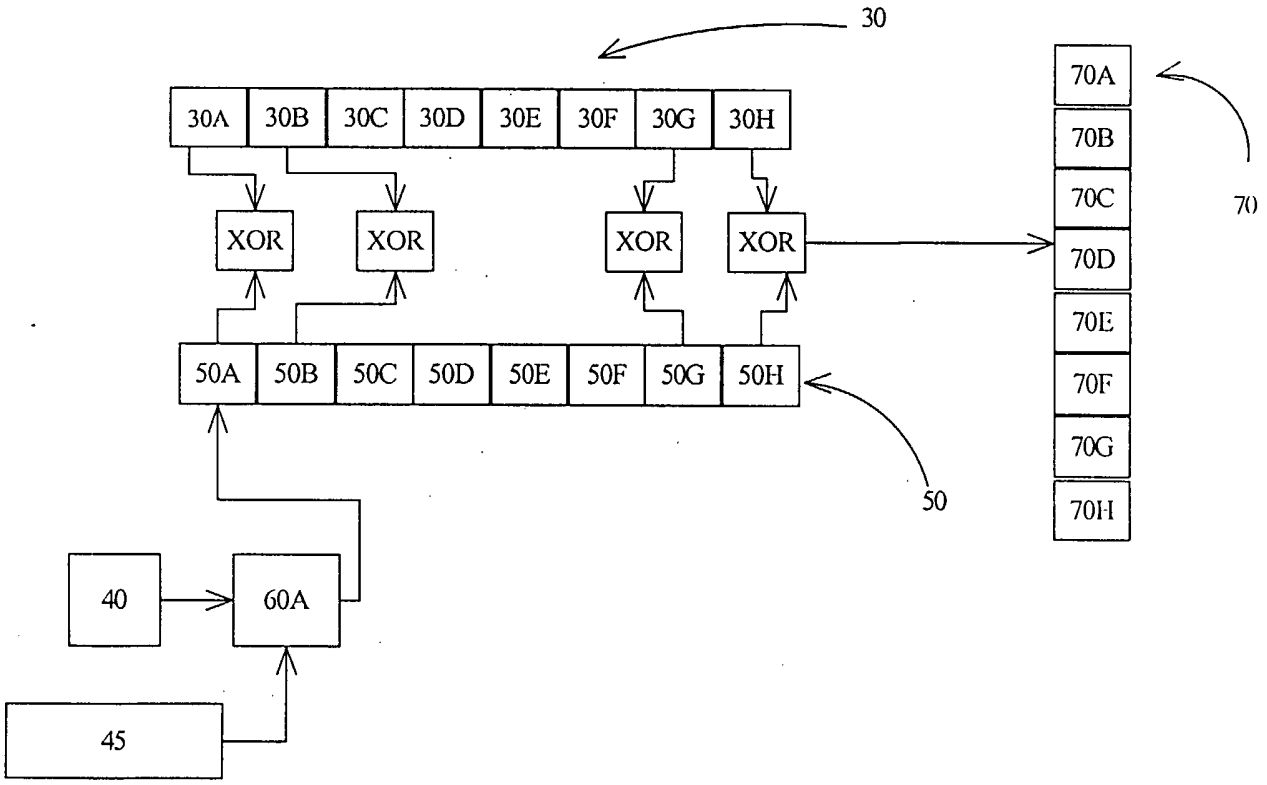


FIGURE 2

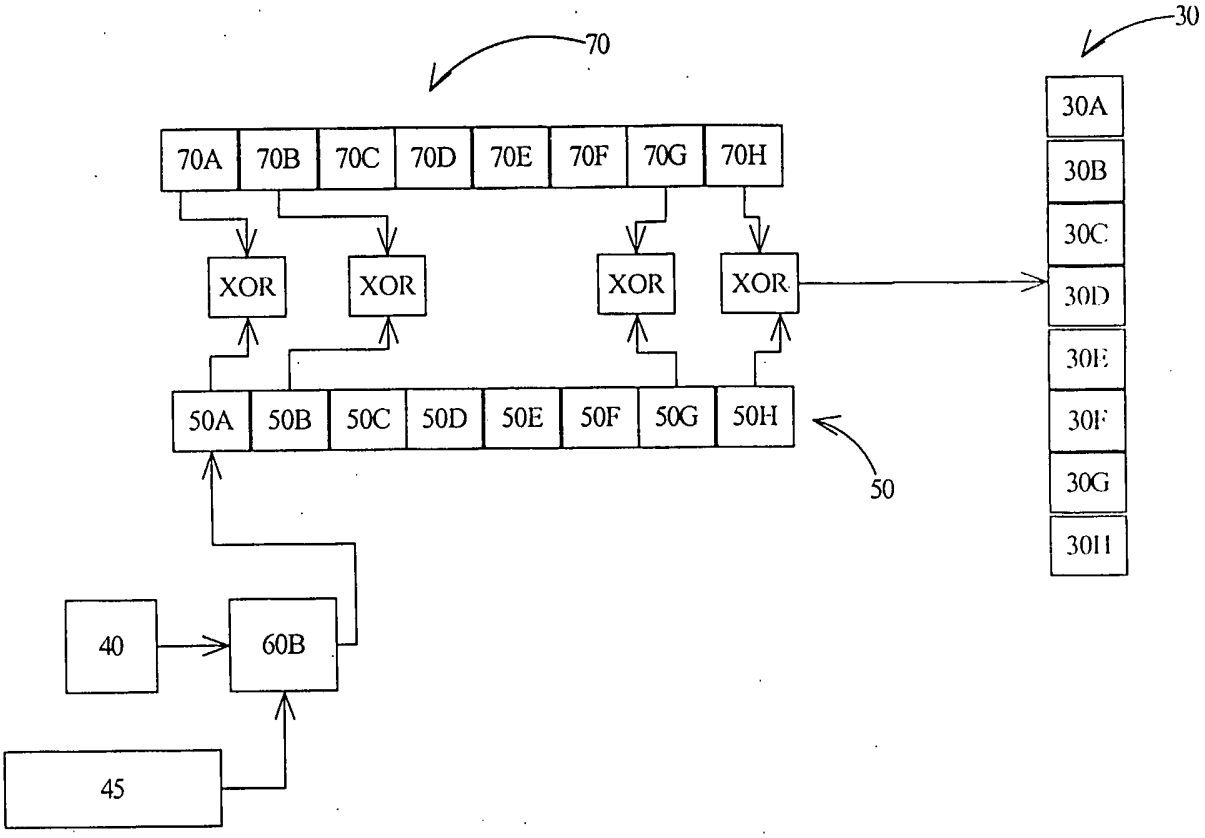


FIGURE 3

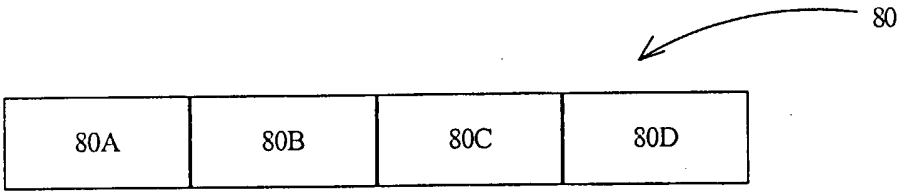


FIGURE 4

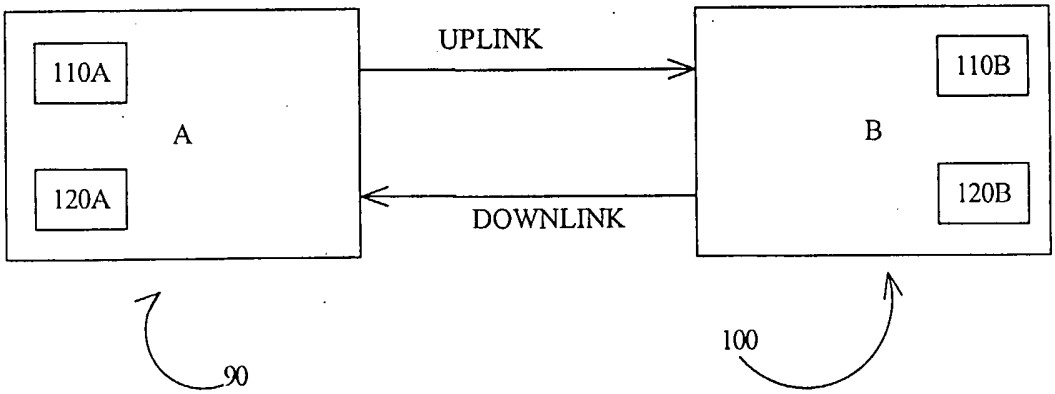


FIGURE 5

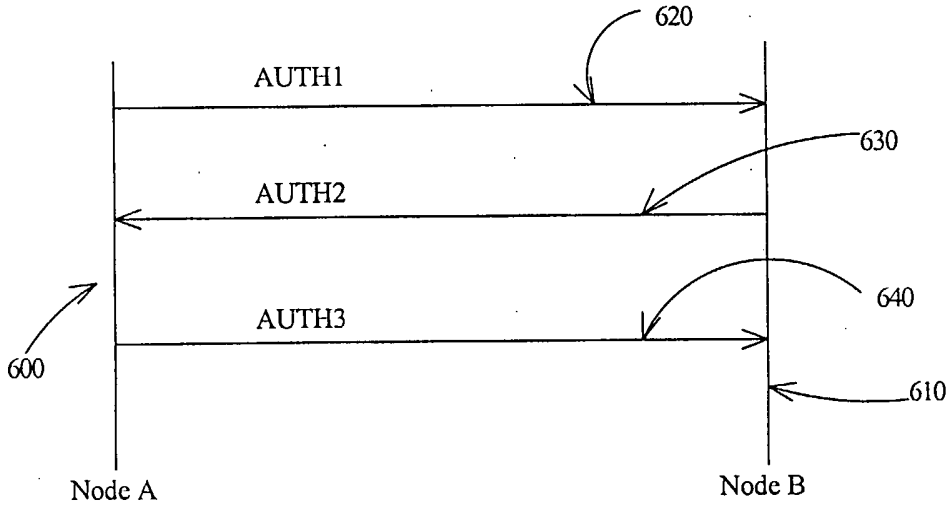


FIGURE 10

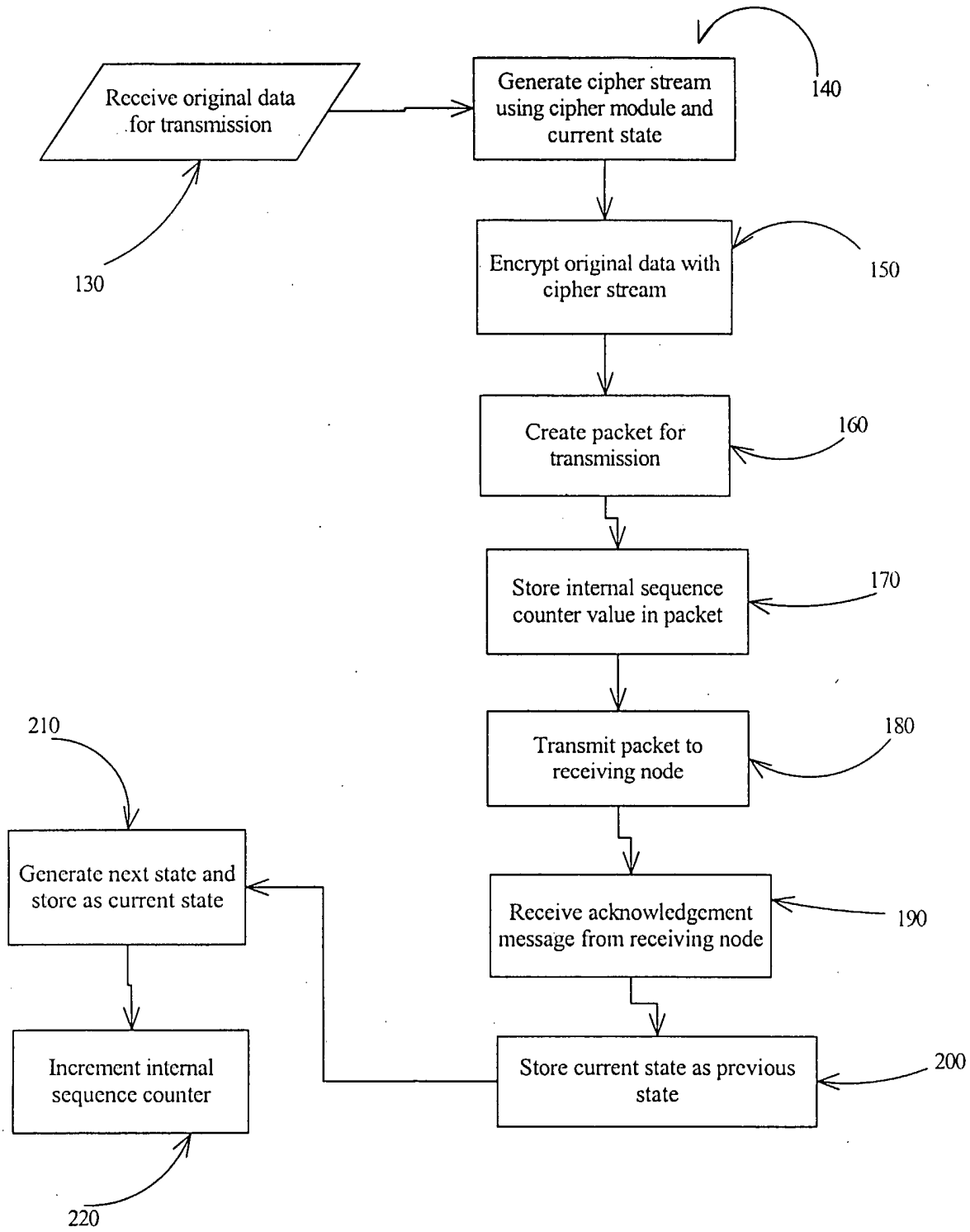

Receive original data for transmission
130
Generate cipher stream using cipher module and current state
140
Encrypt original data with cipher stream
150
Create packet for transmission
160
Store internal sequence counter value in packet
170
Transmit packet to receiving node
180
Receive acknowledgement message from receiving node
190
Store current state as previous state
200
Generate next state and store as current state
210
Increment internal sequence counter
220

FIGURE 6

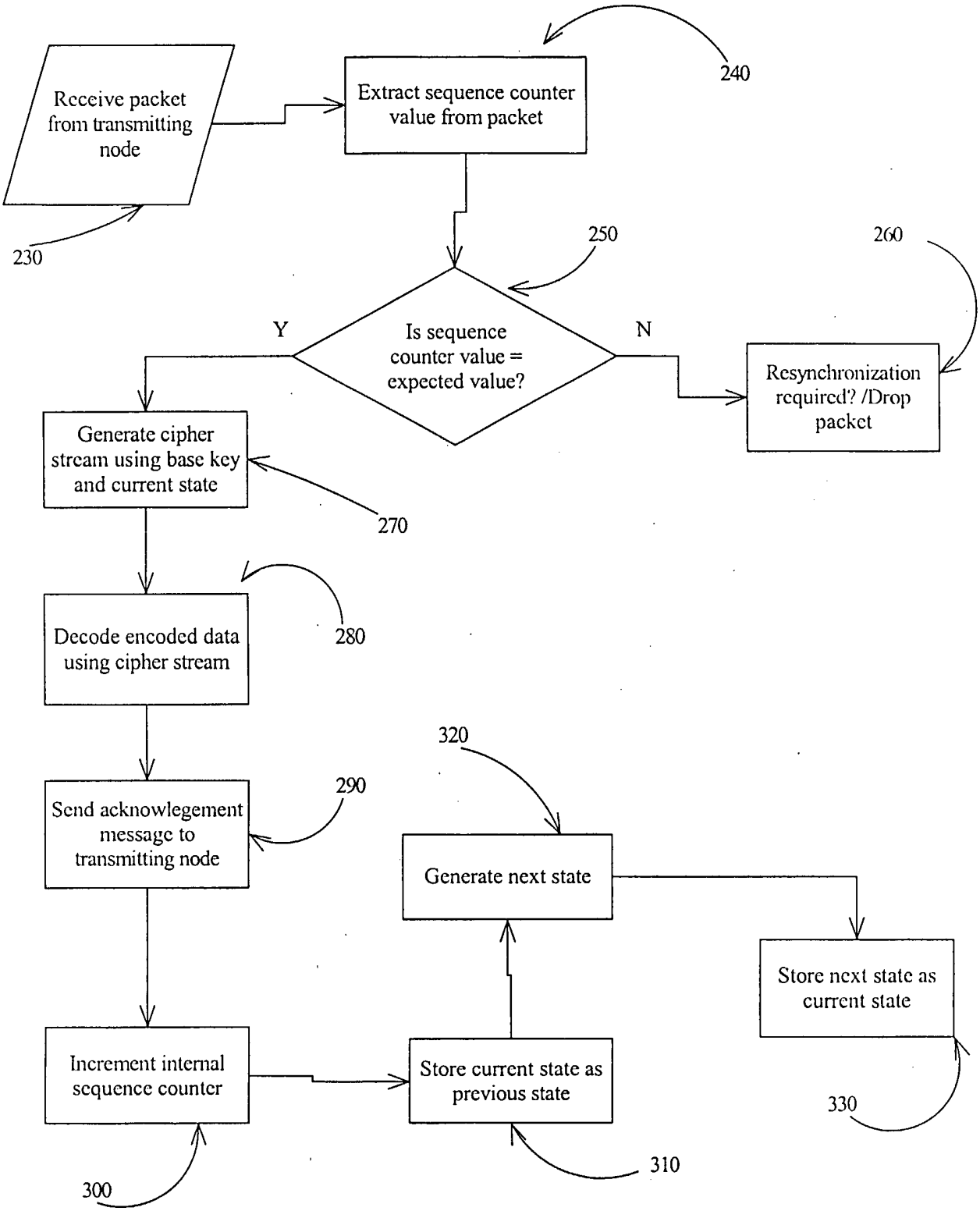

Receive packet from transmitting node
230
Extract sequence counter value from packet
240
Is sequence counter value = expected value?
250
Y
N
Resynchronization required? /Drop packet
260
Generate cipher stream using base key and current state
270
Decode encoded data using cipher stream
280
Send acknowlegement message to transmitting node
290
Increment internal sequence counter
300
Generate next state
320
Store current state as previous state
310
Store next state as current state
330

FIGURE 7

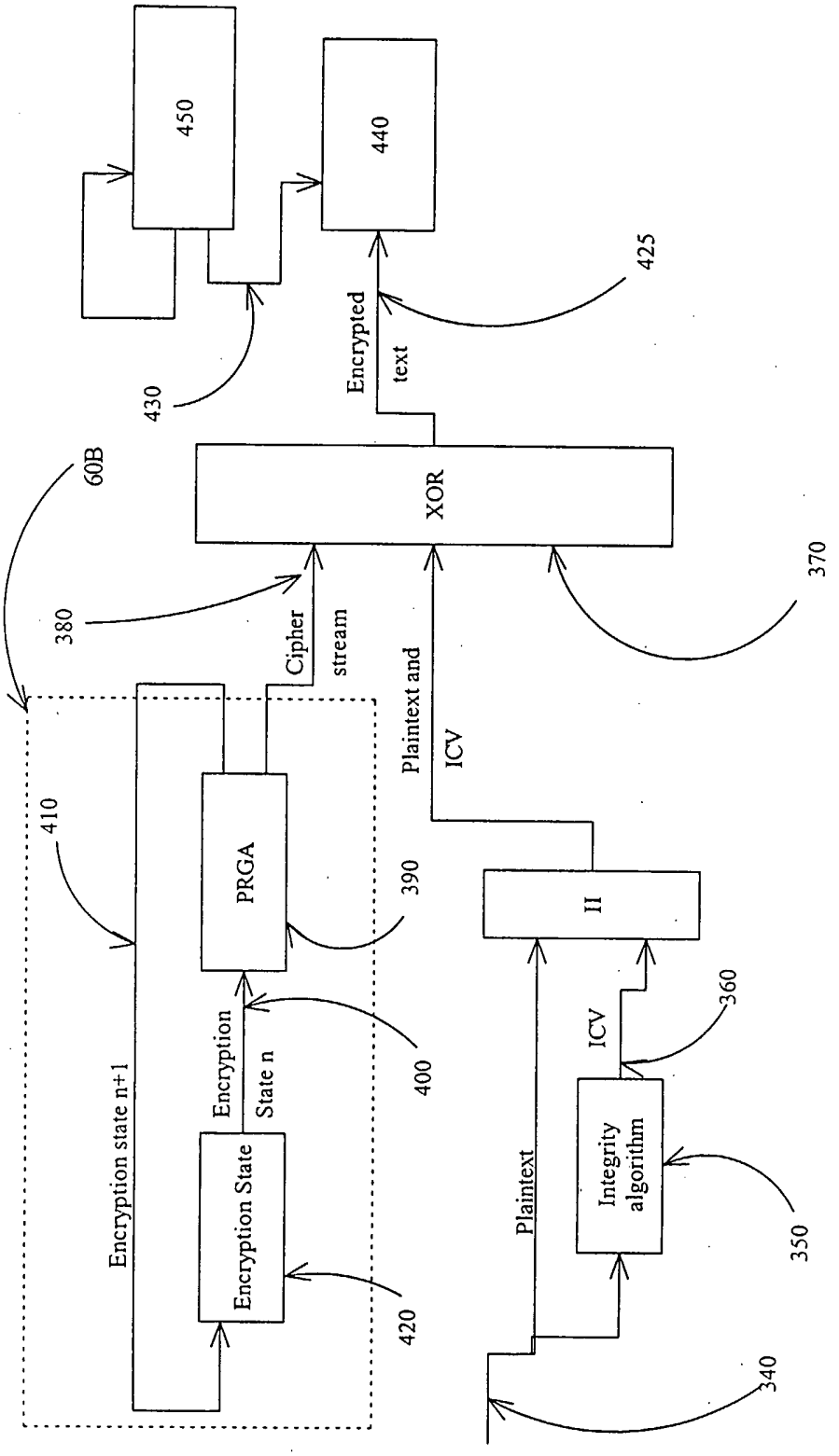

450
440
425
Encrypted text
430
60B
XOR
380
Cipher stream
370
Plaintext and ICV
410
PRGA
390
II
360
ICV
Encryption State n
400
Encryption state n+1
Encryption State
Integrity algorithm
Plaintext
420
350
340

FIGURE 8

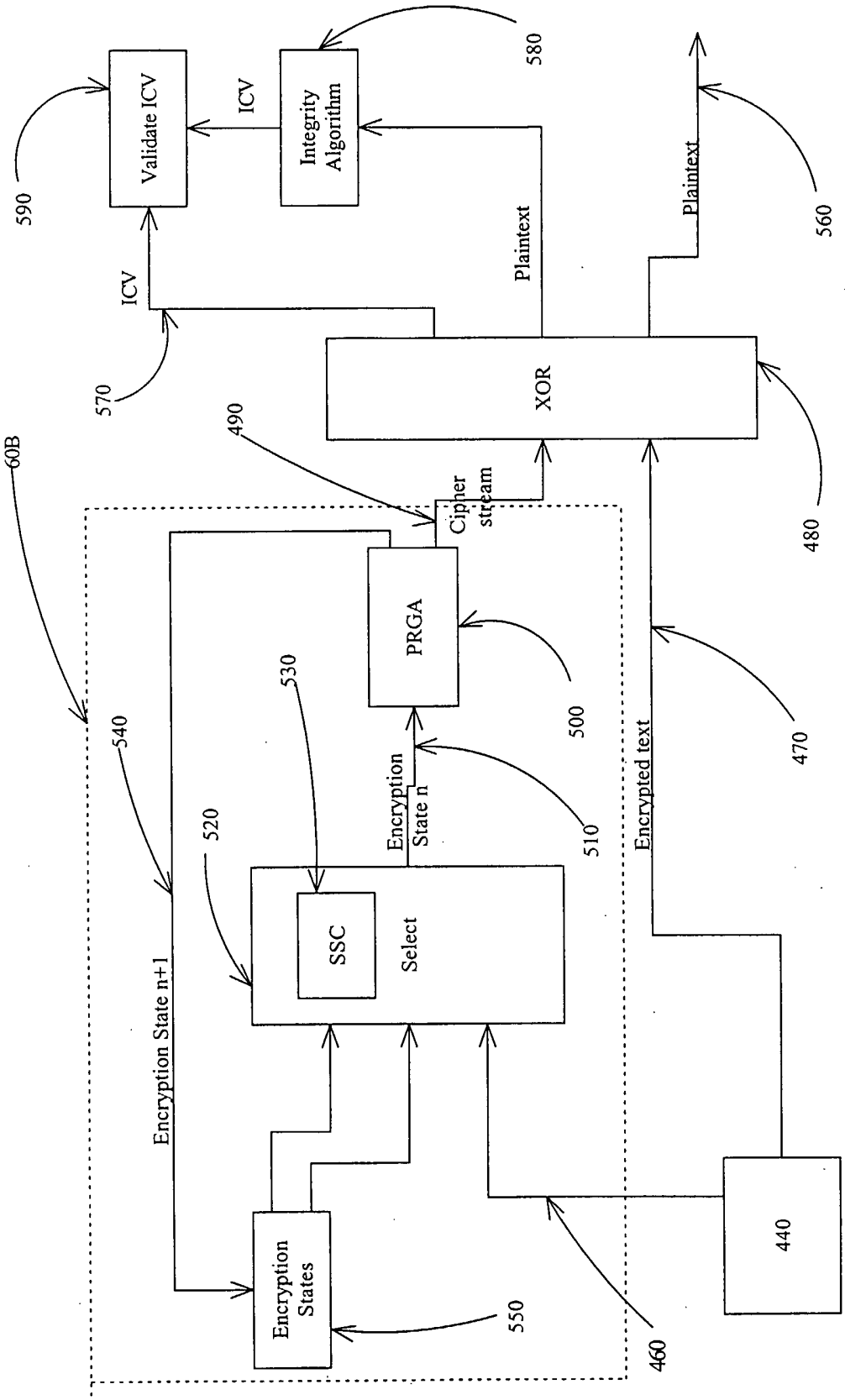

580
Validate ICV
ICV
Integrity Algorithm
590
ICV
Plaintext
Plaintext
560
570
XOR
490
480
60B
Cipher stream
PRGA
530
500
540
470
Encryption State n
520
510
Encrypted text
SSC
Select
Encryption State n+1
440
Encryption States
550
460

FIGURE 9

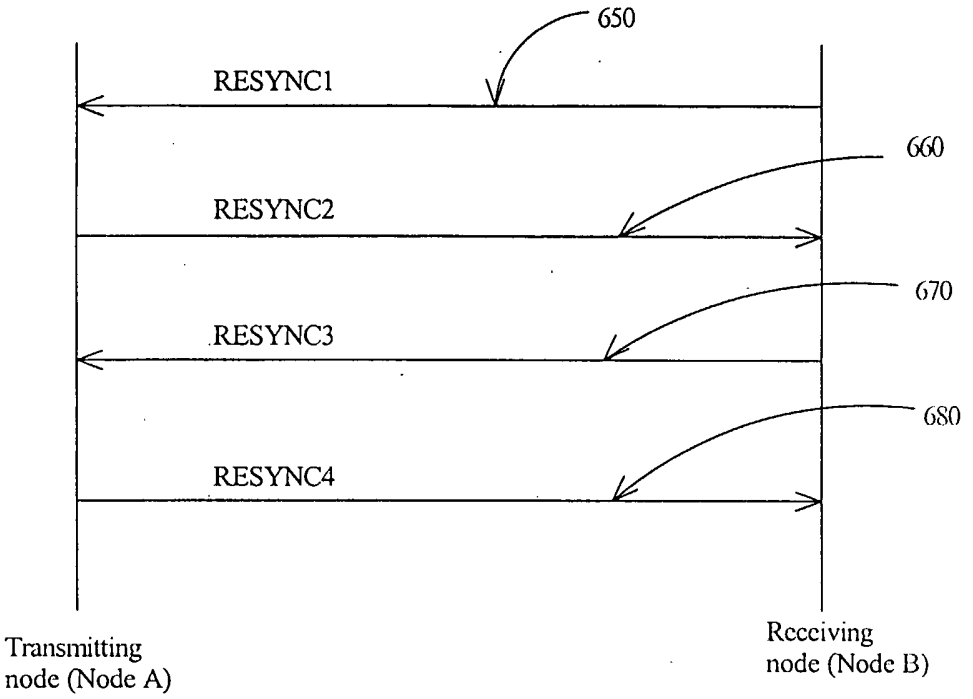

650
RESYNC1
660
RESYNC2
670
RESYNC3
680
RESYNC4
Transmitting node (Node A)
Receiving node (Node B)

FIGURE 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2005/000817

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7): H04L 9/18, H04L 9/32, H04L 9/20

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(7): H04L 9/18, H04L 9/32, H04L 9/20

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Delphion, EPAT, Canadian Patent Database, US Patent Database
Keywords: stream cipher, state based, synchronizing stream cipher

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CA2347011 (MCGROGAN et al.) 04-May-2000 -see whole document	1-26
A	US6909785 (ROSE, Gregory) 21-June-2005 -see whole document	1-26
A	US2002/0006197 (CARROLL et al.) 17-January-2002 -see whole document	1-26
A	US6226742 (JAKUBOWSKI et al.) 01-May-2001 -see whole document	1-26

[ ] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 August 2005 (19-08-2005)	07 September 2005 (07-09-2005)

Name and mailing address of the ISA/CA
Canadian Intellectual Property Office
Place du Portage I, C114 - 1st Floor, Box PCT
50 Victoria Street
Gatineau, Quebec K1A 0C9
Facsimile No.: 001(819)953-2476

Authorized officer
Hassan Bayaa (819) 997-7810

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/CA2005/000817

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
CA2347011	04-05-2000	AU1517600 A	15-05-2000
		AU1599800 A	15-05-2000
		CA2347011 A1	04-05-2000
		CA2347806 A1	04-05-2000
		EP1127421 A1	29-08-2001
		EP1127425 A1	29-08-2001
		US6266418 B1	24-07-2001
		US2001021252 A1	13-09-2001
		WO0025467 A1	04-05-2000
		WO0025476 A1	04-05-2000
US6909785	21-06-2005	AU2904201 A	06-06-2001
		BR0015479 A	08-10-2002
		CA2389397 A1	17-05-2001
		CN1390406 A	08-01-2003
		EP1228597 A2	07-08-2002
		IL149453D D0	10-11-2002
		JP2003514438T T	15-04-2003
		MXPA02004669 A	29-11-2002
		NO20022234 A	05-07-2002
		US6909785 B1	21-06-2005
		WO0135572 A2	17-05-2001
US2002006197	17-01-2002	AU6302801 A	20-11-2001
		US6879689 B2	12-04-2005
		WO0186860 A1	15-11-2001
US6226742	01-05-2001	AU4069299 A	08-11-1999
		EP1074114 A1	07-02-2001
		US6128737 A	03-10-2000
		US6226742 B1	01-05-2001
		WO9955039 A1	28-10-1999