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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 November 2007 (08.11.2007)

PCT

(10) International Publication Number
WO 2007/126763 A2

(51) International Patent Classification:
H04L 9/32 (2006.01)

(21) International Application Number:
PCT/US2007/007483

(22) International Filing Date: 27 March 2007 (27.03.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/407,996 21 April 2006 (21.04.2006) US

(71) Applicant *(for all designated States except US)*: **VERISIGN, INC.** [US/US]; 487 East Middlefield Road, Mountain View, CA 94043 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **M'RAIHI, David** [FR/US]; 762 Cedar Street, San Carlos, CA 94070 (US).

(74) Agents: **MORRIS, Gary, S.** et al.; TOWNSEND AND TOWNSEND AND CREW LLP, Two Embarcadero Center, 8th Floor, San Francisco, CA 941 11 -3834 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, 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, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

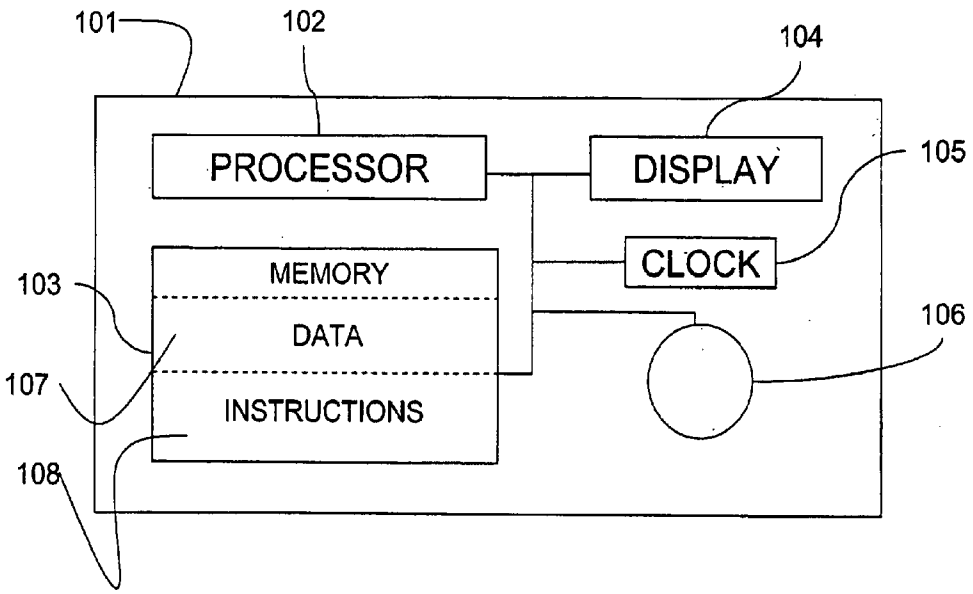
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

(54) Title: TIME AND EVENT BASED ONE TIME PASSWORD



(57) Abstract: A system and method for generating a One Time Password (OTP) based upon a value TEC that can change based both upon the occurrence of an event and the passage of time. The OTP can be computed at a token and sent to a verifier. The verifier stores exact or estimated parameters necessary to compute one or more expected OTPs from the token, including TEC. The value TEC can be synchronized between the token and the verifier.

WO 2007/126763 A2

Time and Event Based One Time Password

Related Applications

[0001] This application is related to U.S. Provisional Patent Appln. 60/618,600 filed on October 15, 2004 and PCT Application No. PCT/US05/37113, filed on October 17, 2005.

Field of the Invention

[0002] The field of the invention is authentication, in particular authentication using one-time passwords.

Background of the Invention

[0003] A user can be authenticated using a token that stores a secret shared only with an authentication service. The token generates a one-time password ("OTP") for each authentication based upon the shared secret and other values. When the OTP is received at the authentication service, the service calculates at least one expected OTP value and compares it to the received OTP value. If they match, the user is authenticated. Otherwise, the user is not authenticated.

[0004] An OTP can be automatically, periodically generated based upon the shared secret and a value that changes as a function of time. For example, a new OTP can be automatically generated every minute based upon the shared secret and a value that corresponds to the date and time of day. The authentication service calculates a series of expected OTPs based upon the shared secret and a range of date/time values meant to encompass the date/time value used at the token. This takes into account any drift between the clock on the token and the clock at the service that causes the token and service to be out of synchronization. If one of the OTPs calculated at the service match the received OTP, then the user is authenticated. Security is enhanced because one of parameters used to calculate the OTP changes over time, making it difficult to guess.

[0005] An OTP can be calculated based upon the shared secret and a characteristic of an event. For example, the characteristic of an event can be the value of a counter that is incremented each time the user pushes a button on

the token. The authentication service calculates a range of OTPs based upon the shared secret and counter values beginning at the last known counter value at the token. This takes into account button-pushes that resulted in OTPs that were not sent to the service. If one of the OTPs calculated at the service match the received OTP, then the user is authenticated.

[0006] Numerous algorithms are known for generating OTPs, including the HOTP algorithm, which is specified in RFC 4226. The following symbols can be used in describing the HOTP algorithm.

Symbol	Represents
C	8-byte counter value, the moving factor. This counter MUST be synchronized between the HOTP generator (prover) and the HOTP validator (verifier).
K	shared secret between prover and verifier; each HOTP generator has a different and unique secret K.
Digit	number of digits in an HOTP value; system parameter.

[0007] The HOTP algorithm is based on an increasing counter value and a static symmetric key known only to the token and the validation service. In order to create the HOTP value, we will use the HMAC-SHA-1 algorithm, as defined in RFC 2104. As the output of the HMAC-SHA-1 calculation is 160 bits, we must truncate this value to something that can be easily entered by a user.

$$HOTP(K,C) = Truncate(HMAC\text{-}SHA\text{-}1(K,C))$$

where "Truncate" represents the function that can convert an HMAC-SHA-1 value into an HOTP value. The Key (K), the Counter (C), and Data values can be hashed high-order byte first. The HOTP values generated by the HOTP generator can be treated as big endian.

Generating an OTP can be done in the 4 following steps:

Step 1: Generate an HMAC-SHA-1 value

Let HS = HMAC-SHA-1(K,C) // HS is a 20-byte string

Step 2: Generate a 4-byte string (Dynamic Truncation)
Let Sbits = DT(HS) // DT, defined below, returns a 31-bit string

Step 3: Compute an HOTP value
Let Snum = StToNum(Sbits) // Convert S to a number in
0...2^{31}-1
Return D = Snum mod 10^Digit // D is a number in the range
0...10^{Digit}-1

The Truncate function can perform Step 2 and Step 3, i.e., the dynamic truncation and then the reduction modulo 10^Digit. The dynamic offset truncation technique can extract a 4-byte dynamic binary code from a 160-bit (20-byte) HMAC-SHA-1 result.

DT(String) // String = String[0]...String[19]
Let OffsetBits be the low-order 4 bits of String[19]
Offset = StToNum(OffsetBits) // 0 <= OffSet <= 15
Let P = String[OffSet]...String[OffSet+3]
Return the Last 31 bits of P

Masking the most significant bit of P can avoid confusion about signed vs. unsigned modulo computations. Different processors can perform these operations differently, and masking out the signed bit can remove ambiguity.

[0008] RFC 4226 requires implementations to extract a 6-digit code at a minimum and possibly 7 and 8-digit code. This requirement can be violated under different circumstances, although the result may not be compliant with the RFC. Depending on security requirements, Digit = 7 or more should be considered in order to extract a longer HOTP value result.

[0009] Here is an example of using this technique for Digit = 6, i.e., a 6-digit HOTP value is calculated from the HMAC value. The following code example describes the extraction of a dynamic binary code given that hmac_result is a byte array with the HMAC-SHA-1 result:

```
int offset   =  hmac_result[19] & 0xf ;
int bin_code = (hmac_result[offset]  & 0x7f) << 24
   | (hmac_result[offset+1] & 0xff) << 16
   | (hmac_result[offset+2] & 0xff) <<  8
   | (hmac_result[offset+3] & 0xff) ;
```

SHA-1 HMAC Bytes (Example)

Byte Number	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
Byte Value	1f	86	98	69	0e	02	ca	16	61	85	**50**	**ef**	**7f**	**19**	da	8e	94	5b	55	5a

The last byte (byte 19) has the hex value 0x5a.
The value of the lower 4 bits is 0xa (the offset value).
The offset value is byte 10 (0xa).
The value of the 4 bytes starting at byte 10 is 0x50ef7f19, which is the dynamic binary code DBC1.
The MSB of DBC1 is 0x50 so DBC2 = DBC1 = 0x50ef7f19.
HOTP = DBC2 modulo 10^6 = 872921.

The dynamic binary code can be treated as a 31-bit, unsigned, big-endian integer; the first byte is masked with a 0x7f. This number can then be taken modulo 1,000,000 (10^6) to generate the 6-digit HOTP value 872921 decimal.

[0010] A disadvantage of using a counter value as an input to compute an OTP is that it can remain the same for a long period of time. For example, if a user does not use his token for several months, the counter value will persist at the same value for the entire period of time between uses. Under certain circumstances, this may allow a third party sufficient time to analyze the token,

guess the exact or approximate value of the counter and possibly compromise the security of the authentication system.

Brief Description of the Drawings

[0011] Figure 1 shows an apparatus in accordance with an embodiment of the present invention.

[0012] Figure 2 shows the method in accordance with an embodiment of the present invention.

Detailed Description

[0013] In accordance with an embodiment of the present invention, the counter value C can be changed as a function of time. Any suitable function of time may be used. A counter C that is changed based upon the occurrence of an event and based upon the passage of time is called a Time and Event-based Counter or TEC. For example,

$$TEC = F(X,Y,W,Z)$$

where F is any suitable function or other mapping, X is the size of the time-based increment of TEC, Y is related to the frequency with which increment X is added to TEC, W is the size of the even-based increment of TEC and Z is the number of occurrences that need to occur before TEC is incremented by W. Note that X and W may be positive or negative numbers and that, in the general case, TEC may be an integer or any other kind of suitable number. For example, TEC can be incremented by X every Y seconds both at the prover (e.g., a token) and at the verifier (e.g., an authentication service.) Likewise, TEC can be incremented by W every Z occurrences of an event E or events from a set of events {E}. An actual OTP value can be computed at the token after an event that increments TEC. For example, TEC can be incremented by 1 every 60 seconds in the background as follows:

Time	Event	TEC	OTP Calculation
23:41:00	None	341	None
23:42:00	None	342	None
23:43:00	None	343	None
23:43:17	Button Push	344	872921
23:44:00	None	345	None

This example shows a linear function for incrementing TEC, but any suitable function or mapping for X and Y may be used.

$$X = G(X_i),\ i = 1,2\ \ldots.$$

$$Y = H(Y_j),\ j = 1,\ 2,\ \ldots$$

X_i and Y_i may be numbers or variables. For example, X may be modulo 8 of a function of the shared secret, Y may be equal to modulo 10 of a pseudorandom number that can be generated at the prover and verifier. X and/or Y could be based upon the intermediate or final results of previous OTP computations, e.g., by using all or part of the last byte from the full value computed for the previous OTP, e.g., the value before truncation. For instance, the last byte (byte 19) of the previous HMAC computation can be stored after each computation and used for this purpose.

[0014] X and Y can also be determined in other ways. For example, Y could be based upon a system parameter, such as the value of a clock at the prover. Different token families could have different time steps. A token for accessing a bank account could operate with a smaller value for Y, e.g., 2 or 3 minutes, but a token for accessing an email account could operate with a larger value, e.g., 30 minutes.

[0015] TEC can be incremented "in the background" at both the prover and the verifier. Any suitable event may trigger the incrementation of TEC and computation of the OTP. For example, an OTP value can be computed at the token when the button on the token is pressed. Likewise, a token may be a "soft token " implemented as a java applet in a mobile phone, in an application running on a PC, etc. In these cases, the event triggering a computation of an OTP can include depressing a given key on a cell phone keypad or computer keyboard, pressing a sequence of keys, pushing a special button, the scanning of a biometric, etc. Similarly, an OTP can be triggered by the selection of an item on a toolbar in a graphical user interface displayed on a screen, by the selection of an item on a pulldown menu, by the activation of an application

program, by the activation of a feature of an application program, or by any other suitable event.

[0016] The initial value of TEC at the prover can be 1 or any other suitable value. The initial value of TEC can be set by the user, set by the authentication service, set by an application program, set by an authorized third party, etc. When the initial value at the prover is set, the initial value (or information sufficient to derive the initial value) should be sent to the verifier. In this way, the verifier can synchronize its value of TEC for the prover with the value of C at the prover itself. A default value can be specified for the value of TEC absent any new setting of TEC. Initial and default values may be similarly set for X and Y. Initial values can be kept secret as between the prover and verifier.

[0017] The prover and verifier should be synchronized and also possibly re-synchronized. In one embodiment, the clock can start incrementing TEC when the token is activated. In another embodiment, the clock can start incrementing TEC when the token is manufactured; a resynchronization pass can occur when the token is activated with the validation service. There can be a simple process for starting the clock, incrementing the value and computing the OTP value, such as depressing the button for 5 full seconds, pushing the button four times within three seconds, etc. The verifier can calculate a set of candidate OTPs for a given prover based upon the verifier's last known prover value TEC and present time known by the verifier. A range around these last known values can be used. If a verifier-computed OTP matches an OTP received from a prover, the verifier can update and store the values for TEC and time for that prover.

[0018] In accordance with embodiments of the present invention, an OTP can be generated by incrementing a value TEC based upon the occurrence of an event and incrementing this value based upon the passage of time. The OTP can be determined based upon TEC. The OTP can also be computed based upon TEC and an adjunct value, "AV". This adjunct value AV can be a shared secret, such as a cryptographic key, a password, etc. AV can be based upon secret and public information. Such public information can be a challenge, a question, etc. TEC can be incremented based upon the occurrence of any suitable event, such as the pressing of a button, selecting a dialog box, meeting of a condition (e.g.,

correctly responding to a challenge), etc. The passage of time can be measured by a value T that can be different for each application. TEC can be incremented after T seconds, T minutes, etc. TEC can also be incremented by a value TI, where after T seconds, TEC can be incremented by TI. For example, after T=60 seconds TEC can be incremented by TI=1. TEC can be controlled by a value EI, where after each event E occurs, TEC can be incremented by EI. For example, after the user presses a button, TEC can be incremented by EI=3. In one embodiment, the one time password is computed using the algorithm specified in RFC 4226.

[0019] In accordance with embodiments of the present invention, a user can activate a token by connecting to an activation web site. The activation web site can authenticate the user using any suitable technique, e.g., by requesting and verifying a username and password. The user can also provide any suitable additional information, such as a token identifier, token serial number, associated credentials, etc. This information can be used (along with one or more OTP values generated by the token and sent to the activation web site) to prove that he does in fact has the particular token. The token (e.g., the token identifier) can thus be linked to a specific user (e.g., username) and can initiate the storage and/or updating of a time value in the token, e.g., based upon data generated by the token clock. Alternatively, the clock signal can originate from outside of the token. The time value can help the verifier to determine the time difference between the moment when the user obtains an OTP value and the time at which the OTP value is received at the verifier.

[0020] An embodiment of the invention can also compute an HOTP value based upon Counter (C), Key (K) and an adjunct value (AV). Adjunct value AV can be based upon a private secret (e.g., a private key), a shared secret (e.g., a password) or a combination of a secret (private or shared) and some public information. Public information is any information that is not a shared or private secret. An example of public information is a user identifier, a challenge, an answer to a question, etc.

[0021] A time window can be set for a token or set of tokens. A re-synchronization of a token can occur when the time elapsed between the actual generation of the OTP (on the token) and the reception of the OTP value by the verifier is longer than a time window that has been set for a particular token or

set of tokens. When a token is turned off, e.g., when a soft token on a Cell phone, PC, etc., the soft token value has to be re-synchronized with the verifier value. Let the_device with the soft token be switched off at time T0 and be dormant until it is switched back on at time T1. Let t be the time period for incrementing TEC by EI. For example, value TEC is increment by EI every t seconds. A re-synchronization can be performed as follows:

1- Compute T = T1 – T0
2- Compute Count = Floor(T / t) * EI
3- Increment TEC by Count
4- Resynch the token clock (with respect to the differential T)

After this resynchronization procedure, TEC is normally incremented by EI every t period of time.

Another TEC-based resynchronization method can be used whenever the prover can send the OTP value but also other information, and more specifically the TEC value. The prover can send his OTP (denoted OTP.prover) and TEC (denoted TEC.prover) values and the OTP can act as a short message authentication code of TEC.
The verifier can resynchronize within a certain range of values determined by an upper bound value of TEC (denoted TEC.bound) as a system parameter.
The verifier can accept the OTP if the following are all true, where TEC.verifier is the verifier's present value of TEC that is associated with the prover:

1. Prover can send OTP.prover and TEC.prover
2. Verifier can check that TEC.prover ≥ TEC.verifier
3. Verifier can check that (TEC.prover – TEC.verifier) ≤ TEC.bound
4. Verifier can check that OTP.prover is correct for TEC.prover
5. If true, the verifier can set TEC.verifier to TEC.prover

After this resynchronization procedure, TEC is normally incremented by EI for every t period of time that elapses. Likewise, the same or a similar protocol can be used to resynchronize the TEC value at the prover.

[0022] Even when a device is not turned off, the clock on the device can drift away from being synchronized with other clocks, e.g., with clocks at verifiers. One way to resynchronize without specific intervention from the end

users is to maintain a "window" around the normal value of TEC. When a verifier receives an OTP from a token, e.g. an OTP calculated at the token and sent to the verifier, it can compute the expected value of TEC based upon its stored parameters for the sending token, e.g., the present number of event occurrences for that token (e.g., the number of times the button has been pushed on that token, based, for example, on the number of OTP verification requests received by the verifier thus far for that token), the time, etc. Recognizing that its stored parameters may be slightly inaccurate in relation to the parameters at the token used to generate the OTP, the verifier can calculate several candidate OTPs based upon a range of magnitudes of parameters near the expected magnitudes for those parameters. For example, the verifier could check the OTP and if is not correct, compute the OTP for C+1, C-1, C+2, etc., until a match is found or the size of the window (e.g., within plus or minus 10 of the expected C) is exceeded.

[0023] The verifier could implement different level of securities:

1- simply accept the OTP value in this window; or

2- require the user to provide another OTP (the OTP will be different even if the time slot is the same because pressing the button (causing the occurrence of an event) will modify the counter value and generate a different OTP from the previous one; or

3- differentially accept an OTP for some actions, refuse the OTP for others, eventually also accept for the first 2-3 resynchronizations and then ask for another OTP, propose to replace the token and/or inform technical support that there is a synchronization issue with a specific user, etc.

[0024] A system in accordance with an embodiment of the present invention is shown in Figure 1. Token 101 can include a processor 102 coupled to memory 103, display 104, clock 105 and button 106. Memory 103 can store data 107, such as shared secrets, such as a password, a Personal Identification Number (PIN), cryptographic key, etc. Memory 103 can also store counter C and a value based upon public information. Memory 103 can also store instructions 108 adapted to be executed by processor 102 to perform at least part of the method in accordance with the present invention.

[0025] Processor 102 can be a general purpose microprocessor or an Application Specific Integrated Circuit that embodies at least part of the method of the present invention in its hardware and/or firmware. An example of a general purpose microprocessor is the Pentium IV processor manufactured by the Intel Corporation of Santa Rosa, California. An example of an ASIC is a digital signal processor.

[0026] Memory 103 can be any device that can store electronic information. Examples of memory 103 can include Random Access Memory, Read Only Memory, Electronic Programmable Read Only Memory, flash memory, a hard disk, a CD, etc. These are also examples of a medium that can store the instructions 107 and data (e.g., secrets, counter value, etc.)

[0027] Button 106 can be any device that can send a signal when it is depressed or otherwise activated by a user. Examples of buttons include membrane-switches, pressure sensors, biometric readers, pressure switches and the like.

[0028] Clock 105 can be any device adapted to indicate the passage of time. An example of a clock includes a quartz crystal that emits a well-known frequency that can be used as the basis to change the value of a variable over time in a consistent way. In some embodiments of the present invention, token 101 does not include clock 105, but rather receives a timing signal from a source external to the token 101.

[0029] Display 104 can be any device capable of displaying information to a user, such as alphanumeric information. Examples of display 104 include a liquid crystal display, a TFT display, a plasma display, etc. The display 104 can show the OTP to the user.

[0030] The method in accordance with an embodiment of the present invention is shown in Figure 2. Value C is assigned an initial value and time T is set to zero, step 201. If an event E is determined to have occurred, step 202, then C is incremented by an amount B, step 203 and an OTP is computed, step 204. If no event E is determined to have occurred, then it is determined if an amount of time T has passed, step 205. If time T has passed, then C is incremented by an amount B, step 206 and T is reset to zero, step 207.

[0031] The foregoing description is meant to illustrate and not limit the scope of the present invention. One of skill in the art will appreciate that many variations beyond the foregoing examples are encompassed by the claims.

5

What is claimed is:

1. A method of generating a one time password, including:
incrementing a value TEC based upon the occurrence of an event and upon the passage of time; and
determining the one time password based upon TEC.

2. The method of claim 1, wherein the one time password is further determined based upon an adjunct value.

3. The method of claim 2, wherein the adjunct value is based upon a shared secret.

4. The method of claim 2, wherein the adjunct value is based upon a shared secret and public information.

5. The method of claim 1, where the event is the pressing of a button.

6. The method of claim 1, wherein the event is the selection of a dialog box.

7. The method of claim 1, wherein the event is the meeting of a condition.

8. The method of claim 1, wherein TEC is incremented after T seconds.

9. The method of claim 1, wherein TEC is incremented by TI after T seconds.

10. The method of claim 1, wherein TEC is incremented by EI after the occurrence of an event E.

11. The method of claim 1, wherein the one time password is computed in accordance with the algorithm specified in RFC 4226.

12. An apparatus for generating a one time password, including a processor constructed and arranged to increment TEC based upon the passage of time and the occurrence of an event and to compute a one time password based upon TEC.

13. The apparatus of claim 12, further comprising a memory and a display coupled to the processor.

14. The apparatus of claim 13, wherein the processor is further constructed and arranged to cause the display to show the one time password.

15. The apparatus of claim 12, wherein the processor is constructed and arranged to increment TEC by TI when the processor detects the passage of T seconds.

16. The apparatus of claim 12, wherein the processor is constructed and arranged to increment TEC by EI when the processor detects the occurrence of an event E.

17. The apparatus of claim 12, wherein the processor is constructed and arranged to compute the one time password in accordance with the algorithm specified in RFC 4226.

18. A medium storing instructions adapted to be executed by a processor to compute a one time password based upon a value that is incremented based upon the occurrence of an event and the passage of time.

19. The medium of claim 18, wherein the one time password is computed based upon an adjunct value.

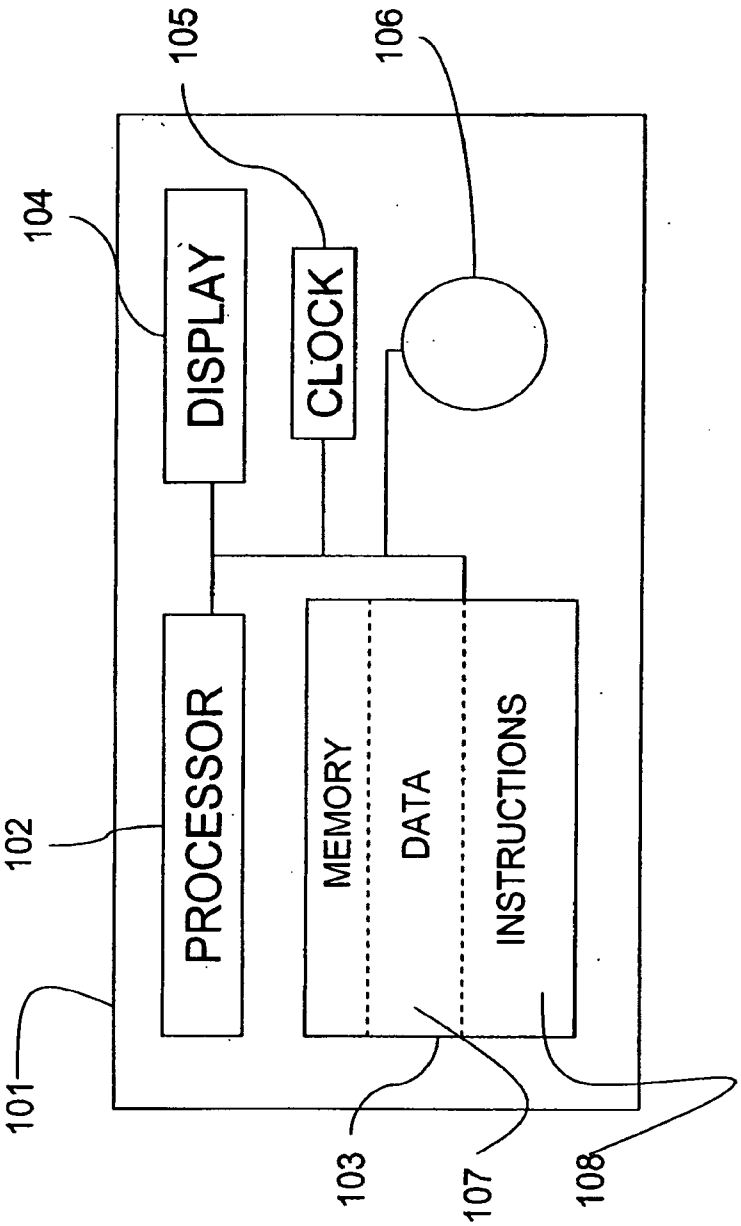

105
106
104
DISPLAY
CLOCK
PROCESSOR
MEMORY
DATA
INSTRUCTIONS
102
101
103
107
108

FIGURE 1

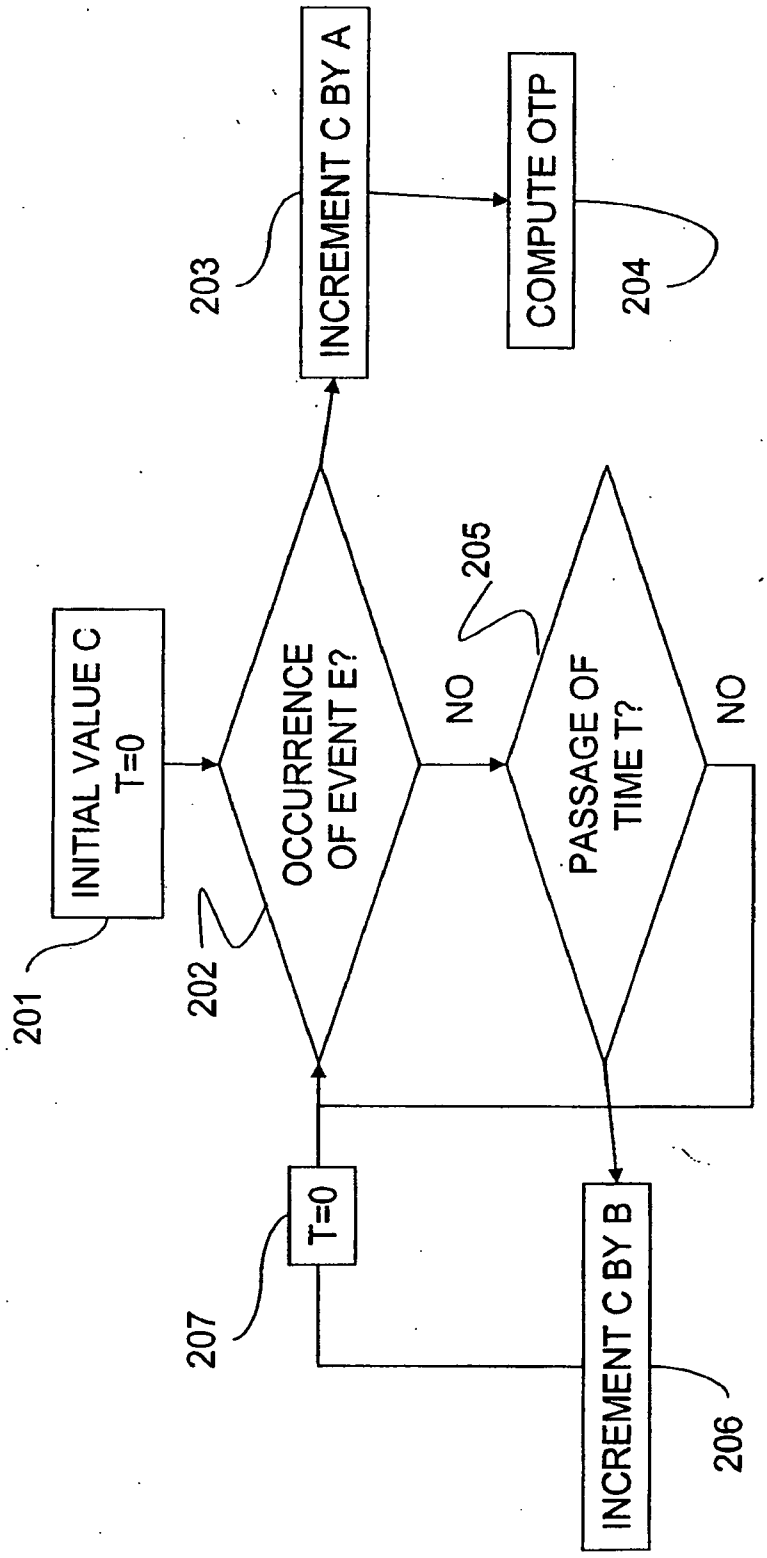

INITIAL VALUE C
T=0
201
OCCURRENCE OF EVENT E?
202
NO
205
PASSAGE OF TIME T?
NO
INCREMENT C BY A
203
COMPUTE OTP
204
T=0
207
INCREMENT C BY B
206

FIGURE 2