

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
2 February 2006 (02.02.2006)

PCT

(10) International Publication Number
WO 2006/012638 A2

(51) International Patent Classification:
H04L 9/28 (2006.01)

(21) International Application Number:

PCT/US2005/026871

(22) International Filing Date: 28 July 2005 (28.07.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/592,310 29 July 2004 (29.07.2004) US

(71) Applicant (for all designated States except US): **VADIUM TECHNOLOGY, INC.** [US/US]; 4507 Pacific Highway East, Suite D, Tacoma, Washington 98424 (US).

(71) Applicant and

(72) Inventor: **ARI, Zsolt** [US/US]; 2304 North 53rd Street, Seattle, Washington 98103 (US).

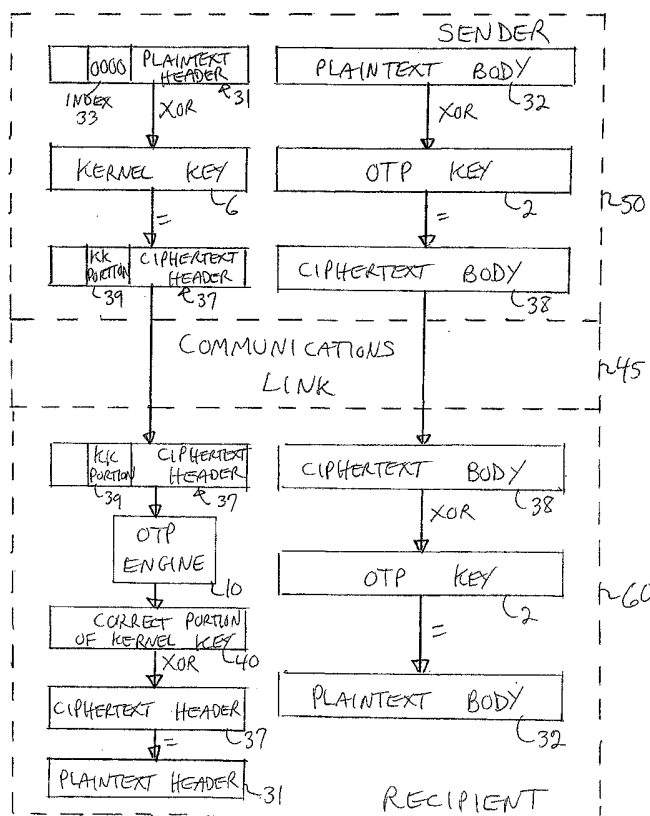
(74) Agent: **RADLO, Edward, J.**; Sonnenschein Nath & Rosenthal, LLP, P.O. Box 061080, Wacker Drive Station - Sears Tower, Chicago, Illinois 60606-1080 (US).

(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.

(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, LV, MC, NL, PL, PT,

[Continued on next page]

(54) Title: TECHNIQUES TO STRENGTHEN ONE-TIME PAD ENCRYPTION



(57) Abstract: Apparati, methods, and computer-readable media for strengthening a one-time pad encryption system. A method embodiment of the present invention comprises the steps of encrypting plaintext (1) with an OTP key (2) in an XOR operation to produce ciphertext (3); and obfuscating the ciphertext (3) with an AutoKey (4) in an XOR operation to produce AutoKeyed ciphertext (5), wherein the AutoKey (4) is a reusable key.



RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA,
GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(15) Information about Correction:

see PCT Gazette No. 11/2006 of 16 March 2006, Section II

Published:

— *without international search report and to be republished
upon receipt of that report*

(48) Date of publication of this corrected version:

16 March 2006

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.

Description**TECHNIQUES TO STRENGTHEN ONE-TIME PAD ENCRYPTION****Inventor:** Zsolt Ari**Related Application:**

This utility patent application is related to, and claims priority benefit under 35 U.S.C. §119(e) to, United States provisional patent application 60/592,310, entitled "Cryptographic Techniques," filed July 29, 2004, which provisional patent application is hereby incorporated by reference in its entirety into the present utility patent application.

Technical Field

This invention pertains to the field of encryption systems using a one-time pad (OTP).

Background Art

U.S. published patent application no. US 2004/0017917 A1, entitled "Cryptographic Key Distribution Using Key Folding," published January 29, 2004, and U.S. published patent application no. US 2004/0136537 A1, entitled "Cryptographic Key Distribution Using Key Unfolding," published July 15, 2004, pertain to improving the distribution of keys used in a one-time pad encryption system. The present invention pertains to strengthening the one-time pad encryption system itself.

Disclosure of Invention

Apparati, methods, and computer-readable media for strengthening a one-time pad encryption system. A method embodiment of the present invention comprises the steps of encrypting plaintext (1) with an OTP key (2) in an XOR

operation to produce ciphertext (3); and obfuscating the ciphertext (3) with an AutoKey (4) in an XOR operation to produce AutoKeyed ciphertext (5), wherein the AutoKey (4) is a reusable key.

Brief Description of the Drawings

These and other more detailed and specific objects and features of the present invention are more fully disclosed in the following specification, reference being had to the accompanying drawings, in which:

Figure 1 is an illustration of a first application of AutoKey 4.

Figure 2 is an illustration of a first application of KernelKey 6.

Figure 3 is an illustration of a second application of KernelKey 6, as well as a general illustration of the functioning of a one-time pad encryption system.

Detailed Description of the Preferred Embodiments

Definitions

The following terms have the following meanings throughout the present specification, including claims.

"AK (AutoKey)" 4 means a reusable key that is used in an XOR operation in the manner described herein and for the purposes described herein.

"GUID (globally unique identifier)" 8 means a series of bits that uniquely identifies an item, in this case a KernelKey 6.

"Key identifier" means a series of bits that identifies which key 2 out of a set of keys 2 should be used for decryption in an OTP encryption system.

"Key offset" means a numerical value indicating the starting point within an OTP key 2 when the key 2 is used in an encryption or decryption procedure in an OTP encryption system.

1 "KK (KernelKey)" 6 means a reusable key that is used in an XOR operation in
2 the manner described herein and for the purposes described herein.

3 "Obfuscate" means the use of a reusable key, such as an AK 4 or a KK 6, in
4 an XOR operation. Obfuscation differs from encryption in that the key used in the
5 XOR operation is a reusable key, as opposed to a true OTP key 2, which is not
6 reusable.

7 "N-byte condition" means the requirement that no consecutive n bytes are
8 repeated within a KK 6, wherein n is a positive integer at least equal to 2.

9 "OTP (one time pad) encryption system" means an encryption system in
10 which the encryption and decryption are performed by an XOR operation, and the
11 encryption/decryption key (OTP key 2) is used just once and then discarded.

12 "OTP engine" 10 means a module, which may be implemented in any
13 combination of hardware, software, and/or firmware, that performs the functions of
14 the OTP encryption system. When implemented in software, the module can be
15 embodied in any computer-readable medium or media, such as computer memory,
16 hard disks, floppy disks, optical disks, etc.

17 "OTP key" 2 is a key that is used for encryption and decryption in an OTP
18 encryption system. The OTP key 2 is used just once and then discarded.

19 "Reusable key," or "revolving key," is a key that is used in an XOR operation
20 and can be reused, by means of using adjacent portions of the key in turn, and when
21 the end of the key is encountered, recommencing with the beginning of the key.

22 "Sliding algorithm" means an algorithm that applies a window having a fixed
23 number of bytes sequentially against an item being examined, such as a key offset,
24 and that performs a comparison at each sequential step between the contents of the
25 fixed length window and the item being examined.

1 "XOR (exclusive OR)" means a Boolean operation in which the combination of
2 a zero and a zero yields a zero; the combination of a one and a one yields a zero;
3 the combination of a zero and a one yields a one; and the combination of a one and
4 a zero yields a one.

5 AutoKey and KernelKey

6 The AutoKey 4 and KernelKey 6 that are described herein can be used in any
7 OTP encryption system, such as the AlphaCipher Encryption System manufactured
8 by Vadium Technology, Inc. of Tacoma, Washington.

9 The basic operation of an OTP encryption system is illustrated in Figure 3. An
10 OTP engine controlled by sender 50 obfuscates plaintext header 31 using KernelKey
11 6 to produce ciphertext header 37, and encrypts plaintext body 32 using OTP key 2
12 to produce ciphertext body 38. The ciphertext header 37 and ciphertext body 38 are
13 then sent over a communications link 45 to recipient 60. Link 45 can be any
14 communications link, such as the Internet, a modulated radio signal, semaphore, etc.
15 An OTP engine 10 controlled by recipient 60 then de-obfuscates ciphertext header
16 37 using KernelKey 6 to recover plaintext header 31, and decrypts ciphertext body
17 38 using OTP key 2 to recover plaintext body 32.

18 Both AutoKey 4 and KernelKey 6 comprise a truly random sequence of bytes,
19 similar to any truly random OTP key 2, but differ from an OTP key 2 in that they are
20 reusable. The length of AutoKey 4 and KernelKey 6 can be any size; the longer they
21 are, the more secure they become. Neither AutoKey 4 nor KernelKey 6 detracts
22 from the operation of the OTP encryption as performed by OTP key 2.

23 AutoKey 4 and KernelKey 6 are revolving keys, meaning that when the end of
24 the key sequence is reached, the sequence starts over from the beginning. The
25 OTP engine 10 controlled by sender 50 keeps track of the portion of AK 4 or KK 6

1 that has been used, and therefore the portion that should be used in the subsequent
2 obfuscation performed by AK 4 or KK 6.

3 KK 6 has a special condition that it must meet: any consecutive n bytes in KK
4 6 must appear only once in that instance of KK 6, wherein n is a positive integer at
5 least equal to 2. This condition is referred to herein as the "n-byte condition." In a
6 preferred embodiment, $n = 4$. This condition limits the length of KK 6; the larger the
7 value of n, the less severe the length limitation. The reason for having this n-byte
8 condition on KK 6 is that it enables an engine 10 to find any n byte sequence easily
9 and uniquely within KK 6, as described below. A quality assurance computer
10 program has been designed to pre-qualify a random series of bytes to meet this
11 condition before it can be used as a KK 6. This quality assurance computer program
12 truncates the series just before the n bytes repeat.

13 As illustrated in Figure 2, KK 6 contains a floating header 7 for unique
14 identification purposes. This floating header 7 is positioned randomly within the
15 random series of bytes used as KK 6. Floating header 7 is removed by OTP engine
16 10 before KK 6 is used for its intended obfuscation purposes. OTP engine 10 has
17 been informed where floating header 7 is positioned within KK 6, and uses that
18 information to extract the floating header 7 from KK 6.

19 Floating header 7 comprises a GUID 8 and a section 9 comprising the first j
20 bytes of AK 4, where j is a positive integer. In a preferred embodiment, $j = 16$. GUID
21 8 uniquely identifies that KK 6. Section 9 enables a unique linkage between KK 6
22 and the AK 4 for that customer. Header 7 is exempt from the n-byte condition,
23 because header 7 is not considered to be part of KK 6 per se.

24 A first purpose of AK 4 is illustrated in Figure 1, namely, the obfuscation of
25 ciphertext 3 to produce AutoKeyed ciphertext 5 after ciphertext 3 has been produced

1 by XORing plaintext 1 with OTP key 2. In Figure 1, the horizontal bar within AutoKey
2 4 reminds the viewer that AutoKey 4 is a repeating key. This obfuscation step is
3 performed so that the decryption/de-obfuscation of the resulting AutoKeyed
4 ciphertext 5 requires the possession of both OTP key 2 and AutoKey 4. If the OTP
5 key 2 is stolen, the thief would still need AutoKey 4 to retrieve the plaintext 1. The
6 thief could recover AutoKey 4, because AutoKey 4 is a repeating key, but it would
7 require an extreme amount of work.

8 A second purpose of AK 4 is to tie a particular installation of the OTP
9 encryption system to a unique set of OTP keys 2. This tying is achieved in that each
10 OTP key 2 contains a floating header that is obfuscated by KK 6, and KK 6 is linked
11 to AK 4 as described above. The effect of this tying is to prevent the unauthorized
12 use of one customer's OTP keys 2 by another customer who is using the same OTP
13 encryption system.

14 The uses of KK 6 are more varied than those of AK 4. A first application of
15 KK 6 is illustrated in Figure 2. After the floating header 7 has been removed from KK
16 6 by engine 10, KK 6 can be used to obfuscate any and all byproducts of the OTP
17 encryption, including, but not limited to, bin file 11, log file 12, and other byproducts
18 13. This obfuscation, which is performed using an XOR procedure as indicated
19 above, produces obfuscated bin file 21, obfuscated log file 22, and obfuscated other
20 byproducts 23, respectively. A bin file 11 is a binary file produced by certain OTP
21 encryption systems, such as the AlphaCipher Encryption System manufactured by
22 Vadium Technology, Inc. of Tacoma, Washington, and is used for storing user
23 preferences. A log file 12 is used by certain OTP encryption systems, such as the
24 AlphaCipher Encryption System manufactured by Vadium Technology, Inc. of Tacoma,
25 Washington, and contains information as to the purpose or purposes of the OTP

1 encryption. Other byproducts 13 can be any other byproducts of OTP encryption
2 produced by any OTP encryption system. Obfuscating bin file 11, log file 12, and
3 other byproducts 13 advantageously makes it harder for nefarious individuals to
4 eavesdrop on the encrypted communications.

5 Both AK 4 and KK 6 are used as constants within an organization, enterprise,
6 or communications group. There is a limitation on this constancy, in that one or both
7 of AK 4 or KK 6 may be lost, stolen, or corrupted. In this case, AK 4 and KK 6 must
8 be replaced or replenished.

9 Normally, there is one set of OTP keys 2, one AK 4, and one KK 6
10 (collectively, "group") per customer. It is possible for there to be several groups per
11 customer, but this would prevent communications from one group to another group.
12 It is also possible for there to be one KK 6 and several AKs 4 per customer, but
13 again the subgroups within the customer having separate AKs 4 would not be able to
14 communicate with each other. In this latter embodiment, let us assume that there
15 are m AKs, where m is a positive integer at least equal to 2. Then there are m KKs,
16 all identical except for a different header 7. Each header 7 has the same GUID 8 but
17 a different AK portion 9, so that all m of the AKs are pointed to by the set of m KKs.
18 Then after the headers 7 are stripped from the KKs by engine 10, there is just one
19 unique KK 6.

20 A second use of KK 6 is illustrated in Figure 3. It is typical in an OTP
21 encryption system for the plaintext 1 to consist of a plaintext body 32 and a
22 conjoined plaintext header 31. Body 32 contains the substantive portion of the
23 message 1 that is to be encrypted. Header 31, which typically appears at the
24 beginning of plaintext 1, contains information that is useful in the decryption process.
25 Such information can include the identity of the intended recipient 60, the length of

1 plaintext body 32, the identity of an OTP key 2, and the key offset for said OTP key
2 2. Body 32 is encrypted by OTP key 2 using an XOR procedure, to produce
3 ciphertext body 38. Plaintext header 31 is obfuscated by KernelKey 6 (after the
4 floating header 7 has been removed from KK 6 by the sender's engine 10), to
5 produce ciphertext header 37. The reason that header 31 is obfuscated is so that
6 attackers will not be able to get access to said useful information.

7 In the present invention, the ciphertext header 37 is de-obfuscated without the
8 use of a key identifier or key offset, using a special indexing system that takes
9 advantage of the n-byte condition of the KK 6 defined above. This indexing
10 technique works as follows: plaintext header 31 contains a unique n-byte index 33
11 that has been carved out of KK 6. Index 33 is always in the same location within
12 plaintext header 31, and therefore each OTP engine 10 knows where to find index
13 33. Index 33 is initially set to zero (all of its bits are zeroes). When plaintext header
14 31 is obfuscated by KK 6, whatever is present within KK 6 at the same relative
15 location as index 33 is carried over into a corresponding KK portion 39 within
16 ciphertext header 37, due to the very nature of the XOR operation, i.e., a zero
17 XORed with any given bit yields that bit. In a subsequent step, the OTP engine 10 of
18 the recipient 60 goes to KK portion 39 to find the starting point of the portion 40 of
19 KK 6 that was used in the obfuscation and hence must be used in the de-
20 obfuscation. (In any OTP process, the same portion of the key must be used for the
21 decryption (or de-obfuscation) as was used for the encryption (or obfuscation)).
22 Because of the n-byte condition, we know that this portion 40 of KK 6 is unique.
23 Therefore, we know that the de-obfuscation will be performed correctly. This de-
24 obfuscation consists of the correct portion 40 of KK 6 being XORed with ciphertext
25 header 37 to recover plaintext header 31.

Variable Persistent Headers

The material herein pertaining to variable persistent headers applies equally to KK headers 7 and plaintext headers 31, but for purposes of ease of illustration, the following discussion is directed to plaintext headers 31. In order for header 31 to perform its various functions, at least some of header 31 must persist (carry over from sender 50 to recipient 60). Traditionally, header 31 has been written in such a way that it could be expanded should the need arise in the future to include additional data, such as, for example, biometric information pertaining to the holder of the OTP key set 2. This has traditionally been done by using a size counter, wSize, as the first data member (field) of header 31, indicating the length of header 31. This allowed for the definition of a header 31 with known data members, with the possibility of adding more data members to the end of the data structure comprising header 31 as needed. This principle is illustrated in the following C language example:

```
struct
{
    WORD        wSize;
    WORD        wDay;
    WORD        wMonth;
    WORD        wYear;
} DATE_STRUCT
```

DATE_STRUCT defines storage for a date as it would be persisted on a computer-readable medium of the recipient 60. The wSize member variable should be preset to be equal to the size of DATE-STRUCT. If, in the future, the structure needed to be expanded while backward-supporting the original DATE_STRUCT, a new structure with the first few data members being identical to DATE_STRUCT would have to be created, as shown below:

```
struct
```

```
1  {
2      WORD          wSize;
3      WORD          wDay;
4      WORD          wMonth;
5      WORD          wYear;
6      WORD          wHour;
7      WORD          wMinute;
8      WORD          wSecond;
9  } DATE_TIME_STRUCT
10
```

11 In this second example, the wSize member variable is preset to equal the size
12 of DATE_TIME_STRUCT. With these two data structures in place, the old data
13 structure could be distinguished from the new data structure by examining the value
14 of the wSize member variable, as it would indicate whether additional information
15 was present. While this method has been used frequently, it has drawbacks.
16 Suppose, at a later date, the data structure needed to be expanded to, for example,
17 keep track of the number of days since January 1st of the current year. The only
18 place to add this information would be at the end of the data structure, so a third
19 DATE_TIME_STRUCT definition would be required. As another example, suppose
20 that it was no longer necessary, at a later date, to keep track of the year. Yet
21 another data structure would be needed to eliminate the data member wYear.
22 These data structures are static and are therefore nearly impossible to change once
23 they have been released to the public. This creates a housekeeping problem and
24 wastes processing cycles.

25 From a security standpoint, there is another problem with the static headers of
26 the prior art. Suppose that a 64-bit number is used as a key offset for OTP key 2,
27 and that the key offset is stored within header 31. When the OTP key 2 is used for
28 the first time, the key offset will be zero. Since a zero XORed with any value will
29 return the XORed value, a sliding algorithm could be used by an attacker (who might
30 intercept the header as it traverses the communications link 45) to reliably detect the

1 section of the OTP key 2 that the rest of the information around the offset was
2 encrypted with, thereby weakening security.

3 The present invention solves these problems caused by static headers by
4 using a header 31 where the first member variable (wFlags) consists of a set of flags
5 that determine the contents of the rest of the header 31 as it is persisted. An
6 example of a DATE_TIME_STRUCTURE using the present invention is:

```
7 struct  
8 {  
9     WORD          wFlags;  
10    WORD          wDay;  
11    WORD          wMonth;  
12    WORD          wYear;  
13    WORD          wHour;  
14    WORD          wMinute;  
15    WORD          wSecond;  
16 } DATE_TIME_STRUCT  
17
```

18 In the above DATE_TIME_STRUCT, wFlags is an arbitrary number of bits
19 long. Each bit of wFlags represents a different member variable in the remainder of
20 the data structure. In an alternative embodiment, a given bit within wFlags
21 represents more than one data member, such as the combination "wDay wMonth
22 wYear." The meaning of the bits within wFlags is agreed to in advance by sender 50
23 and recipient 60 according to a set of rules ("convention" or "persist code"). Let us
24 assume that the initial value of wFlags is established by sender 50 as 110110. Let
25 us further assume that the convention stipulates that these six bits within wFlags
26 refer to wDay, wMonth, wYear, wHour, wMinute, and wSecond, respectively. Thus,
27 wFlags contains six individual one-bit flags. Flags 1, 2, 4, and 5 are "set" (have a
28 value of 1), and flags 3 and 6 are "cleared" (have a value of 0). This means that
29 sender 50 wishes for the values of wDay, wMonth, wHour, and wMinute to persist,
30 and for the values of wYear and wSecond to not persist. The reason for wanting to

1 persist only certain variables and not others is to save processing time and space.
2 For example, sender 50 might have concluded that wYear is not important, because
3 all of the communications are going to take place in the same year, and that
4 wSecond is not important, because the messages 38 don't need to be tracked with
5 that level of specificity. When the recovered persisted plaintext header 31 is read
6 into a computer-readable medium of recipient 60 for processing by OTP engine 10,
7 the first field that is read in is wFlags, which can then be used to read in the rest of
8 the persisted header 31. Data is typically read in a word (i.e, two bytes) at a time. At
9 the end of each word, a bit can be used to inform recipient's engine 10 as to whether
10 additional portions of wFlags are present and therefore must be read in. In the case
11 of a flag that has been cleared, engine 10 will assign a default value to that particular
12 member variable. The defaults can be different for the different data members,
13 according to the convention. For example, wYear can default to the current year,
14 and wSecond can default to zero. If a flag has been set, the corresponding member
15 variable is initialized with the value in the persisted header 31.

16 In the above method, the order of member variables depends upon the pre-
17 selected convention. For example, it is possible to persist wHour before wDay. To
18 not persist a member variable, its flag is cleared; and to add a new member variable,
19 it is not necessary to append it to the end of the data structure, since the convention
20 is customizable. The convention can be changed after the OTP encryption system
21 has been initially deployed, e.g., by sending a software patch to sender 50 and
22 recipient 60 changing the mapping of the flag bits to the data members.

23 When the OTP encryption system is first deployed, modules containing OTP
24 keys 2, OTP engines 10, and conventions pertaining to wFlags are distributed to the
25 prospective senders 50 and recipients 60. These modules can be implemented in

any combination of hardware, firmware and/or software. When implemented in software, the modules can be embodied in any computable-readable medium or media, such as computer memory, hard disks, floppy disks, optical disks, etc. The initial convention for wFlags can specify that a certain subset of bits within wFlags is reserved for future use. The recipient's engine 10 encountering the value of 1 in any of these bits will try to process the bit, but conclude it cannot do anything with it. Later, after a patch has been sent out to sender 50 and recipient 60 amending the convention to define the significance of this bit within wFlags, the recipient's engine 10 will be able to process said bit.

A portion of header 31 is normally devoted to the key offset for OTP key 2. Let us assume that the key offset is 64 bits long. During the early stages of life of OTP key 2, the beginning portions of the key offset will be zero. Sending these initial zeroes over communications link 45 wastes processing cycles, and presents a security risk, since an attacker could use a sliding algorithm against the key offset portion to undesirably obtain information as to which portion of KK 6 was used in the obfuscation of the key offset portion of plaintext header 31. As a solution to this problem, one or more bits within wFlags can be devoted to indicating a portion of the key offset that is persisted. For example, flags within wFlags could be used to indicate whether 8 bits, 16 bits, 32 bits, or 64 bits of the key offset are persisted.

In summary, the use of flags as described above allows great flexibility in persisting information contained within header 31. The information that is persisted is determined by flags within wFlags, and the order in which the information is used is customizable for the particular application, and can be changed, as long as both the sender 50 and the recipient 60 are informed of the convention being used.

1 The above description is included to illustrate the operation of the preferred
2 embodiments and is not meant to limit the scope of the invention. The scope of the
3 invention is to be limited only by the following claims. From the above discussion,
4 many variations will be apparent to one skilled in the art that would yet be
5 encompassed by the spirit and scope of the present invention.

What is claimed is:

Claims

1. A computer-implemented method for strengthening an OTP encryption system, said method comprising the steps of:

encrypting plaintext with an OTP key in an XOR operation to produce ciphertext; and

obfuscating the ciphertext with an AutoKey in an XOR operation to produce AutoKeyed ciphertext, wherein the AutoKey is a reusable key.

2. The method of claim 1 wherein the AutoKey is uniquely linked to a set of OTP keys.

3. The method of claim 1 wherein the AutoKey is uniquely linked to a KernelKey, wherein the KernelKey is a reusable key.

4. The method of claim 3 wherein the plaintext comprises a plaintext header and a plaintext body, said method further comprising the step of obfuscating the plaintext header with the KernelKey.

5. The method of claim 3 further comprising the step of obfuscating byproducts of the OTP encryption with the KernelKey.

6. The method of claim 5 wherein the byproducts comprise items from the group of items consisting of:

bin files storing user preferences; and

log files containing information as to a purpose of the OTP encryption.

1 7. A computer-implemented method for strengthening an OTP encryption
2 system, said method comprising the steps of:

3 obfuscating a plaintext header with a KernelKey in an XOR operation to
4 produce a ciphertext header, wherein:
5 the KernelKey is a reusable key; and
6 the plaintext header contains information useful in
7 processing an associated plaintext body.

8 8. The method of claim 7 wherein the information comprises items from
9 the group of items consisting of:

10 identity of an intended recipient of the plaintext body;
11 length of the plaintext body;
12 identity of an OTP key; and
13 key offset of said OTP key for use in decrypting the plaintext body.

14 9. The method of claim 7 further comprising the step of de-obfuscating
15 the ciphertext header, thereby recovering the plaintext header, by combining the
16 ciphertext header with the KernelKey in an XOR operation without using a key offset.

17 10. The method of claim 9 wherein the de-obfuscating step is performed
18 using an indexing technique comprising:

19 selecting a preselected index location within the plaintext header; and
20 finding said index location within the KernelKey.

21 11. The method of claim 7 wherein no consecutive n bytes are repeated
22 within the KernelKey, where n is a positive integer at least equal to 2.

1 12. The method of claim 7 wherein the KernelKey contains a randomly
2 located header comprising:

3 a GUID; and

4 a portion of an AutoKey, wherein the AutoKey is a reusable key.

5 13. The method of claim 12 wherein the AutoKey is used to obfuscate
6 ciphertext that is produced by the OTP encryption system.

7 14. A variable persistent header for use in a one-time pad encryption
8 system, said header comprising:

9 a plurality of fields containing information useful in processing a
10 message that is encrypted using the encryption system; and

11 a set of flags indicating which of said fields are persisted from a sender
12 of the message to a recipient of the message.

13 15. The header of claim 14 wherein said header comprises at least one of:

14 identity of an intended recipient of the message;

15 length of the message;

16 identity of an OTP key; and

17 key offset of said OTP key for use in decrypting the message.

18 16. The header of claim 14 wherein the header is contained within a
19 KernelKey that is used to perform at least one of the following two tasks:

20 obfuscate byproducts of the encryption;

21 obfuscate a header that is useful in processing the message.

22 17. The header of claim 14 wherein there is a one-to-one mapping
23 between the flags and the fields.

1 18. The header of claim 14 wherein there is a one-to-many mapping
2 between at least one of the flags and the fields.

3 19. The header of claim 14 wherein meanings of the flags are agreed to by
4 a sender of the message and a recipient of the message prior to the message being
5 sent from the sender to the recipient.

6 20. The header of claim 19 wherein meanings of the flags can be changed
7 after the message has been sent, and before a subsequent message is sent.

8 21. At least one computer-readable medium containing computer program
9 instructions for strengthening an OTP encryption system, said computer program
10 instructions performing the steps of:

11 encrypting plaintext with an OTP key in an XOR operation to produce

12 ciphertext; and

13 obfuscating the ciphertext with an AutoKey in an XOR operation to

14 produce AutoKeyed ciphertext, wherein the AutoKey is a

15 reusable key.

16 22. At least one computer-readable medium containing computer program
17 instructions for strengthening an OTP encryption system, said computer program
18 instructions performing the steps of:

19 obfuscating a plaintext header with a KernelKey in an XOR operation to

20 produce a ciphertext header, wherein:

21 the KernelKey is a reusable key; and

22 the plaintext header contains information useful in

23 processing an associated plaintext body.

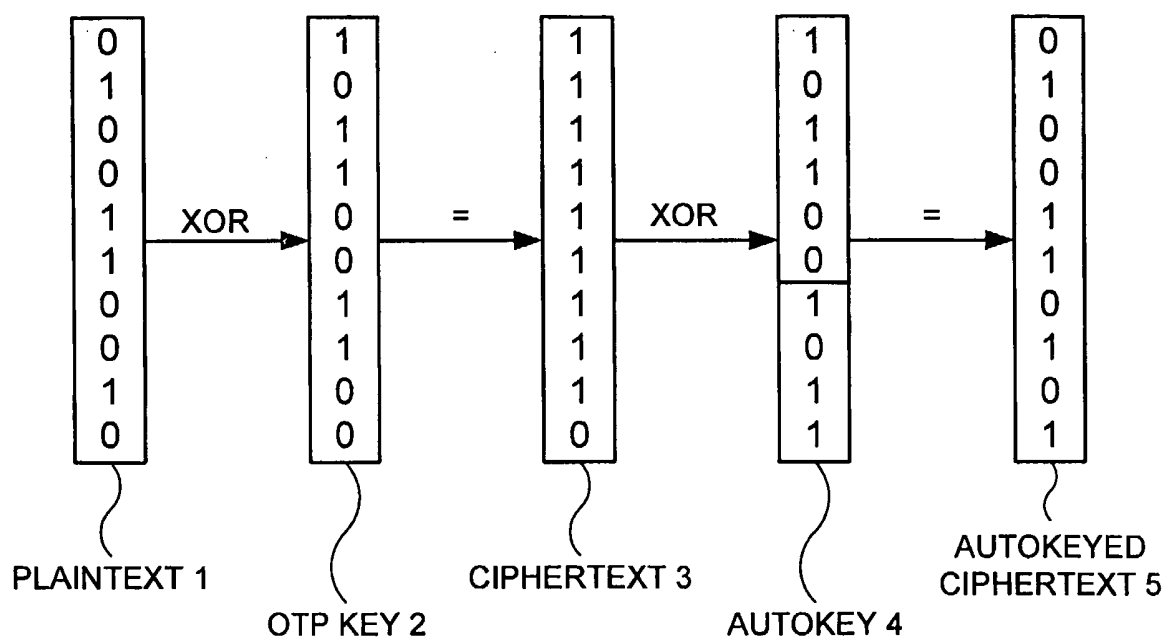


FIG. 1

2/3

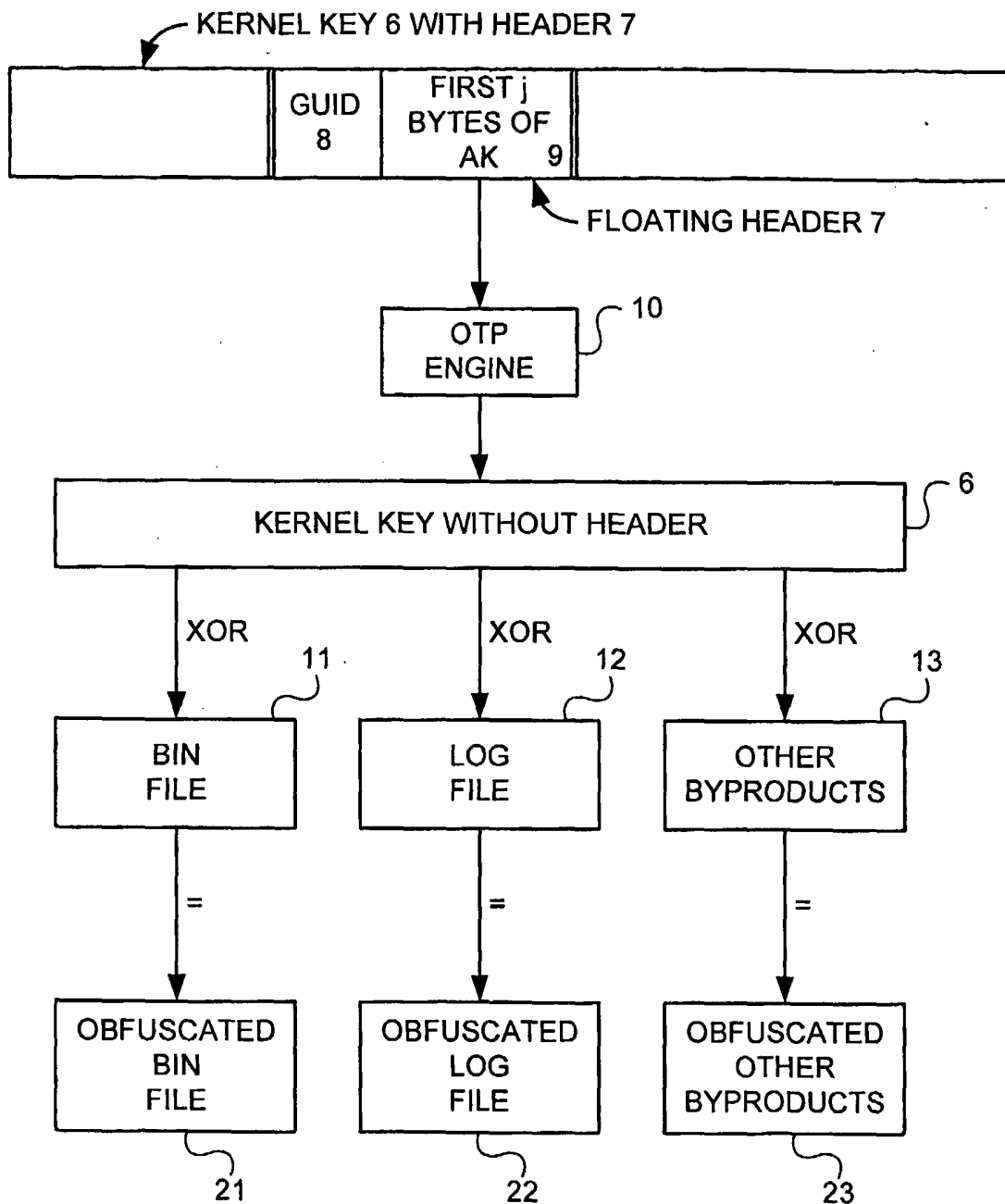


FIG. 2

3/3

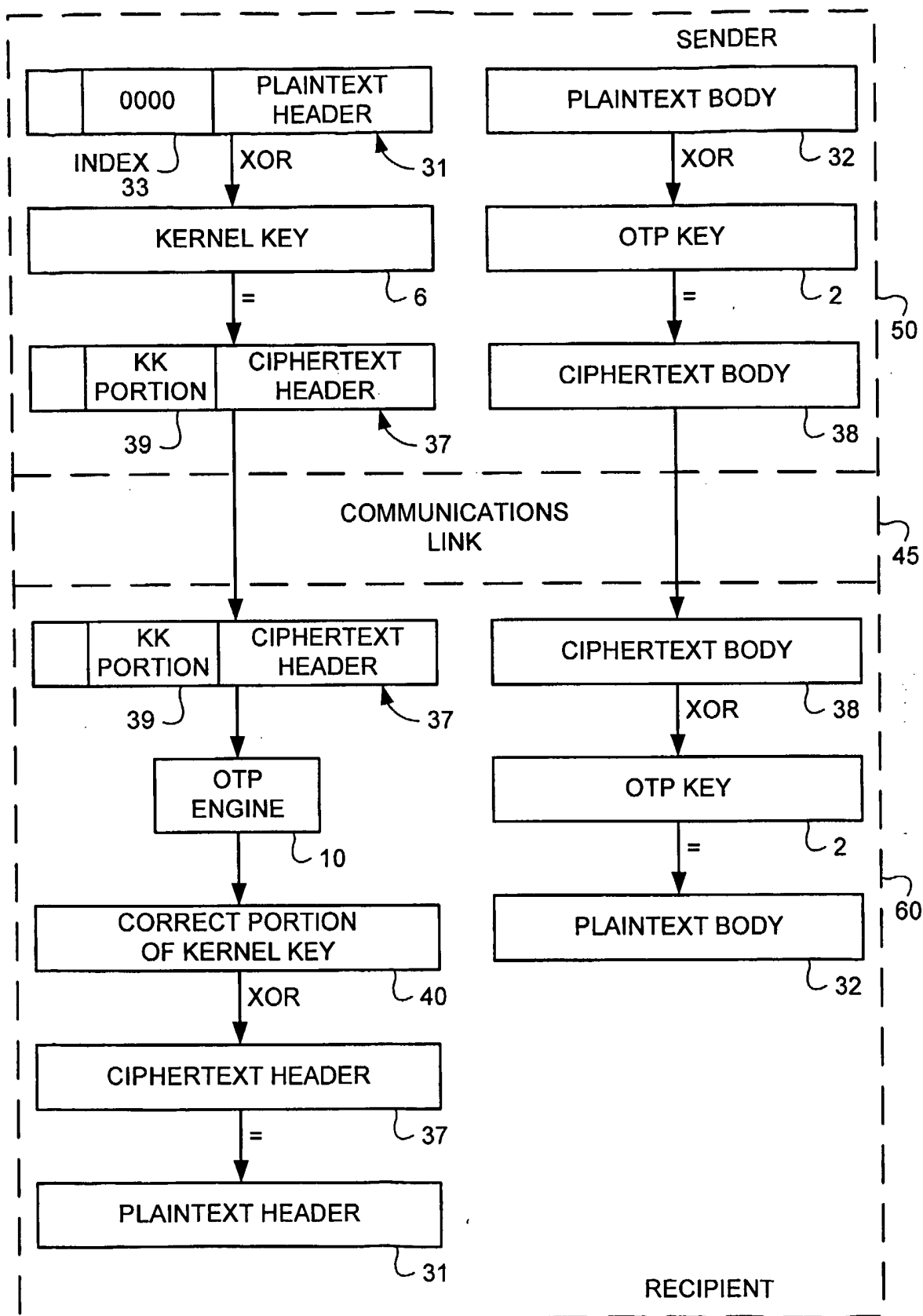


FIG. 3