(19) DANMARK

(10) DK/EP 2158717 T3

(12) Oversættelse af europæisk patentskrift

Patent- og Varemærkestyrelsen

(51) Int.Cl.: *G 06 F 21/33 (2013.01)* *G 06 F 21/31 (2013.01)* *G 06 F 21/34 (2013.01)* *G 06 Q 20/38 (2012.01)* *H 04 L 9/00 (2006.01)* *H 04 L 9/32 (2006.01)* *H 04 L 29/06 (2006.01)*

(45) Oversættelsen bekendtgjort den: **2017-12-11**

(80) Dato for Den Europæiske Patentmyndigheds bekendtgørelse om meddelelse af patentet: **2017-09-06**

(86) Europæisk ansøgning nr.: **08827860.1**

(86) Europæisk indleveringsdag: **2008-05-30**

(87) Den europæiske ansøgnings publiceringsdag: **2010-03-03**

(86) International ansøgning nr.: **US2008065216**

(87) Internationalt publikationsnr.: **WO2009025905**

(30) Prioritet: **2007-05-31 US 756088**

(84) Designerede stater: **AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MT NL NO PL PT RO SE SI SK TR**

(73) Patenthaver: **Vasco Data Security International GMBH, , World-Wide Business Center, Balsberg, 8058 Zurich-flughafen, Schweiz**

(72) Opfinder: **COULIER, Frank, St-hubertusstraat 12, B-1850 Grimbergen, Belgien**
HOORNAERT, Frank, Koningin Astridilaan 164, B-1780 Wemmel, Belgien

(74) Fuldmægtig i Danmark: **RWS Group, Europa House, Chiltern Park, Chiltern Hill, Chalfont St Peter, Bucks SL9 9FG, Storbritannien**

(54) Benævnelse: **FJERNAUTENTIFIKATIONS- OG TRANSAKTIONSSIGNATUR**

(56) Fremdragne publikationer:
US-A- 5 937 068
US-A- 6 088 450
US-A1- 2002 087 860
US-A1- 2005 050 330
US-A1- 2006 136 748
US-B1- 6 308 266
US-B1- 6 715 078
MRAIHI VERISIGN M BELLARE UCSD F HOORNAERT VASCO D NACCACHE GEMPLUS O RANEN ALADDIN D: "HOTP: An HMAC-based One Time Password Algorithm; draft-mraihi-oath-hmac-otp-04.txt", 20041001, no. 4, 1 October 2004 (2004-10-01), XP015039457, ISSN: 0000-0004
WARD ET AL: "EMV card payments - An update", INFORMATION SECURITY TECHNICAL REPORT, ELSEVIER ADVANCED TECHNOLOGY, AMSTERDAM, NL, vol. 11, no. 2, 1 January 2006 (2006-01-01), pages 89-92, XP028033322, ISSN: 1363-4127, DOI: 10.1016/J.ISTR.2006.03.001 [retrieved on 2006-01-01]

Fortsættes ...

DESCRIPTION

Background

[0001] As remote access of computer systems and applications grows in popularity the number and variety of transactions which are accessed remotely over public networks such as the Internet has increased dramatically. This popularity has underlined a need for security in particular;

1. a. How to insure that people who are remotely accessing an application are who they claim they are and how to insure the transactions being conducted remotely are initiated by legitimate individuals. This subject is referred to as authentication.
2. b. How to insure that transaction data has not been altered before being received at an application server. This is referred to as data integrity.
3. c. How to guarantee that an individual, once having engaged in a transaction, is not in a position to repudiate it. This is referred to as non-repudiation.

[0002] In the past, application providers have relied on static passwords to provide the security for remote applications. In the last couple of years it has become evident that static passwords are not sufficient and that more advanced security technology is required.

PKI SMART CARDS

[0003] One way of solving the security problems associated with remote access to computer systems and applications over public networks is provided by a Public Key Infrastructure. In a Public Key Infrastructure one associates a public-private key pair with each user. The key pair is associated with a certificate (issued by a trusted Certificate Authority) that binds that public-private key pair to a specific user. By means of asymmetric cryptography this public-private key pair can be used to:

1. a. authenticate the user,
2. b. sign transactions, documents, e-mails (so as to prevent repudiation), and
3. c. set up encrypted communication channels.

[0004] To guarantee an adequate level of security it is mandatory that each user's private key remains secret and can only be accessed (e.g. to create a signature) by the legitimate user associated with that key. It is common to rely on a smart card to store the public-private key pair and the certificate and to carry out the cryptographic calculations involving the private key.

The use of the private key by the card is then often PIN-protected.

[0005] PKI-enabled smart cards are, and have been issued by:

1. a. Corporations to their employees or customers to secure the log-in to their computer networks or the remote access to their applications,
2. b. Banks to their customers to secure e.g. Internet banking applications, and
3. c. Governments to their citizens as electronic ID cards to create legally binding electronic signatures.

[0006] Apart from the advantages, there are also some disadvantages associated with PKI and the smart cards carrying the PKI keys and certificates:

1. a. Building a Public Key Infrastructure is generally quite complicated and therefore expensive when compared to competing security technologies.
2. b. PKI is inherently limited to environments and applications where there is a digital connection between clients and servers. In other words it is unsuitable for telephone banking or other delivery channels where it is not possible to provide a digital connection between the container of the PKI certificate and private key on the one hand and an application server on the other hand.
3. c. PKI smart cards do not have a power supply or a user interface. PKI smart cards therefore rely on the presence of an interfacing device that provides electrical power to the card, that is capable of digitally exchanging data with the card, and that is capable of interfacing with the user (e.g. capturing the card's PIN and presenting the data that should be signed). In most cases a PC with a connected transparent smart card reader is used. This reduces the mobility of the user (many PCs are not equipped with smart card readers). It also presents a security problem:

   all user interaction (such as approving a signature or capturing the card's PIN) is done on the inherently insecure PC.

Strong Authentication Tokens

[0007] An alternative technology for authentication and transaction signature capabilities is offered by what are called 'strong authentication token devices'. A typical example of strong authentication token is any one of the Digipass tokens offered by Vasco Data Security Inc., see the website Vasco.com.

[0008] A strong authentication token is a small autonomous battery-powered device with its own display and keyboard. In some cases the keyboard is reduced to a single button or even

completely omitted. The main purpose of a strong authentication token is to generate so-called 'One-Time Passwords' (OTPs). In some cases strong authentication tokens are also capable of generating electronic signatures or Message Authentication Codes (MACs) on data that has been entered on the token's keyboard. If the token has a keyboard, the usage of the token is often protected by a PIN.

To be able to generate OTPs or MACs, strong authentication tokens are capable of doing cryptographic calculations based on symmetric cryptographic algorithms parameterized with a secret value or key. Typical examples of such symmetric cryptographic algorithms parameterized with a secret value or key are symmetric encryption/decryption algorithms (such as 3DES or AES) and/or keyed one-way hash functions (such as MD5 or SHA-1 in OATH compliant tokens). In the remainder of the text the output of such algorithms will sometimes be referred to as *'symmetric cryptogram'*. The terminology *'symmetric cryptogram'* shall thus be understood as not only the output of a symmetric encryption algorithm but also of symmetric decryption algorithms or keyed hash functions. Strong authentication tokens are personalized with one or more secret keys that are supposed to be different for each individual token. To generate a one-time password or signature, the token typically performs the following steps (refer to Fig. 1):

1. a. Step 10: The token takes some input value (this could be a challenge generated by a server and typed-in on the keyboard by the user, and/or the value of the token's internal real-time clock, and/or the value of an internal counter managed by the token, and/or transaction data typed-in on the keyboard by the user).
2. b. Step 11: The token puts the input value into a specified format.
3. c. Step 12: The token then submits this formatted input to a symmetric encryption/decryption algorithm and/or one-way hash function parameterized by a personalized secret key 15 stored securely in the token. The result is a cryptogram or a hash value.
4. d. Step 13: The token transforms the cryptogram or hash value that is the outcome of this encryption/decryption or one-way hash into the actual OTP or MAC. i.e., the cryptogram or hash is typically truncated, converted in a human readable format (e.g. through decimalization) and visualized on the display. The user may submit this value to the application server.

[0009] In most cases a strong authentication token is a physical device, however in some cases the functionality of these strong authentication tokens to generate OTPs or MAC signatures is emulated by software running on a PC, a workstation, a mobile phone, a personal organizer, a PDA, etc. The latter are referred to as "soft tokens".

[0010] Once the OTP or MAC has been produced it is conveyed to an entity where the value can be verified as authenticating the user or the message, see fig. 2. Typically the entity is an application server. The application server stores data for each token, including which secret key(s) the token has been personalized with, and the identity of the user associated with the token. To validate a one-time password or signature, the server retrieves the secret key (115)

which is a copy of the key personalized in the token, takes the same inputs that were used by the token and in essence performs the same algorithm 112 as the token. The server then compares 120 the result it obtained with the value it received. (In practice, the validation of an OTP or MAC is often somewhat more convoluted if the strong authentication algorithm is time-based or counter-based, due to synchronization issues.) Since a one-time password or signature generated by a strong authentication token is a function of the token's individual secret key and the always different values of the input(s) to the token algorithm, validating the correctness of the one-time password or signature gives the application server a very high degree of confidence that the person submitting the one-time password or signature possesses the correct token and knows its PIN (if the token is PIN protected), which in turn gives a high degree of confidence that that person is indeed the legitimate user associated with that token device.

[0011] Because the OTP verification server and the OTP token in essence perform the same algorithm with the same key, the OTP generation algorithm can be a one-way or non-reversible function. That means that the actual OTP can be shorter than the cryptogram or hash value from which it is derived. This allows for OTP or MAC lengths that are sufficiently short so that it is not too inconvenient for users to manually copy the OTP or MAC values from the token display onto a PC. As a consequence strong authentication tokens don't require a digital connection between the token and the verification server.

[0012] The major advantages of strong authentication tokens when compared to PKI cards are:

1. a. They are fully autonomous (tokens have their own power supply and their own user interface);
2. b. They are independent of the delivery channel or communication medium (tokens don't require any digital or electronic connection with any other device; all input and output of data is done by the user via the token's display and keyboard); and
3. c. They offer a very high level of security (all user interaction such as capturing the PIN or providing transaction data to be signed is done via the token's own secure user interface).

[0013] In some cases where smart cards have been issued, one wants to get around the disadvantages and limitations associated with smart cards and achieve the same advantages that strong authentication tokens offer i.e. full autonomy, independence of the delivery channel, and a secure user interface.

[0014] One alternative is to combine the smart card with an unconnected, battery-powered smart card reader that has its own display and keyboard. The idea is that the combination of the smart card and the unconnected smart card reader emulates a strong authentication token. The functionality normally provided by a strong authentication token is then split over the smart card and the unconnected reader. The unconnected reader takes care of all user

interface, and all or a part of the other token functionality is delegated to the card.

[0015] Typically, all personalized secrets and security sensitive data are stored and managed by the card (e.g. the PIN is stored and verified by the card, the secret keys are stored on the card and all cryptographic operations involving those keys are done by the card, counters used as input for the token algorithm are stored and managed by the card). Part of the token functionality that is less sensitive (e.g. truncating and converting the generated hashes or cryptograms) often happens in the reader. An example of this combination is discussed below.

[0016] This principle is often used by banks that combine the bank cards they issue (for usage at Automatic Teller Machines or Point Of Sale terminals) with unconnected readers to secure their remote banking applications (such as internet banking or telephone banking). A good example of this is the Mastercard Chip Authentication Programme (CAP), which specifies how EMV smart cards can be used in combination with unconnected smart card readers to generate one-time passwords and electronic transaction data signatures.

[0017] This technology relies on the smart cards being capable of doing symmetric cryptographic operations and having been personalized with a secret key to be used for symmetric cryptographic operations. However, PKI-enabled smart cards are designed to store *asymmetric* keys and do *asymmetric* cryptographic operations. Many PKI-enabled smart cards don't support *symmetric* cryptographic operations or (if they do) have never been personalized with an individual symmetric secret key.

Traditional PKI signatures

[0018] The usual way to create an electronic signature with a PKI smart card, is that the input data (usually, the input data consist of a hash of the actual transaction data one wants to sign) are encrypted by the card's private key.

[0019] The usual way to validate such a signature, is that the validating entity decrypts the received signature with the public key. If the decryption of the signature results in the same value as the input data that were supposed to have been encrypted by the private key, the signature is validated successfully. Note that thanks to this asymmetric characteristic the validating entity never needs to have access to the card's private key. This allows the private key to be kept secret from any party other than the signing party, even from any verifying party, thus providing for true non-repudiation.

[0020] This can only be done successfully if the signature itself is in its *entirety* available to the validating entity. The decryption of an incomplete signature would only result in meaningless data that can not be compared with the input data that were supposed to have been signed.

[0021] This condition can not be fulfilled in practice when small hand-held unconnected smart card readers are being used: given that a typical PKI signature size is in the order of 100 bytes,

the display of these readers is far too small to display a full signature and it is in any case totally unrealistic to expect a user to manually transfer a 100-byte value from the reader's display to a PC without making a single mistake. The 100-byte typical PKI signature should be compared to a typical 6 to 8-digit or 3 to 4-byte OTP or MAC of a traditional strong authentication token. This is certainly a reason why asymmetric cryptography and private keys have not been used to generate OTPs and MACs by e.g. strong authentication tokens.

[0022] Patent application publication no. US 2005/050330 A1, entitled "Security token", discloses a security token, a security system and a method for authenticating a client. The security token includes a one-time password mechanism, for rendering one-time password functionality; a public-key mechanism, for rendering public-key functionality with respect to the one-time password functionality; and wired communication means with a host, for connecting the security token to the host and for providing the security token the power supply required for operating at least the public-key mechanism, thereby enabling rendering one-time password functionality and/or public-key functionality by the security token. This system relies on a digital communication link for transmitting the asymmetric cryptogram to the host for verification.

[0023] The informational internet draft "HOTP: An HMAC-based One Time Password Algorithm", draft-mraihi-oath-hmac-otp-04.txt (October 2004), describes an algorithm to generate one-time password values, based on HMAC [BCK1]. A security analysis of the algorithm is presented, and parameters related to the secure deployment of the algorithm are discussed.

[0024] The article by M. Ward, "EMV card payments - An update", Information Security Technical Report, vol. 11 (2006), no. 2, pages 89-92, presents a report on the status of EMV card payment technology and a look at where this payment technology is heading.

[0025] US patent no. 5,937,068, entitled "System and method for user authentication employing dynamic encryption variables", discloses a system including a first card-like unit adapted to communicate with a second unit giving only conditionally access to a function. Both units are capable of running software for generating a password by means of encryption of a plurality of dynamic variables produced separately but in concert (so as to have a predetermined relationship, such as identity, with one another) in the units. The encryption is carried out in each unit by a public algorithm using a dynamically varying encryption key. Each time an access request is issued by a card user, the key is modified as a function of the number of access requests previously formulated by the card user. Access to the function is granted when the passwords generated in the units have a predetermined relationship (such as identity) with each other. In a "virtual token" implementation, the first unit can be a smart card, which stores the dynamic key and the variable representing the number of formulated authentication requests and executes an encryption algorithm, a smart card reader and a computer such as a personal computer. Either the smart card reader or the personal computer can generate the time dependent variable. In a "software token" implementation, the functions of the first unit are performed by a personal computer, thus eliminating the need for a smart card or a smart card reader.

[0026] Patent application publication no. US 2006/136748 A1, entitled "Method and system for using a compact disk as a smart key device", discloses a data processing system adapted to accept a removable storage media, which becomes electrically engaged with a system unit within the data processing system, after which the removable storage media and the hardware security unit mutually authenticate themselves. The removable storage media stores a private key of a first asymmetric cryptographic key pair and a public key of a second asymmetric cryptographic key pair that is associated with the hardware security unit, and the hardware security unit stores a private key of the second asymmetric cryptographic key pair and a public key of the first asymmetric cryptographic key pair that is associated with the removable storage media. In response to successfully performing the mutual authentication operation between the removable storage media and the hardware security unit, the system unit is enabled to invoke cryptographic functions on the hardware security unit while the removable storage media remains engaged with the system unit. This document does not pertain to the field of strong authentication tokens as such, and relies wholly on asymmetric crtyptography. United States patent no. 6,088,450, entitled "Authentication system based on periodic challenge/response protocol", discloses a wireless authentication system to control an operating state of a node being a computer, door control mechanism or any multi-state product based on the proximity of an authorized user to the node. The wireless authentication system comprises a security device implemented within the computer and a user authentication token ("token") in possession of the authorized user. A Challenge/Response protocol is configured between the security device and the token. The first successful Challenge/Response message exchange between the security device and the token places the node in an operational state allowing the authorized user access to the contents and/or networked resources of the node. Later Challenge/Response message exchanges are set to occur periodically to check whether the authorized user possessing the token has left the node unattended thereby causing the node to be placed in a non-operational state. In that patent, the envisaged applications of the authentication system rely on the fact that the security device implemented within the computer interacts with a user authentication token via wireless communication. This security model is not compatible with the use of a smart card and a smart card reader.

[0027] What is desired is a method and apparatus that:

1. a) allows the usage of a device storing PKI private keys (such as PKI-enabled smart cards or USB sticks) to authenticate users and to sign transactions,
2. b) without the need for any user application to have some kind of a direct or indirect digital connection with the device containing the private key, in particular a digital connection that would allow the user application to submit data to the card for signing by the card's private key and that would allow retrieval of the entire resulting signature from the card should not be necessary,
3. c) without the need for the PKI-enabled device containing the private key (e.g. a PKI smart card or USB stick) to:
    1. 1) either support symmetric cryptographic operations, or
    2. 2) to have been personalized with some secret or confidential data element that can be read by a suitable reader.

Summary of the Invention

[0028] This application provides a description of a method and apparatus which meets the foregoing desire. In particular this application describes a number of embodiments which use the private key of a public-private key pair (a key which is meant to be used for asymmetric cryptography such as for example the RSA algorithm) to authenticate a user (via generation of a OTP) or to sign data (via generation of a MAC).

[0029] The embodiments described here differ from the traditional use of private keys to authenticate users and sign data (as described above) in that:

1. a) the same cryptographic key is used to generate and verify the OTPs and MACs; and
2. b) the length in bits of the OTP and MAC values can safely be considerably less than the length in bits of the cryptograms generated by the private keys.

[0030] All embodiments have in common that:

1. a) They all calculate a dynamic value using one or more variable inputs by means of a cryptographic algorithm that uses a secret that is also known or accessible to a verifying server.
2. b) These variable inputs can be any of:
    1. 1) Time value, or
    2. 2) Counter value, or
    3. 3) Challenge value, or
    4. 4) Transaction Data, or
    5. 5) Any combination of the above.
3. c) The dynamic value is then transformed into an OTP or MAC.
4. d) At some point in the course of developing the OTP or MAC an asymmetric cryptographic operation with a private key (i.e. an encryption/decryption or a signature) is carried out.
5. e) The transformation of the dynamic value into an OPT or MAC is such that the length or size of the OTP or MAC is smaller than the size of the cryptogram that was generated by the asymmetric cryptographic operation with the private key.

[0031] The precise role of the asymmetric cryptographic operation with the private key in the overall process of generating the OTP or MAC can be different from one embodiment to another.

[0032] In some embodiments the asymmetric cryptographic operation with the private key is performed each time an OTP or MAC has to be generated. In other embodiments more than one OTP or MAC can be generated in connection with a single asymmetric cryptographic operation with the private key. In the latter case, criteria that can determine whether or not a new asymmetric cryptographic operation with the private key is required when a new OTP or MAC needs to be generated can include:

1. a) The time that has passed since the last asymmetric cryptographic operation.
2. b) The number of OTPs and/or MACs that have already been generated.
3. c) Whether or not a communication session between a device containing the private key and a device capturing the inputs and making available the OTPs has been uninterrupted (e.g. whether a PKI smart card has not been removed from a smart card reader).
4. d) The type of OTP or MAC. For example the generation of a MAC might always require a new asymmetric cryptographic operation but the generation of an OTP would not.

[0033] In a typical embodiment only one private key is used and only one asymmetric cryptographic operation is performed with that private key. However, some embodiments may perform a number of cryptographic operations with either a single private key or with a number of private keys. Examples:

1. a) If the OTP is a function of the encryption result of the variable inputs by a private key, then a variant could be that the OTP is a function of more than one cryptogram, or that the variable inputs are encrypted by more than one private key to generate the OTP.
2. b) If the generation of an OTP only takes place after the presence of a specific smart card is verified by checking the result of an encryption of a challenge by the card's private key, then a variant could be that more than one challenge is submitted to the card to be encrypted by the card's private key.
3. c) In many cases a PKI card contains a so-called utility private key and a signature private key. In that case the utility key might be used if an OTP is generated and the signature key might be used if a MAC is generated.

[0034] In a preferred embodiment both OTPs to authenticate a user and MACs to sign data can be generated. However alternative embodiments can be limited to only being capable of generating OTPs or only being capable of generating MAC signatures.

[0035] In a typical embodiment the asymmetric cryptographic algorithm used with the private key will be the RSA algorithm. However, other embodiments can use other asymmetric algorithms provided they are capable of either encryption or decryption or signing functionality by using the private key. Examples of such algorithms include: RSA, knapsack algorithms such as Merkle-Hellman or Chor-Rivest, Pohlig-Hellman, Diffie-Hellman, ElGamal, Schnorr, Rabin,

Elliptic Curve cryplosystems, Finite Automaton public key cryptosytems, the Digital Signature Algorithm (DSA, DSS).

[0036] In a typical embodiment the component that contains the private key and the component that generates the OTP and MAC values are two different components, each being a part of two different devices. However, embodiments can easily be conceived in which these two components are parts of the same device or are even the same component.

[0037] In atypical embodiment the private key is stored on a smart card. In a preferred embodiment the cryptographic calculations involving the private key are performed by that smart card. In a typical embodiment the OTP and/or MAC values are generated by a device that is equipped with or connected to a component or device that can communicate with the smart card containing the private key.

[0038] In a preferred embodiment the card reading device is an unconnected smart card reader with its own power supply and running the appropriate software to communicate with a PKI smart card which has been inserted into the smart card reader to generate OTPs or MACs.

[0039] In another embodiment the card reading device is the combination of some computing device such as a PC, PDA, cell phone, etc., equipped with a smart card reader and running the appropriate software to generate OTPs or MACs.

[0040] In a typical embodiment the physical, electrical and protocol aspects of the communication between the smart card and the smart card reading device is the same or similar to those described in the ISO 7816 standard. Other embodiments could use other communication means such as a contactless smart cards as described in ISO 14443.

[0041] Alternative form factors are available for the private key containing device, as well as alternative form factors for the OTP or MAC generating device, and alternative means for the communication between the private key containing component or device on the one hand and the OTP and MAC generating component or device on the other hand. These alternatives are within the scope of the invention as described herein.

[0042] In one embodiment the OTPs or MACs values are visualized on a display of the card reading device. An OTP can e.g. consist of a series of symbols. In a typical embodiment these symbols are decimal digits. In other embodiments these symbols can for example include:

1. a) hexadecimal digits, or
2. b) base 64 digits, or
3. c) characters from a writing system such as an alphabet, or
4. d) pictograms.

[0043] In one embodiment the generated OTPs or MACs are communicated to the user by

means of audible signals. For example the OTP can be a string of digits or characters or words that each have a characteristic associated tone or that are read by a text-to-speech converter.

[0044] In one embodiment the generated OTPs or MACs are directly communicated to an application by some electronic wired or wireless communication mechanism. This mechanism can include a USB connection or an infrared connection or a Near Field Communication connection or an RF connection or a Bluetooth connection.

[0045] Other output mechanisms for the OTPs or MACs can be provided. In some embodiments the private key-based function is PIN protected.

[0046] The following description describes the basic embodiments in more detail. In some embodiments the card's private key-based function is directly or indirectly used in the OTP or MAC generation. Either

1. a. an asymmetric cryptographic operation involving the card's private key is an integral phase or part of the transformation of the variable inputs into an OTP or MAC (Using the asymmetric algorithm in a symmetric way), or
2. b. the card's private key-based function is used more indirectly to provide a seed value that is used to derive a secret symmetric key that is used by the OTP or MAC generation algorithm. (Using an asymmetric cryptogram as a seed to derive a secret key).

[0047] In some of the embodiments the value of the OTPs and/or MACs is a function of the actual value of the card's private key. In yet other embodiments the card's private key-based function is used to unlock the OTP or MAC generation algorithm in the reader:

1. a. Either the card is linked to an already personalized reader and recognized on the basis of stored challenge-response pair(s), or
2. b. the card is authenticated by the reader through traditional PKI certificate based verification.

[0048] In the embodiments described in the immediately preceding paragraph the value of the generated OTPs and/or MACs is not a function of the actual value of the card's private key.

[0049] Thus in one aspect the invention provides a method to generate a security value comprising a One-Time Password (OTP) or a Message Authentication Code signature (MAC) comprising:

obtaining an intermediate dynamic value created using one or more variable inputs and a cryptographic algorithm employing at least one secret;

transforming said dynamic value into said security value,

wherein an asymmetric cryptographic operation with a private key is carried out producing a cryptogram, in order to transform said dynamic value, and

said transforming includes producing said security value of a size which is smaller than the size of a cryptogram that was generated by said asymmetric cryptographic operation.

[0050] In another aspect the invention provides a device generating a security value comprising a One-Time Password (OTP) or a Message Authentication Code signature (MAC) using the method described immediately above.

[0051] In another aspect the invention provides a method of validating a security value provided by a user in order to authenticate the user or data associated with the user, said security value comprising a One Time Password or a signature comprising a Message Authentication Code; said method comprising:

creating a reference cryptogram using a reference cryptographic algorithm applied to one or more reference inputs using a server key related to a PKI private key of an authentic user, the reference cryptographic algorithm and the one or more reference inputs selected as identical to corresponding elements used in creating the security value by the authentic user;

thereafter either

operating on said reference cryptogram alone by transforming said reference cryptogram into a reference security value including producing said reference security value of a size which is smaller than the size of the reference cryptogram and effecting a comparison of said reference security value and said security value, or

operating on both said reference cryptogram and said security value to produce a modified reference cryptogram and a modified security value, said operation on said reference cryptogram identical, in part to an operation carried out to create said security value, and effecting a comparison of said modified reference cryptogram and said modified security value, and

determining validity of said security value from results of said comparison.

[0052] In still another aspect the invention comprises a computer readable medium supporting a sequence of instructions which, when executed perform a method of generating a security value comprising a One-Time Password (OTP) or a Message Authentication Code signature (MAC), said method comprising:

obtaining an intermediate dynamic value created using one or more variable inputs and a cryptographic algorithm employing at least one secret;

transforming said dynamic value into said security value,

wherein an asymmetric cryptographic operation with a private key is carried out producing a cryptogram, in order to transform said dynamic value, and said transforming includes producing said security value of a size which is smaller than the size of a cryptogram that was generated by said asymmetric cryptographic operation.

[0053] Finally in still another aspect the invention comprises an information bearing signal comprising a sequence of instructions which, when executed in a processor perform a method of generating a security value comprising a One-Time Password (OTP) or a Message Authentication Code signature (MAC), said method comprising:

obtaining an intermediate dynamic valuecreated using one or more variable inputs and a cryptographic algorithm employing at least one secret;

transforming said dynamic value into said security value,

wherein an asymmetric cryptographic operation with private key is carried out producing a cryptogram, in order to transform said dynamic value, and said transforming includes producing said security value of a size which is smaller than the size of a cryptogram that was generated by said asymmetric cryptographic operation.

Brief description of the drawings

[0054] Several embodiments of the invention are now further described in then following portions of the specification when taken in conjunction with the attached drawings in which:

Fig. 1 is a flow diagram of the operation of a prior art strong authentication token in generating an OTP of MAC;

Fig. 2 is a flow diagram of the operation of a prior art server in authenticating an OTP or MAC generated by a strong authentication token and its relation to the OTP or MAC generation;

Fig. 3 is a flow diagram of an embodiment of the invention relying on an asymmetric cryptographic operation using a PKI private key to create a cryptogram from which an OTP or MAC is generated;

Fig. 4 is a flow diagram of an embodiment of the invention showing OTP/MAC generation at the client (as in fig. 3, for example) and the related authentication at a server;

Fig. 5 is the flow diagram of another embodiment of the invention which uses an asymmetric cryptogram as a seed to derive a key which is used in creating a cryptogram representing an

OTP or MAC;

Figs 6 and 7 are flow diagrams of still another embodiment of the invention in which the smart card is used to authenticate the user to the reader, which in turn produces a cryptogram from which an OTP or MAC is derived, in this embodiment the user's smart card is bound to the reader in an initial operation (fig. 6) and operation thereafter is represented in fig. 7;

Fig 8 and 9 are flow diagrams of still another embodiment of the invention in which the smart card, including a PKI certificate, is used to authenticate the user to the reader, which in turn produces a cryptogram from which an OTP or MAC is derived, in this embodiment a random user may be authenticated;

Figs 10 and 11 illustrate actions taken in an initial session to capture information allowing operation of various embodiments of the invention;

Fig. 12 illustrates the context in which embodiments of the invention operate;

Fig. 13 is an illustration of a first validation procedure, and

Fig. 14 is an illustration of another validation procedure.

Detailed Description

[0055] Important components of embodiments of the invention are illustrated in Fig. 12 as including a smart card reader 20 (or simply reader) and an authentication server 30 (or simply server).

[0056] At a minimum the reader 20 includes an interface 28 to accept a smart card and a power supply 27. Some readers also include one or more user operable buttons or keys; this is represented in fig. 12 by the keyboard 25. As used herein a user inserts a smart card into the smart card interface 28. As a consequence of some operation carried out by the reader 20, information is generated by the reader. That information may be a One-Time Password (OTP). If transaction data is input to the reader the information which is generated may include a signature such as a MAC. The output information may be presented on a display, such as the display 26. Alternatively the reader may be digitally connected to a network. In that event the information may be presented to another entity also connected to the network and the display 26 may be unnecessary. Typically the information which is generated by the reader 20 is used to authenticate a person or a message. A person may be authenticated by use of a smart card (proving possession of the card) and some other information (such as a PIN or other user data). The reader accepts the smart card and other information and creates an OTP. The OTP is communicated to server 30. Alternatively the message is signed by the reader 20, producing a MAC and the MAC is communicated to server 30.

[0057] Server 30 is typically implemented as a computer with processing capability and a data base 35. The information generated by the reader is communicated to the server 30 via the data path 40. Data path 40 may take various forms. Typically the user manually transfers information from the display 26 to a client device that is connected to the server 30. Alternatively data path 40 may comprise a digital path allowing information to be communicated from reader 20 to server 30. As another alternative the data path may carry audio information, such as a telephone circuit which carries the voice of a user enunciating information presented to the user on the display 26; where the information may be an OTP or MAC. Data path 40 may carry optical signals representing the information generated at reader 20. In general data path 40 is any path which can be used to communicate information from the reader 20 to the server 30. The server 30 accepts either the OTP or MAC and with the assistance of data in the data base 35 determines whether to accept or reject the information as validating the identity of the user (OTP) or the authenticity of the message (MAC). The particular procedures and data which are used by the server 30 are more particularly described below. One output of the server 30 selects either the accept or reject for status 36, reflecting either acceptance of the OTP as validating the authenticity of the user's claim of identity or the validation of the MAC as authenticating the associated message.

Using the asymmetric algorithm in a symmetric way

[0058] In this embodiment (see fig. 3) a smart card 100 cooperates with a smart card reader 105. Smart card 100 stores a PKI private key 301 which is used in an asymmetric cryptographic operation. The card's private key-based function (i.e. an asymmetric cryptographic operation involving the card's private key such as signing or decrypting) is an integral phase or part of the process which produces the OTP or MAC.

[0059] Generation of the OTPs and/or MACs happens in the following way:

Step 99: Input values which will be used in later steps are captured.

Step 101: the input(s) for the OTP or MAC generation algorithm are transformed or formatted into an initial value.

Step 102: the initial value is signed or encrypted/decrypted by the card's private key 301.

Step 103: the resulting cryptogram is transformed into an OTP or MAC.

[0060] In the example of Figure 3 the OTP or MAC is a function only of the result of the asymmetric cryptographic operation. However, in other embodiments the OTP or MAC may also be function of other data elements including values that are functions of the variable inputs but that are not functions of the private key 301.

[0061] In a typical embodiment the input(s) to the OTP or MAC generation algorithm are the same or similar as the inputs for the strong authentication algorithm(s) used in traditional strong authentication tokens. In other words these inputs may be selected as a:

time value, or

challenge (typically provided by a server), or

counter value, or

transaction data, or

any combination of the above.

[0062] In some embodiments additional input(s) or parameter(s) to the OTP/MAC generation algorithm can include:

data identifying a device (e.g. a reader serial number). or

secrets stored in the device, or

user identification data, or

secret codes or secret values provided by the user.

[0063] Formatting these input(s) into the initial value, step 101 can include operations such as:

Concatenation, or

Hashing. or

encryption/decryption with a symmetric cryptographic algorithm (e.g. using a secret key stored in the device or provided by the user).

[0064] Transforming the resulting cryptogram into the final OTP or MAC value, step 103 can include the following operations:

hashing (possibly a keyed hashing using a secret key stored in the reader 105 or provided by the user), or

encryption/decryption with a symmetric cryptographic algorithm (e.g. using a secret key stored in the reader 105 or provided by the user), or

truncation, or

selection of certain bits, nibbles or bytes, or

decimalization. The latter may be accomplished by:

interpreting the string of bits to be decimalized as a large binary representation of a number, or

dividing the string of bits to be decimalized in groups of bits and mapping each group of bits onto a decimal digit. A typical example is dividing the string of bits into nibbles and mapping each nibble onto a decimal digit according to the following rule. If the hexadecimal value of the nibble is 0x0 to 0x9, take the decimal digit with the same value;

if the hexadecimal value of the nibble is 0xA to 0xF, subtract a constant (between 0x6 and 0xA) and then take the decimal digit with the same value as the result of the subtraction, or

many other decimalization algorithms known to those skilled in the art.

[0065] The validation phase is now described. In this embodiment the validating server has a copy of the private key 301 that was used to generate the OTP or MAC value and uses it to perform essentially the same algorithm as the algorithm to generate the OTP or MAC value. The validating server:

(refer to Fig. 4) somehow obtains or reconstructs or guesses the value(s) of the data elements that were used as input(s) to the OTP or MAC generation algorithm when the OTP or MA.C was generated:

in case of a time value, the validating server may have its own clock that is synchronized with the clock used for generating the OTP or MAC,

in case of a challenge, the challenge may have been generated by the validating server itself or may have been passed to the validating server by the application together with the received OTP or MAC,

in case of a counter, the validating server may maintain its own counter value synchronized with the counter value used for generating the OTP or MAC,

in case of transaction data, these data may have been passed to the validating server by the application together with the received OTP or MAC;

the input(s) for the OTP or MAC generation algorithm are transformed into an initial value.

The initial value is thereafter signed or encrypted/decrypted (402) using the copy of the private key 301 held by the validation server. The validating server then compares (403) the resulting reference cryptogram with the OTP or MAC value that was received. If the resulting reference cryptogram matches the OTP or MAC value that was received, the signature is validated successfully. This comparison might be done in a number of ways:

the validation server might in some embodiments transform the reference cryptogram into a reference OTP or MAC value and compare the reference OTP or MAC value with the received OTP or MAC value (e.g. by checking whether they are identical), or

the validation server might reconstruct, from the received OTP or MAC value a part of the original cryptogram generated by the private key, and compare this partial cryptogram with the corresponding part(s) of the reference cryptogram,

or

the validation server might transform the reference cryptogram into a first intermediate validation value, and transform the received OTP or MAC into a second intermediate validation value, and compare the first and second intermediate validation values.

[0066] This can be illustrated by the following example (see Fig. 14). In this example the OTP or MAC is produced based on a cryptogram which is the result of an asymmetric encryption using a private key 1308. The server produces a reference cryptogram which is also the result of an asymmetric encryption using a key 1324 which is a copy of the private key 1308. As shown in fig. 14

- the reader 1350 calculates the OTP or MAC from said original cryptogram by:
  - selecting every first bit of every byte of said resulting cryptogram (1355), and
  - concatenating said selected bits into a bit string (1356), and
  - interpreting said bit string as the binary interpretation of a number and obtaining the OTP or MAC by taking the decimal representation of said number (1357)
- the validation server validates this OTP or MAC as follows:
  - the server modifies the reference cryptogram by setting all bits except every first bit of every byte to 1 (1364), and
  - the server interprets the received OTP or MAC as the decimal representation of a number and obtains a bit string by taking the binary representation of that number (1359), and
  - the server expands said bit string by replacing every bit of said bit string by a byte that consists of the bit being expanded appended with seven 1-bits (1360), and,
  - the server compares said expanded bit string with said modified reference cryptogram (1365).

[0067] The parameters of this procedure (choosing one bit of every byte) is illustrative. Those skilled in the art will be able to select an appropriate parameter to suit their needs and context. In particular, a typical RSA cryptogram is about 100 bytes. Selecting one bit of each byte will produce 100 bits. At about 3 bits per decimal digit this will produce about 30 decimal digits for the OTP or MAC which is more practical than 300 decimal digits, but may still be considered awkward. In that event we can select one bit of every 40 bits for a total of 20 bits or about 6 decimal digits. The same procedure for generating the OTP or MAC from a cryptogram (transforming by selecting some but not all bits of the cryptogram) can also be used in the event a symmetric key is used in lieu of the asymmetric key. A typical symmetric cryptogram includes about 100 bits. In this case selecting one of every eight bits will leave us with about 12 bits or 4 decimal digits. This may be considered too small a number to be safe from attack. To avoid this problem we merely use one of every 4 bits (instead of 1 of every 8) to leave us with about 25 bits or about 8 decimal digits.

[0068] An alternative validation procedure is illustrated in fig. 13. The procedures of fig. 13 are the same as the procedures of fig. 14 in producing the cryptogram on the client side (operation 1305) and the reference cryptogram on the server side (operation 1323). As shown in fig. 13:

the cryptogram is transformed into the OTP or MAC by a sequence of two transformations, first a transform A (1306) and then a transform B (1307)

the validation server subjects the reference cryptogram to an operation 1325 to produce a modified reference cryptogram, operation 1325 is identical to the operation of transform A,

the validation server also subjects the OTP or MAC to an operation (1327) which is the inverse of transform B to produce a modified OTP or MAC,

validation depends on a comparison (1328) of the modified OTP or MAC with the modified reference cryptogram.

[0069] As was the case for the validation procedure of fig. 14, the technique of fig. 13 can be used regardless of whether the cryptogram is produced with a symmetric or asymmetric key.

[0070] In contrast to traditional PKI signature verification, the method of fig. 3 doesn't require the full signature to be available to the server (as was demonstrated in connection with either fig. 13 or 14). The solution can offer a very high level of security, even if no additional secret codes or keys (provided by the user or stored in the device) are being used other than the private key.

[0071] However, the technique of fig. 3 can only be used if the validating server has a copy of the card's private key when it has to validate an OTP or MAC. The whole point of PKI is exactly that, in order to guarantee true non-repudiation, the private key is never accessible to anyone other than the user associated with that key. In many cases this is guaranteed by the card generating the private and public key pair on-board without any possibility of extracting the

private key from the card. In other cases the key pair is generated externally and then injected into the card, but then procedures would normally ensure that the private key in the card personalization system is immediately destroyed after injection into the card and no copy of the private key is allowed to exist outside the card. In other words, this method will in many cases not be a suitable solution.

Using an asymmetric cryptogram as a seed to derive a secret key (Fig. 5)

[0072] In the following embodiment, the requirement that the validation server has access to a copy of the private key at the time of validation is eliminated. In this embodiment an OTP/MAC is generated in the same way as a traditional strong authentication token. All the steps of this algorithm (capturing the inputs, formatting the inputs, encrypting or hashing the formatted inputs, transforming the resulting cryptogram of hash into an OTP/MAC) are performed by the reader 505. In this embodiment the invention differs from conventional practice in how the reader 505 obtains the symmetric secret strong authentication key. To obtain this secret symmetric authentication key, the reader 505 relies on an operation of the card 500 involving the card's private key 520.
The main steps of a basic embodiment of this method are as follows:

1. 1. If required (i.e. the.card protects usage of the private key by a PIN) the reader asks the user to enter the PIN and submits that PIN to the card.
2. 2. Assuming the card 500 accepts the PIN, the unconnected card reader submits a fixed value to the card to be signed by the private key. This fixed value is further referred to as the 'reader-to-card challenge'.
3. 3. The card signs the given challenge with its private key and returns the resulting cryptogram to the reader. This resulting cryptogram is further referred to as the 'card-to-reader signature response'.
4. 4. The reader uses the resulting cryptogram as a seed to derive a symmetric secret key. This key is further referred to as the 'derived strong authentication secret key'.

[0073] The reader dynamically personalizes the strong authentication algorithm (that is entirely carried out by the reader) with that derived strong authentication secret key. In other words the reader carries out the strong authentication token algorithm using the derived strong authentication secret key.

[0074] Fig. 5 illustrates a suitable embodiment showing the interaction of reader 505 and card 500. The process may require the user to enter a PIN 510 in order to unlock the card 500. This step is optional, but if performed, the PIN entered at 510 by the user is communicated 511 to the card 500 to be tested. The card either accepts or rejects the PIN. The response of the card 500 is tested, 512 and only if accepted does the process continue. Thereafter function 513 captures input values from some or all of the reader, the user or the card. Function 514 may

format some or all of the input values. Some or all of these values, or others, may form a reader-to-card challenge 515a which is sent (function 515) to the card 500. The card 500 uses the challenge 515a by performing a cryptographic operation with the card's private key 520. The resulting cryptogram, the card to reader signature response 516a, is communicated back to the reader, function 516. The response 516a is then used as a seed to create a secret value or key 517a via function 517. Key 517a is termed a derived secret strong authentication key. The key 517a is then used in a cryptographic operation, at function 518 along with the formatted value provided by function 514. Finally the resulting cryptogram is transformed at function 519 to produce the OTP or MAC.

[0075] The 'reader-to-card challenge' 515a could be any of the following:

1. 1. A fixed value that is the same for all readers of a certain batch.
2. 2. A fixed value that is fixed for a given reader but that has a different value for each reader.
3. 3. A fixed value that is constant for a given user but that can be different for different users and thatis entered at least once in the reader by the user. In practice it is very likely that this value will be entered either every time the card is used, or only the first time that a given card is used with a certain reader and will then be remembered by the reader.
4. 4. Static data stored on the card that can be read by the reader (e.g. the public key and certificate, or a card serial number).
5. 5. A combination of any of the above.
6. 6. A value derived from any of the above. The derivation optionally including the use of some reader secret.

[0076] The algorithm to derive the strong authentication secret key from the 'card-to-reader signature response' could make use of the following techniques (among others):

1. 1. Extracting bits of some data elements
2. 2. Concatenating some parts of some data elements
3. 3. Symmetric encryption/decryption algorithms (e.g. DES, AES, ...)
4. 4. Hashing algorithms (e.g. SHA-1)

[0077] The algorithm to derive the strong authentication secret key 517a from the `card-to-reader signature response' 516a could make use of the following extra data elements besides the 'card-to-reader signature response' 516a:

1. 1. A fixed value that is the same for all readers of a certain batch.
2. 2. A fixed value that is fixed for a given reader but that has a different value for each reader.
3. 3. A fixed value that is constant for a given user but that can be different for different

users and that is entered at least once in the reader by the user.

4. 4. Static data stored on the card that can be read by the reader (e.g. data associated with the private key such as the public key and certificate, or a card serial number).
5. 5. A combination of any of the above.

[0078] This description only mentions the use of a single private key of a smart card and a single operation with that key; if the card contains more than one private key the reader could submit the 'reader-to-card challenge' 515a to each of these card private keys and combine the resulting 'card-to-reader signature responses' 516a in the derivation of the 'derived strong authentication secret key' 517a.

[0079] Similarly the reader could also submit different 'reader-to-card challenge' values 515a to the card and combine the resulting 'card-to-reader signature responses' 516a in the derivation of the 'derived strong authentication secret key' 517a.

[0080] In yet another embodiment the reader does not rely on a single 'reader-to-card challenge' 515a and corresponding `card-to-reader signature response' 516a and 'derived strong authentication secret key' 517a, but instead uses a set of `reader-to-card challenges'515a and corresponding 'card-to-reader signature responses' 516a and 'derived strong authentication secret keys' 517a. To obtain a 'derived strong authentication secret key' 577a the reader selects one of these 'reader-to-card 515a challenges' and submits it to the card. Which 'reader-to-card challenge' 515a is selected determines the corresponding 'card-to-reader signature response' 516a and 'derived strong authentication secret key' 517a. This selection therefore must happen in a way that is predictable to the validation server. The reader can e.g. cycle through the set of 'reader-to-card challenges' 515a in a fixed order or can select a 'reader-to-card challenges' 515a depending on the value of the inputs) to the strong authentication token algorithm. A simple example of the latter method is that the strong authentication token algorithm works in challenge-response mode and that one specific digit (e.g. the last digit) of the challenge indicates the index of the 'reader-to-card challenge' to be used.

[0081] Because the private key is different for each card, the derived secret key will for a given challenge be specific to a given card. In other words, the secret key that is used in the strong authentication algorithm in the reader is function of the card (or more precisely: of the private key 520 in that card). That means that in principle one needs to have access to the correct card to be able to generate a correct OTP.

[0082] In most cases the private key is PIN protected, so that in addition to having access to the correct card, one also needs to know the card's PIN to be able to generate a correct OTP.

[0083] If the fixed value which the reader submits to the card to be signed by the private key can be different for different readers, then one needs besides the other elements (e.g. access

to the correct card and knowledge of the card's PIN) also the correct reader. Note: such usage of a value that is different for different readers, effectively 'binds' the reader to the card.

[0084] For the validation server to be able to validate the strong authentication OTPs and/or MACs generated in this way, it must know the value of the derived strong authentication secret key 517a. The server must therefore know the card's signature response 516a. The card signature response for a given card challenge is determined by the card's private key 520 and can not be calculated without access to the private key 520. One consequence of this is that the server must have access to the card's private key 520 (directly or indirectly) at least once.

[0085] If the key pair is generated internally on the card this means that the server needs access to the card at least once, so that the server can submit to the card the card challenge(s) that will be applicable for this user and retrieve and store the card response(s) to that challenges) (indirect access to the private key). If the key pair is generated externally and then injected in the card, the server could use the private key directly to encrypt the challenge(s) before the private key outside the card is destroyed.

[0086] Only then is the server able to calculate the corresponding derives strong authentication key from the encrypted card challenge. The disadvantage of this is that, in practice, either the user will have to grant the server access to his/her card during a sort of registration phase, or (in case of external key generation) the server must be allowed to encrypt the challenge with the private key value before that private key value is destroyed.

[0087] Another consequence is that in practice for a certain user, the derived strong authentication secret key must remain unchanged. Since the derived strong authentication secret key is derived from the card's signature response to a certain card challenge, that card challenge and the corresponding 'card-to-reader signature response' must remain fixed for a given user. The disadvantage of this is that, if an attacker obtains the value of the 'card-to-reader signature response' of a certain user, then that attacker could potentially make fake cards that always return that recorded 'card-to-reader signature response' value when inserted in a reader.

[0088] Including reader specific or user specific data elements in the generation of the 'reader-to-card challenge' and/or the derivation of the 'derived strong authentication secret key' from the 'card-to-reader signature response' can make it harder for an attacker to obtain the value of the correct 'card-to-reader signature response' or to exploit that value with a reader to generate in a fraudulent way correct OTPs or MACs.

[0089] Another way to make it harder for an attacker to obtain the correct 'card-to-reader signature response' is to hot rely on a single 'reader-to-card challenge' and corresponding 'card-to-reader signature response' and 'derived strong authentication secret key', but instead use a set of 'reader-to-card challenges' and corresponding 'card-to-reader signature responses' and 'derived strong authentication secret keys' as explained above.

[0090] In the following embodiment, the requirement for the server to have access at least once to the card to perform a private key operation is eliminated altogether.

[0091] In this embodiment, the value of the symmetric secret authentication key is not dependent (directly or indirectly) on the value of the card's private key. The symmetric secret authentication key is not derived from a seed that is generated by the card by means of an asymmetric cryptographic operation involving the card's private key. Instead the reader is personalized with the symmetric secret authentication key or with secret data from which the reader can dynamically derive the symmetric secret authentication key. With this symmetric secret authentication key the reader can generate OTPs or MACs just like a traditional strong authentication token. Usage of the reader is protected and reserved to the legitimate user by logically binding the user's card to the reader. Once the user's card has been bound to the reader, the reader will only generate an OTP or MAC if the user inserts the card that was bound to the reader. The card thus functions as an access key to unlock the personalized reader.

[0092] At first usage, the reader will request the user's card to be inserted. Upon insertion of the card, the reader binds itself logically to the inserted card in the following way. The reader determines and remembers some specific individual characteristics of that card. These characteristics can include:

◦ card serial number

◦ card's public key and/or certificate

◦ the card's response to a given challenge (where the response is defined as the encryption of the challenge by the card's private key. Note: this would typically require the user to submit the PIN to unlock the private key). This challenge and the corresponding card's response must be remembered by the reader. The challenge can be:

▪ a fixed over-all challenge (same for all cards and all readers)

▪ fixed challenge per reader

▪ fixed challenge per card (e.g. randomly generated by the reader upon first presentation of the card and then remembered by the reader)

▪ challenge provided by user

▪ a combination of any of the above

[0093] An example of this operation is illustrated in fig. 6. The reader 600 awaits receipt of card data (function 616). The card provides some card data 611 to the reader (function 610). When the reader receives the card data 611, that data is stored (function 617).

[0094] If the user wants to generate a dynamic password or signature (see fig. 7), the reader asks for the card that was bound to that reader. The reader checks whether the presented card is indeed the expected card. I.e. it will retrieve the characteristics of the presented card (function 710) and compare them with the stored characteristics of the card bound to the reader (function 711). This step can include:

- reading the card's serial number
- reading the card's public key and/or certificate
- submitting a (stored) challenge to the card for encryption by the card's private key (which may require the user to provide the PIN to unlock the private key) and receiving the card's response.

[0095] Upon successful validation of the presented card, the reader proceeds with performing the strong authentication algorithm as an ordinary strong authentication token.

[0096] To strengthen the security, many variations are possible. The reader can derive the symmetric secret authentication key from:

- a data element pre-personalized in the reader,
- and/or a data element provided to the reader by the user,
- and/or a data element that the reader reads from the card.

[0097] Preferably, these data elements are secret Instead of using always the same challenge and corresponding card response that was used and obtained when the card was bound to the reader, the reader can use multiple pairs of challenges and corresponding responses. Variations on this principle include:

- When the card is bound to the reader, the reader generates and submits more than one challenge to the card and remembers the corresponding card responses. When the reader later on needs to validate the card, it can submit any subset of these challenges to the card and check whether the card's responses match the stored responses.
- When the reader has successfully validated the inserted card, it can generate a new challenge and obtain a corresponding response from the card. This new challenge-response pair can then be remembered by the reader as an alternative or additional pair to the already previously known challenge-response pair(s).
- These two variations can be combined.

[0098] The principle of yet another embodiment (Figs 8 and 9) is as follows. On behalf of the server, the reader locally authenticates the user by means of a traditional certificate based authentication of the user's PKI card.

[0099] If the user was successfully authenticated by the reader, the reader generates an OTP or MAC (using a traditional strong authentication token algorithm) that can be validated by the validation server. The user can then submit this OTP or MAC to the server as proof that.he has been successfully authenticated by the reader.

[0100] The reader locally authenticates the user by means of the user's inserted PKI card and using traditional PKI technology, In a typical embodiment this can be done as follows (refer to Fig 8):

1. 1. The reader 800 validates the card's certificate 806 (or certificate chain).
    1. a. Note: this assumes that the reader has access to the trusted public key of the (root) Certificate Authority. This can be done by storing the trusted public key of the (root) Certificate Authority in the reader.
    2. b. Note: the reader 800 does not have to do an explicit verification of the entire certificate (chain) starting from the (root) CA public key each time the card is inserted in the reader. Instead the reader 800 can do the entire verification when a card 805 is inserted for the first time into the reader. The reader can then store the verified certificate or the certificate's public key or a reference value derived from the verified certificate or public key (e.g: a hash of the certificate or public key). If the card 805 is then reinserted at a later time, the reader 800 no longer has to do all the calculations associated with certificate validation, but can just compare the certificate on the card with the certificate or reference value stored in the reader.
2. 2. The reader 800 does a challenge-response authentication of card's private key:
    1. a. Reader (810) generates a challenge 811, e.g. typically a random number or some other non-predictable value that is e.g. derived from a time value or counter value with a cryptographic algorithm using some secret stored in the reader.
    2. b. The user provides the PIN protecting the card's private key.
    3. c. The reader 800 submits the PIN to the card.
    4. d. The reader 800 submits a random challenge 811 to card to be encrypted by card's private key.
    5. e. The card signs (815) the reader challenge with its private key and returns response (= encrypted challenge 816).
    6. f. The reader 800 decrypts card's response with card's public key (from the certificate).
    7. g. The reader compares 820 decrypted card's response with originally generated challenge. If the decrypted card's response is the same as the originally generated challenge, then the card's private key is authenticated and hence the user is authenticated.

[0101] In essence the reader generates (825) an OTP/MAC in the same way as a traditional strong authentication algorithm. All the steps of this algorithm (capturing the inputs, formatting the inputs, encrypting or hashing the formatted inputs, transforming the resulting cryptogram of hash into an OTP/MAC) are done by the reader 800 in essentially the same way as a traditional strong authentication token. In one embodiment the reader is personalized with a symmetric secret strong authentication key. In that case the reader 800 is also typically configured to expect a specific card. The reader recognizes this card by means of some characteristic value of a data element of the card. Typically the card's certificate is used as such a data element. In other embodiments(see Fig 9), in order to avoid having to personalize and configure the readers, the reader 800 derives (835) a card-specific value for the symmetric secret strong authentication key from the following data elements:

◦ public card data preferably related to the card's certificate or public key (e.g. card serial number, certificate serial number, public key, etc.)

◦ a master key 846 stored in the reader and known to the server. This master key can be:

▪ an identical value for all readers

▪ a specific/unique value for each individual reader. This requires some kind of assignment of the reader to the user, and registration of this assignment at the server.

◦ an (optional) extra derivation data element could be a (secret) data element that is provided to the reader by the user. The user must explicitly provide this data element:

▪ either, each time the reader and card are used in this way or,

▪ only when this card is used for the first time with this reader (after which the reader will remember the provided value of the data element for this card)

[0102] The reader 800 uses the derived card-specific symmetric authentication key 836 in a symmetric strong authentication algorithm (such as the Digipass algorithm or OATH) to generate (845) a dynamic password (challenge-response and/or time and/or event based) or generate (845) a MAC-type of electronic signature on some transaction data (optionally including time and/or event counter information).

[0103] A Server validates the generated dynamic password or signature as follows:

- The server derives the same card-specific symmetric strong authentication key as the reader. This assumes that the server has a database (or an alternative way of retrieving the required information) that links the user to:

  ◦ the public card data,

- the data element provided by the user (if applicable)
- and the reader's master key

Note: instead of doing this derivation each time a validation must be done, the derivation can also be done once and the resulting derived key can be stored in a database for future use.

- The server validates the dynamic password or signature in the same way as it would do for a traditional strong authentication token.

[0104] A typical embodiment operates as follows (Figs 10-11):

In an enlistment phase, a bank customer 1001 goes to a bank branch 1003. Using his national electronic identity card (e-id card 1002) with a Bank Branch Terminal (BBT), the customer electronically signs an e-banking contract 1004.

[0105] While the customer's e-id card is inserted in the BBT (1010), the BBT:

- captures the customer's certificate (1011),
- generates a random seed challenge (1012),
- submits the random seed challenge to the e-id card (1002) to be encrypted by the card's private key (1013),
- captures the card's cryptogram on that challenge (1014).

[0106] Finally, the BBT sends the customer's certificate, generated seed challenge, and the card's cryptogram on the seed challenge to a server (1015). The server stores this data in a database linked to the customer. The bank then delivers an unconnected smart card reader to the customer. This reader contains a secret master key. The bank also sends the customer a PIN mailer with the value of the seed challenge that was generated and used by the BBT. The authentication server is also informed of the value of the secret master key.

[0107] When the customer uses the reader for the first time:

- The reader asks for the customer's e-id card to be inserted.
- The reader also asks for the PIN mailer's seed challenge and stores it in memory.
- The reader reads the card's certificate and stores it also in memory.
- The reader generates a random reader challenge and submits it to the card to be encrypted

by the card's private key. The reader stores both the reader challenge and the corresponding cryptogram generated by the card.

[0108] If the customer wants to generate an OTP (or MAC or response or ...) the reader goes through the following steps:

◦ The reader asks for the customer's e-id card to be inserted.

◦ The reader validates the card:

▪ The reader reads the card's certificate and compares it to the certificate that was stored.

▪ If that checks OK, the reader submits the stored reader challenge to the card for signature and compares the card's cryptogram with the stored cryptogram.

◦ If the reader has successfully validated the card, the reader generates the secret authentication key:

▪ The reader submits the stored PIN mailer seed challenge to the card to be encrypted by the card.

▪ The reader now derives a secret authentication key from:

- the secret master key in the reader,
- the PIN mailer seed challenge,
- the card's cryptogram on that PIN mailer seed challenge,
- the card's certificate.

◦ The reader now uses the generated secret authentication key in a strong authentication algorithm (e.g. to generate an OTP or a MAC).

[0109] The authentication server is capable of verifying the resulting OTP (or MAC) since it had access to all the data necessary to generate the secret authentication key:

◦ the reader's secret master key,

◦ the card's certificate,

◦ the PIN mailer challenge,

◦ the card's cryptogram on the PIN mailer challenge.

[0110] Using the generated secret authentication key, the authentication server can validate

the OTPs or MACs in the same way it would validate OTPs or MACs generated by traditional strong authentication tokens.

[0111] Alternatively the authentication server can use either of the procedures shown in figs 13 or 14 for a validation operation.

[0112] In connection with the procedure of fig. 13, we assume that the cryptogram produced by the reader is transformed using a sequence of transform A (1306) and transform B (1307). For validation purposes the server subjects the OTP or MAC to the reverse transForm B(1327) to produce a modified OTP or MAC and then subjects the reference cryptogram to transform A (1325) to produce the modified reference cryptogram. Finally the server effects a comparison of the modified reference cryptogram and the modified OTP or MAC.

[0113] In connection with the procedure of fig. 14, we assume that the cryptogram produced by the reader is transformed using a sequence of the bit selection (1355), concatenation (1356) and bit string transformation (1357) as shown in fig. 14 to produce the OTP or MAC. For validation purposes the server subjects the OTP or MAC to the bit stream and expansion processes 1359 and 1360 of fig. 14 to produce a modified OTP or MAC. The server subjects the reference cryptogram to operation 1364 to produce the modified reference cryptogram. Finally the server effects a comparison (1365) of the modified reference cryptogram and the modified OTP or MAC to effect validation.

[0114] The foregoing has described several aspects or embodiments comprising methods or devices. In another aspect the invention comprises a sequence of instructions recorded on a computer readable medium which, when executed by a processor perform methods as already described. Software delivery can also be effected over digital networks such as the Internet. Accordingly in still a further aspect the invention comprehends an information bearing signal which comprises a sequence of instructions which, when executed by a processor perform methods as already described.

[0115] While several embodiments of the invention have been described with some particularity it should be understood that this description is exemplary and not limiting; the scope of the invention is to be determined by the claims appended hereto.

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US2005050330A1 [0022]
- US5937068A [0025]
- US2006136748A1 [0026]
- US6088450A [0026]

Non-patent literature cited in the description

- HOTP: An HMAC-based One Time Password Algorithm, 2004, [0023]
- **M. WARD**EMV card payments - An updateInformation Security Technical Report, 2006, vol. 11, 289-92 [0024]

Patentkrav

1. Fremgangsmåde til anvendelse af en første anordning (600, 800) og en anden anordning (605, 805) til generering af en sikkerhedsværdi, der omfatter en engangsadgangskode eller en signatur til meddelelsesautentifikationskode, hvilken fremgangsmåde omfatter:

anvendelse af den første anordning til at sende først information til den anden anordning og anvendelse af den anden anordning til at modtage den første information, hvorved den første anordning indgiver (710, 811) en anmodning om adgang til en anordning til den anden anordning

anvendelse af den anden anordning til at lagre en privat nøgle og til at udføre en asymmetrisk kryptografisk operation på anmodningen om adgang til en anordning med den private nøgle for at producere et resulterende kryptogram

anvendelse af den anden anordning til at sende anden information til den første anordning og anvendelse af den første anordning til at modtage den anden information, hvorved den anden anordning returnerer (816) det resulterende kryptogram til den første anordning

anvendelse af den første anordning til at verificere (711, 810) det resulterende kryptogram

anvendelse af den første anordning til at erhverve værdier af et eller flere variable inputs, som indeholder en eller flere af: en tidsværdi, en tællerværdi, en adgangsanmodningsværdi og transaktionsdata

anvendelse af den første anordning til at opnå en dynamisk mellemværdi ved hjælp af en beregning ved anvendelse af det ene eller de flere variable inputs og en symmetrisk kryptografisk algoritme, der anvender mindst én hemmelighed, hvorved oprettelsen af den dynamiske mellemværdi omfatter anvendelse af en hemmelighed eller hemmeligheder, der er lagret i den første anordning, og data, som er knyttet til den private nøgle

anvendelse af den første anordning til at omdanne den dynamiske værdi til sikkerhedsværdien, der omfatter en engangsadgangskode eller en signatur til

meddelelsesautentifikationskode,
hvorved omdannelsen omfatter fremstilling af sikkerhedsværdien af en størrelse, som er mindre end størrelsen af det kryptogram, der blev genereret ved den asymmetriske kryptografiske operation
og ved anvendelse af den første anordning til at fremvise sikkerhedsværdien på et display (26) i den første anordning (600, 800),
hvorved den første anordning genererer (825) sikkerhedsværdien på betingelse af (820), at verifikationen af det resulterende kryptogram er vellykket,
hvorved den første anordning (600, 800) er en smartkortlæser (20), der omfatter en grænseflade for at acceptere et smartkort, og en strømforsyning, og den anden anordning (605, 805) er et smartkort.

2. Fremgangsmåde ifølge krav 1, hvorved:
den symmetriske kryptografiske algoritme anvender en symmetrisk hemmelighed, der er afledt (835) af et eller flere af det følgende:
data, der identificerer den første anordning (600, 800)
en hemmelighed eller hemmeligheder (846), der er gemt i den første anordning
data (806), der er knyttet til den private nøgle
data (806), der er gemt i den anden anordning
data, der identificerer en bruger af den første anordning, eller
en hemmelighed, der tilvejebringes af brugeren af den første anordning.

3. Fremgangsmåde ifølge krav 1 eller 2, hvorved en første anmodning om adgang til en anordning indgives til den anden anordning (605) i det mindste en første gang og i det mindste en anden gang, idet den anden anordning producerer et indledende kryptogram, når den første anmodning om adgang til en anordning indgives den første gang, og producerer et senere kryptogram, når den første anmodning om adgang til en anordning indgives (710) den anden gang, hvorved

fremgangsmåden yderligere omfatter:
afledning af en referenceværdi ud fra det oprindelige kryptogram
lagring (617) af referenceværdien i den første anordning (600), og
verifikation (711) af det senere kryptogram ved at sammenligne det senere kryptogram med den lagrede referenceværdi.

4. Fremgangsmåde ifølge krav 1 eller 2, der yderligere omfatter:
opretholdelse under lagring i den første anordning (600) af mindst én første anmodning om adgang til en anordning sammen med mindst én tilsvarende verifikationsværdi, hvorved verifikationen (711) af det resulterende kryptogram omfatter sammenligning af det resulterende kryptogram med den tilsvarende verifikationsværdi.

5. Fremgangsmåde ifølge krav 1 eller 2, der yderligere omfatter:
udførelse under en første brug af den første anordning (600) med den anden anordning (605) af en sekvens, der omfatter:
generering af mindst én anmodning om adgang
indgivelse til den anden anordning af den mindst ene anmodning om adgang for en asymmetrisk kryptografisk operation ved hjælp af den anden anordning ved anvendelse af den private nøgle
modtagelse fra den anden anordning af mindst ét resulterende referencekryptogram
afledning af en verifikationsværdi ud fra det i det mindste ene resulterende referencekryptogram, og
lagring af data, der repræsenterer den mindst ene anmodning om adgang sammen med den mindst ene verifikationsværdi.

6. Fremgangsmåde ifølge krav 5, hvorved den første anordning (600) fra tid til anden udfører en sekvens, der omfatter:
generering af en ny anmodning om adgang til en anordning
indgivelse til den anden anordning (605) af den nye anmodning om adgang til en anordning for en asymmetrisk kryptografisk operation ved hjælp af den anden anordning ved anvendelse af

den private nøgle
modtagelse fra den anden anordning af mindst ét tilsvarende referencekryptogram
afledning af en ny verifikationsværdi ud fra det tilsvarende nye referencekryptogram, og
lagring i hukommelse af data, der repræsenterer den nye anmodning om adgang til en anordning, sammen med den nye verifikationsværdi.

7. Fremgangsmåde ifølge krav 1 eller 2, hvorved:
verifikation (810) ved hjælp af den første anordning (800) af det resulterende kryptogram omfatter brug af en offentlig nøgle (806), der svarer til den anden anordnings (805) private nøgle.

8. Fremgangsmåde ifølge krav 7, hvorved:
den første anordning (800) verificerer et certifikat (806) eller en certifikatkæde, som svarer til den offentlige nøgle, der svarer til den anden anordnings (805) private nøgle.

9. Fremgangsmåde ifølge krav 8, hvorved:
den første anordning (800) opretholder i hukommelsen en certifikatmyndigheds offentlige nøgle, og hvorved verifikationen (810) ved hjælp af den første anordning (800) af certifikatet eller certifikatkæden omfatter anvendelse af den af en certifikatmyndighed udstedte offentlige nøgle.

10. Apparat (20, 600, 800) til generering af en sikkerhedsværdi, der omfatter en engangsadgangskode eller en signatur til meddelelsesautentifikationskode, hvilket apparat omfatter:
erhvervelsesmidler for variabelt input, beregnet til at erhverve værdier af et eller flere variable inputs, som indeholder en eller flere af: en tidsværdi, en tællerværdi, en adgangsanmodningsværdi og transaktionsdata
behandlingsmidler, der er beregnet til at opnå en dynamisk mellemværdi ved hjælp af en beregning ved anvendelse af data tilknyttet det ene eller de flere variable inputs og en

symmetrisk kryptografisk algoritme, der anvender mindst én hemmelighed, og til at omdanne den dynamiske værdi til sikkerhedsværdien, hvorved oprettelsen af den dynamiske mellemværdi omfatter anvendelse af en hemmelighed eller hemmeligheder, der er lagret i den første anordning, og data, som er knyttet til den private nøgle

kommunikationsmidler (28), der er beregnet til at sende (811) inputdata til et andet apparat (605, 805) og til at modtage (816) et asymmetrisk kryptogram fra det andet apparat, hvilket asymmetrisk kryptogram produceres (815) af det andet apparat ved anvendelse af en asymmetrisk kryptografisk operation og en asymmetrisk privat nøgle, og

et display (26), der er konfigureret til at vise sikkerhedsværdien, hvorved behandlingsmidlerne ydermere er indrettet til at anvende (711, 820) det modtagne asymmetriske kryptogram til generering af sikkerhedsværdien, og

størrelsen af sikkerhedsværdien er mindre end størrelsen af det modtagne asymmetriske kryptogram, og

anvendelsen af det modtagne asymmetriske kryptogram omfatter verifikation (711, 820) af det modtagne asymmetriske kryptogram og generering af sikkerhedsværdien på betingelse af, at verifikationen af det modtagne asymmetriske kryptogram er vellykket

hvorved apparatet (20, 600, 800) er en smartkortlæser (20), der omfatter en grænseflade for at acceptere et smartkort, og en strømforsyning, og det andet apparat (605, 805) er et smartkort.

11. Apparat (600) ifølge krav 10, der yderligere omfatter en hukommelse til lagring (617) af en referenceværdi, hvorved de inputdata, der sendes til det andet apparat (605), omfatter en anmodning om adgang til en læser, og anvendelsen af det modtagne asymmetriske kryptogram omfatter verifikation (711) af det modtagne asymmetriske kryptogram ved sammenligning af det modtagne asymmetriske kryptogram med den lagrede referenceværdi og generering af sikkerhedsværdien på betingelse af, at verifikationen af det modtagne asymmetriske kryptogram er vellykket.

12. Apparat (800) ifølge krav 10, hvor de inputdata, der sendes (811) til det andet apparat (805), omfatter en anmodning om adgang til en læser, og anvendelsen af det modtagne (816) asymmetriske kryptogram omfatter verifikation (820) af det modtagne asymmetriske kryptogram ved anvendelse af en offentlig nøgle (806), der svarer til den asymmetriske private nøgle, og generering af sikkerhedsværdien på betingelse af (820), at verifikationen af det modtagne asymmetriske kryptogram er vellykket.

13. Apparat (800) ifølge krav 12, hvor verifikationen af det modtagne asymmetriske kryptogram desuden omfatter verifikation af et certifikat (806) eller en certifikatkæde, der svarer til den offentlige nøgle.

14. Apparat (800) ifølge krav 13, der desuden omfatter en hukommelse, i hvilken hukommelse der er lagret en certifikatmyndigheds offentlige nøgle, der anvendes til at verificere (810) det certifikat eller den certifikatkæde, som svarer til den offentlige nøgle.

DRAWINGS

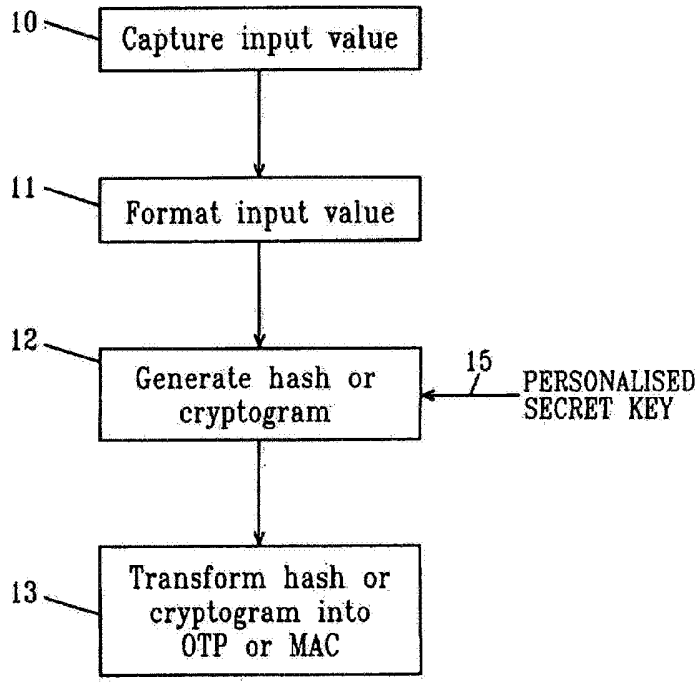

FIG.1

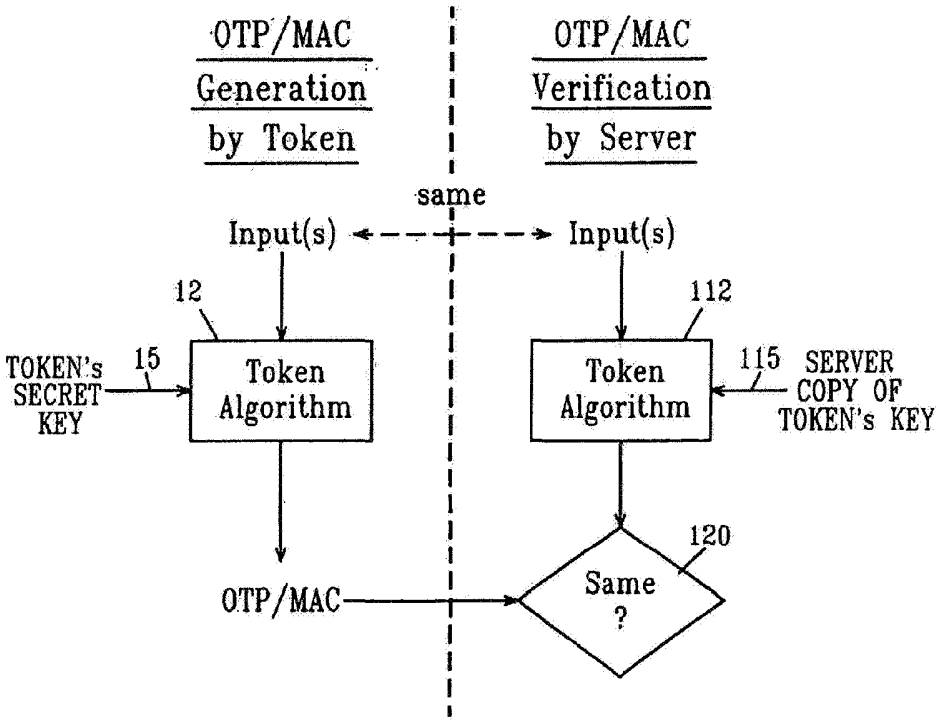
OTP/MAC
Generation
by Token
OTP/MAC
Verification
by Server
same
Input(s)
Input(s)
12
112
TOKEN's
SECRET
KEY
15
Token
Algorithm
Token
Algorithm
115
SERVER
COPY OF
TOKEN's KEY
120
OTP/MAC
Same
?

FIG.2

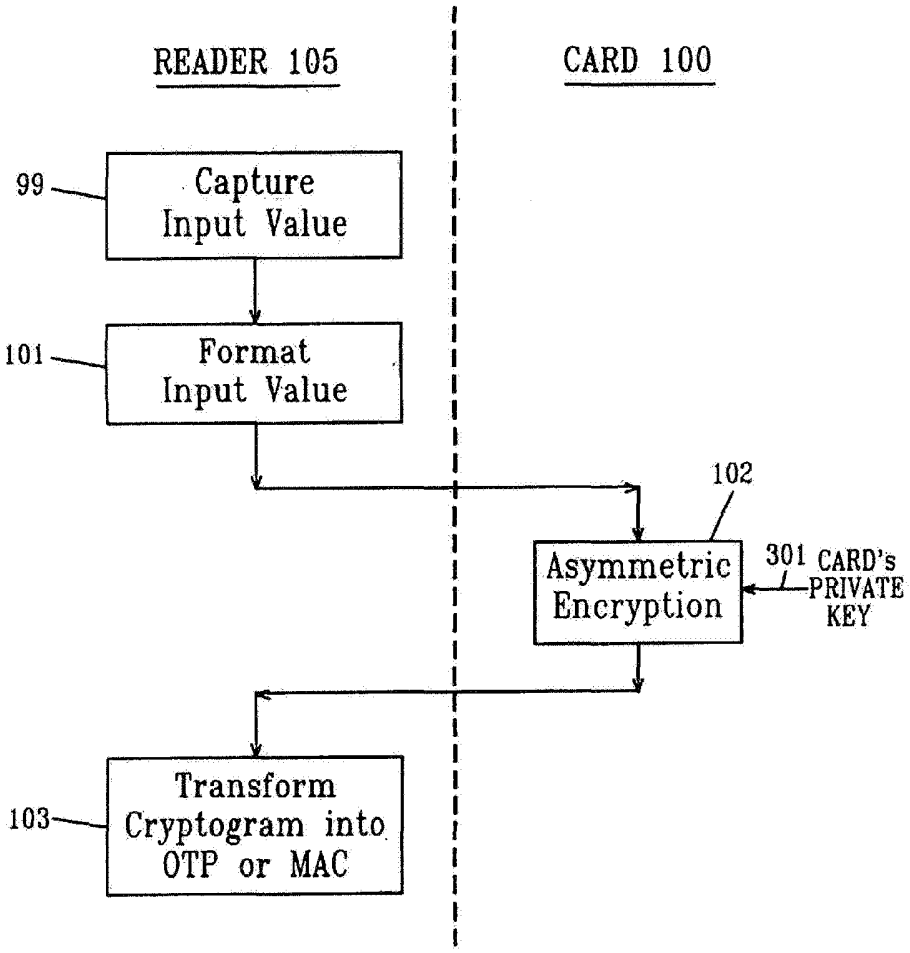
READER 105
CARD 100
99
Capture Input Value
101
Format Input Value
102
Asymmetric Encryption
301
CARD's PRIVATE KEY
103
Transform Cryptogram into OTP or MAC

FIG.3

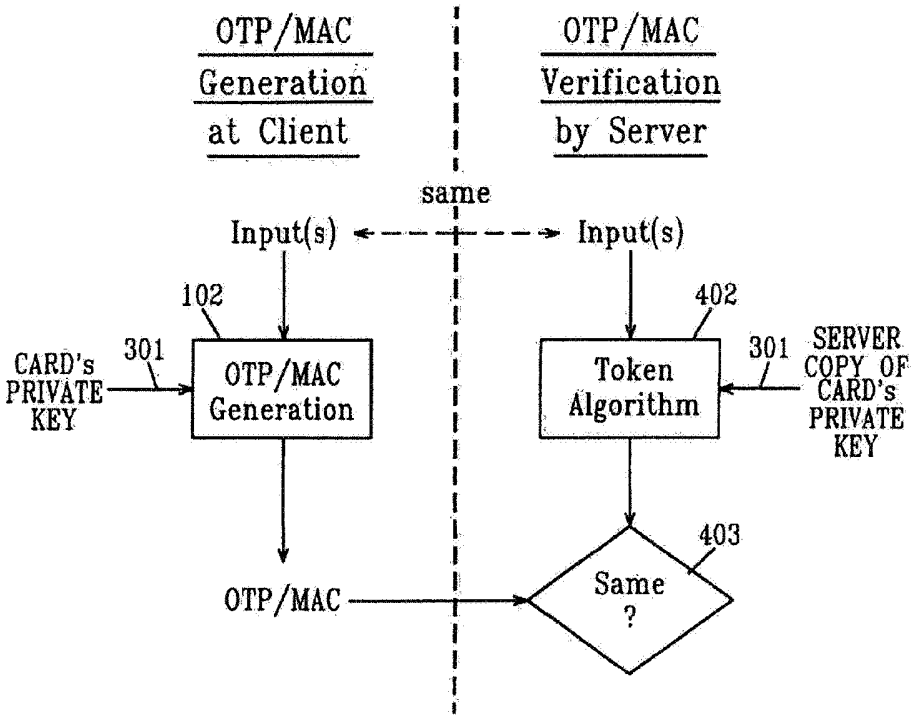
OTP/MAC
Generation
at Client
OTP/MAC
Verification
by Server
same
Input(s)
Input(s)
102
402
CARD's
PRIVATE
KEY
301
OTP/MAC
Generation
Token
Algorithm
301
SERVER
COPY OF
CARD's
PRIVATE
KEY
403
OTP/MAC
Same
?

FIG.4

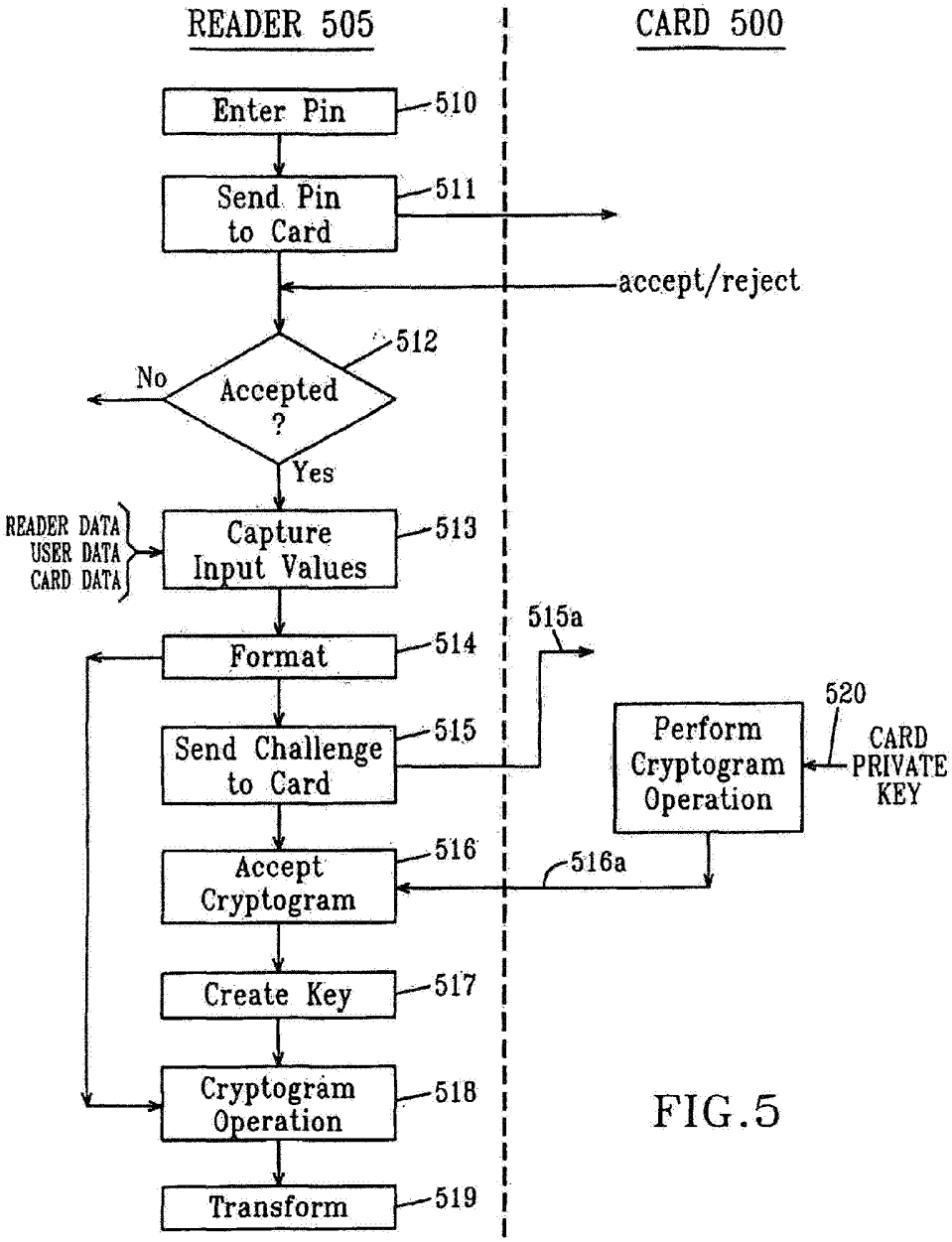
READER 505
CARD 500
Enter Pin
510
Send Pin to Card
511
accept/reject
512
No
Accepted ?
Yes
READER DATA
USER DATA
CARD DATA
Capture Input Values
513
Format
514
515a
Send Challenge to Card
515
Perform Cryptogram Operation
520
CARD PRIVATE KEY
Accept Cryptogram
516
516a
Create Key
517
Cryptogram Operation
518
Transform
519

FIG.5

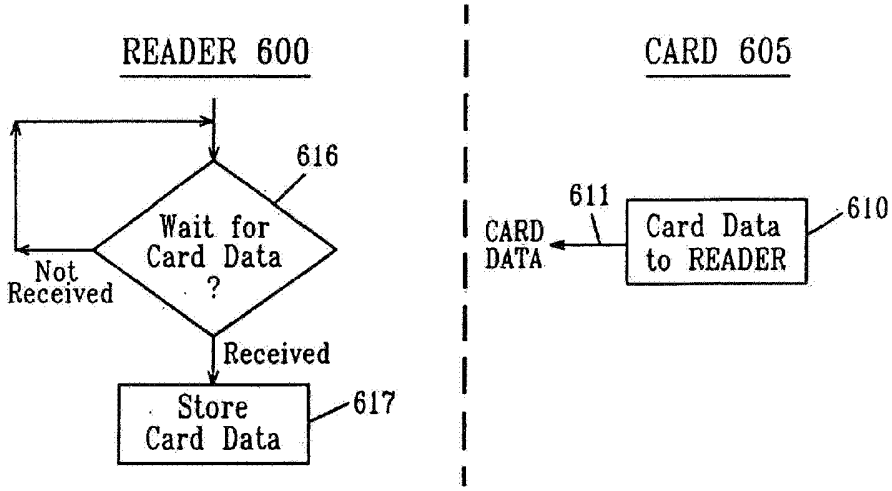
READER 600
CARD 605
616
Wait for Card Data ?
Not Received
Received
Store Card Data
617
611
CARD DATA
Card Data to READER
610

FIG.6

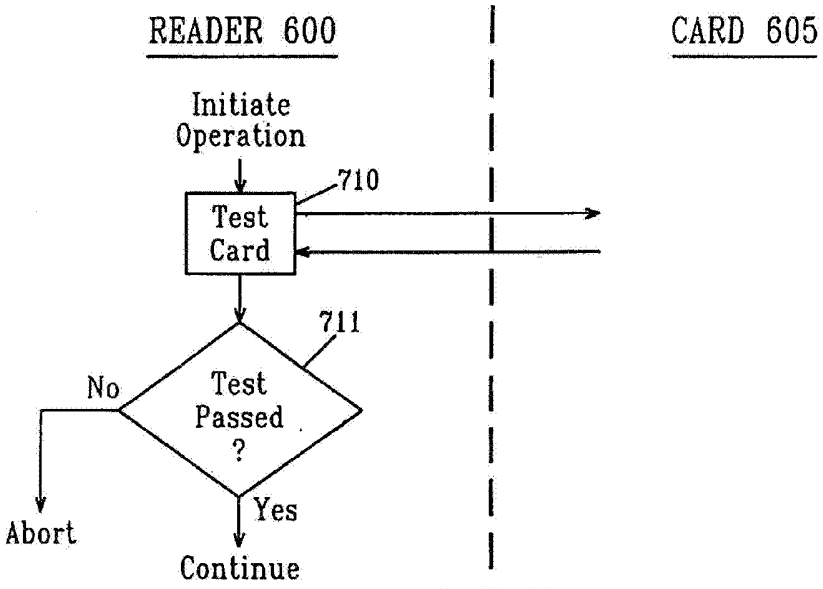
READER 600
CARD 605
Initiate Operation
710
Test Card
711
Test Passed ?
No
Abort
Yes
Continue

FIG.7

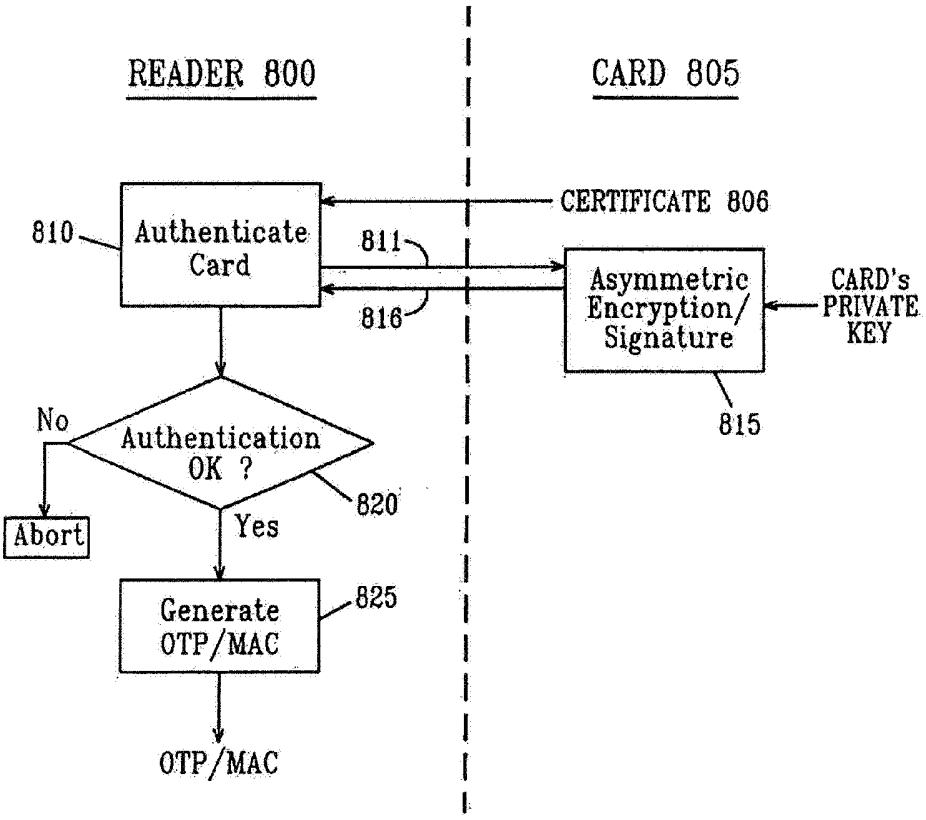
READER 800
CARD 805
CERTIFICATE 806
810
Authenticate Card
811
816
Asymmetric Encryption/ Signature
CARD's PRIVATE KEY
815
No
Authentication OK ?
820
Abort
Yes
Generate OTP/MAC
825
OTP/MAC

FIG.8

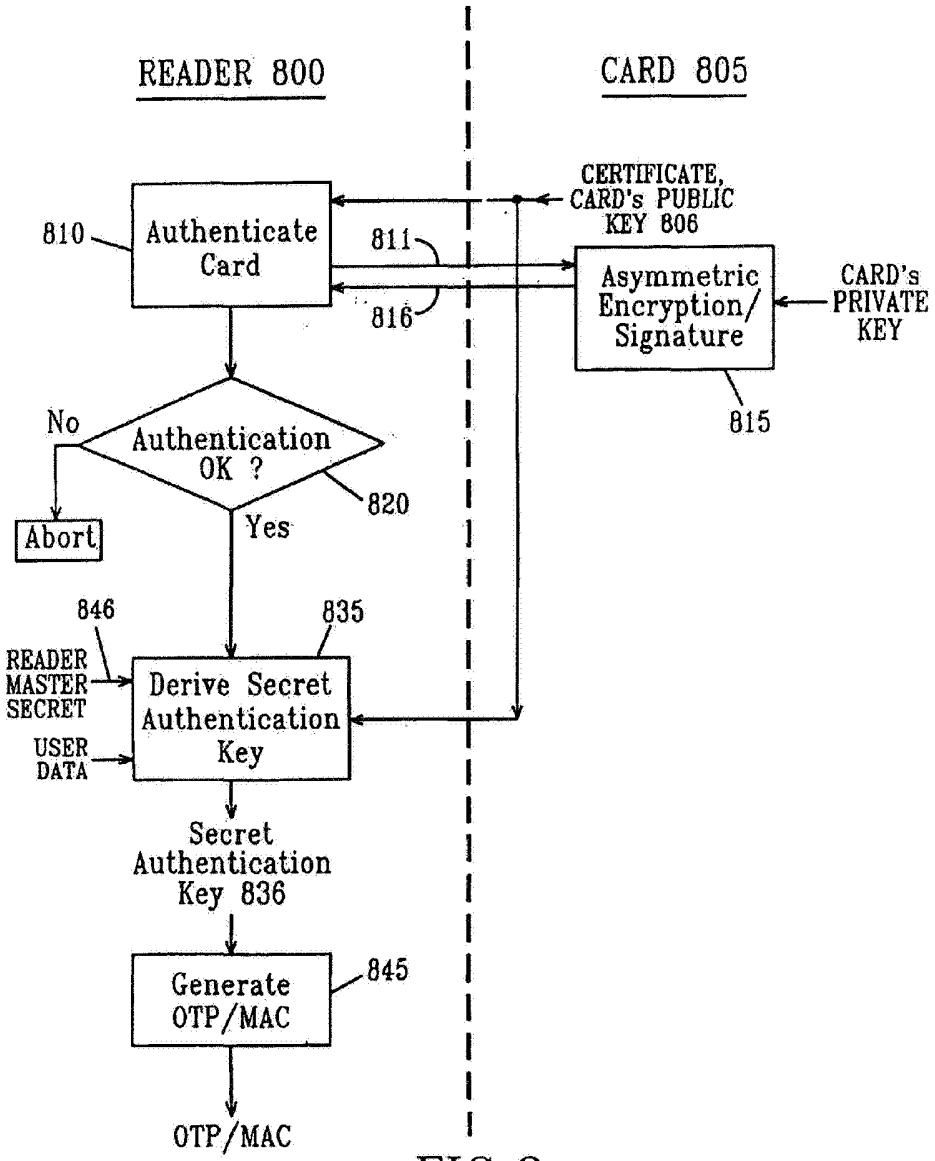
READER 800
CARD 805
CERTIFICATE, CARD's PUBLIC KEY 806
810
Authenticate Card
811
816
Asymmetric Encryption/ Signature
CARD's PRIVATE KEY
815
No
Authentication OK ?
820
Yes
Abort
846
835
READER MASTER SECRET
USER DATA
Derive Secret Authentication Key
Secret Authentication Key 836
Generate OTP/MAC
845
OTP/MAC

FIG.9

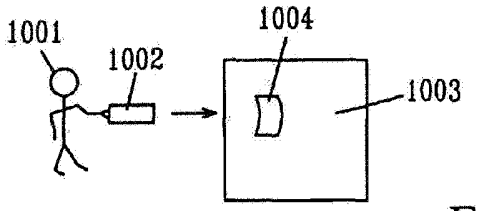
1001
1002
1004
1003

FIG.10

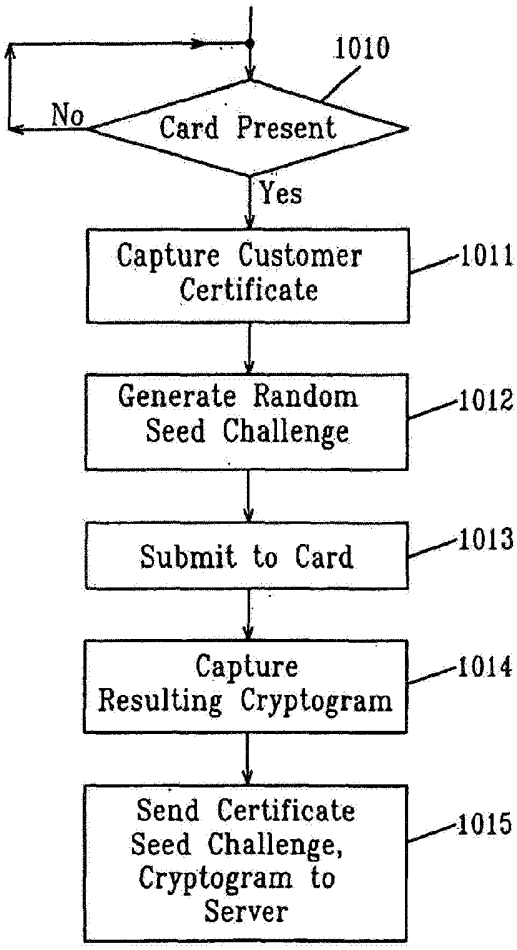
1010
No
Card Present
Yes
Capture Customer Certificate
1011
Generate Random Seed Challenge
1012
Submit to Card
1013
Capture Resulting Cryptogram
1014
Send Certificate Seed Challenge, Cryptogram to Server
1015

FIG.11

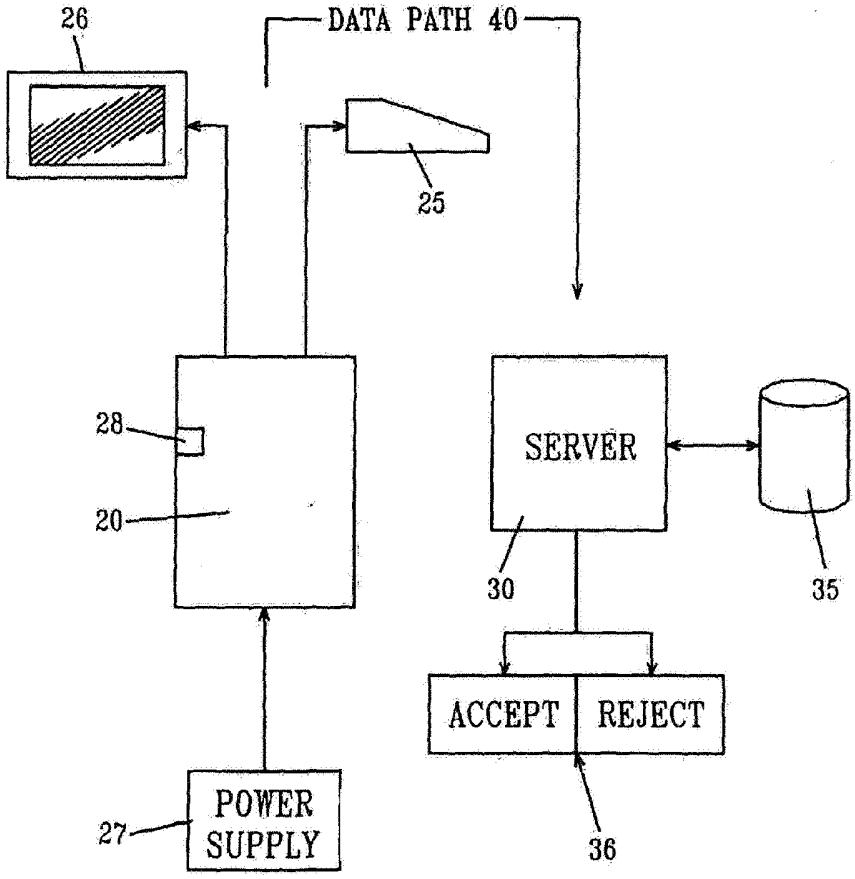
26
DATA PATH 40
25
28
20
SERVER
30
35
ACCEPT
REJECT
27
POWER SUPPLY
36

FIG.12

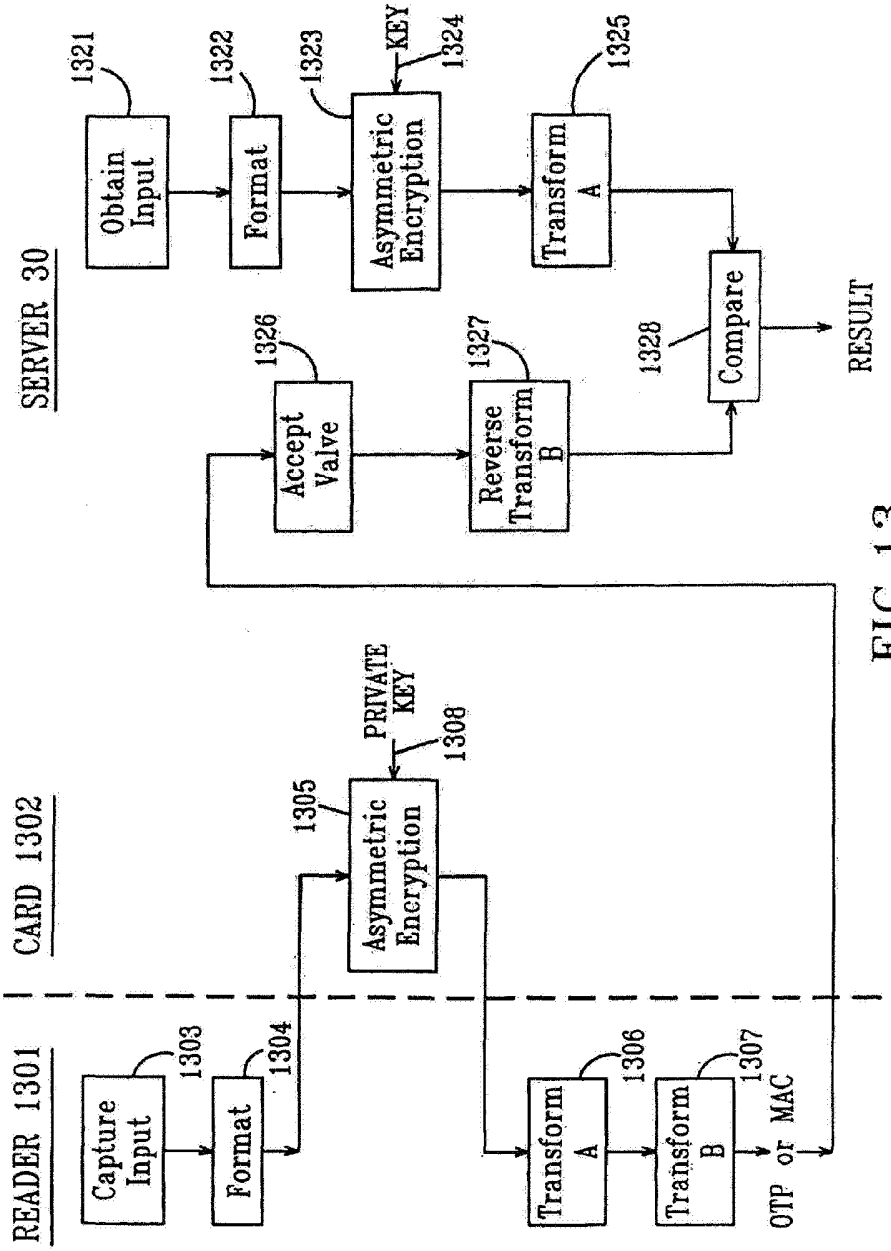
SERVER 30
1321
Obtain Input
1322
Format
1323
Asymmetric Encryption
KEY
1324
1325
Transform A
1326
Accept Valve
1327
Reverse Transform B
1328
Compare
RESULT
CARD 1302
PRIVATE KEY
1308
1305
Asymmetric Encryption
READER 1301
1303
Capture Input
1304
Format
1306
Transform A
1307
Transform B
OTP or MAC

FIG. 13

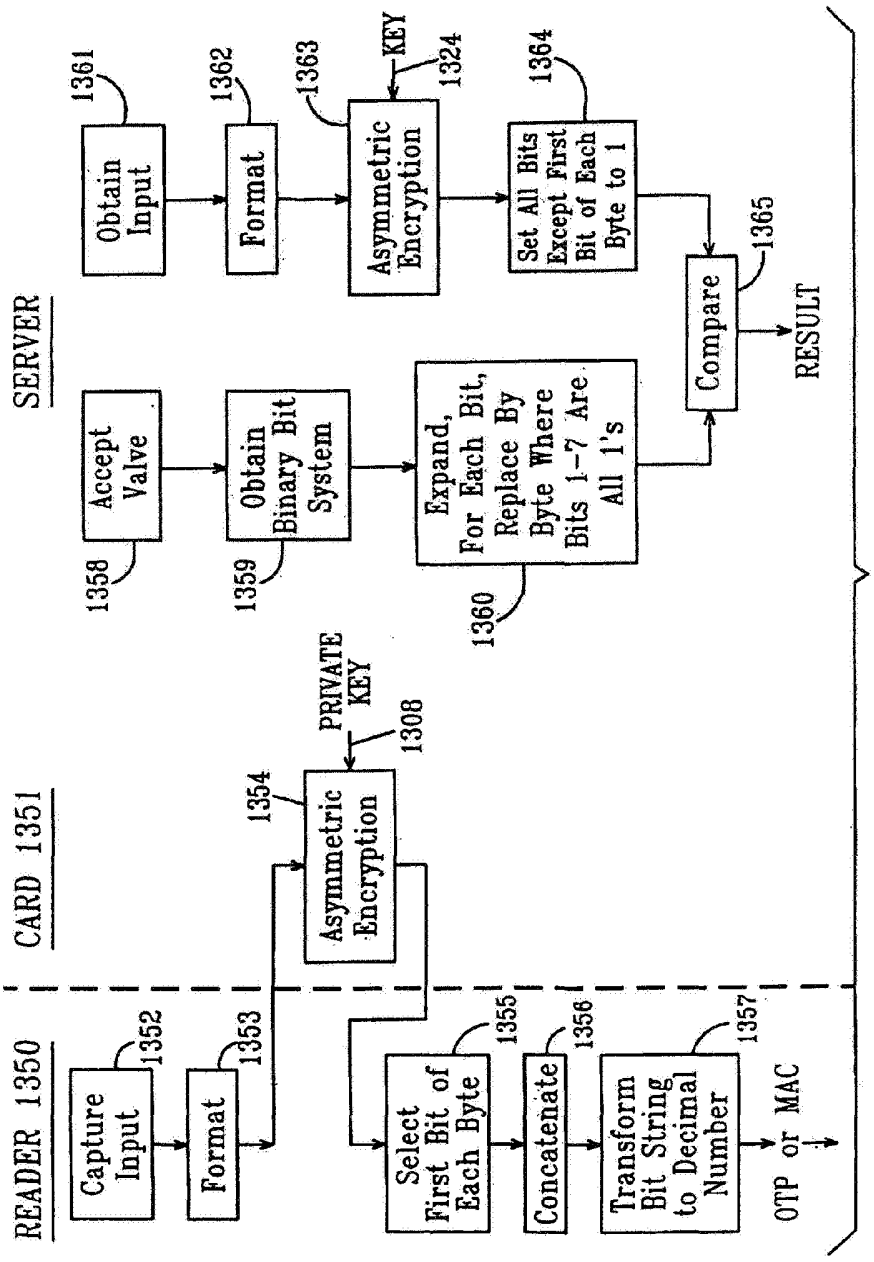
READER 1350
CARD 1351
SERVER
Capture Input
1352
Format
1353
Asymmetric Encryption
1354
PRIVATE KEY
1308
Select First Bit of Each Byte
1355
Concatenate
1356
Transform Bit String to Decimal Number
1357
OTP or MAC
Accept Valve
1358
Obtain Binary Bit System
1359
Expand, For Each Bit, Replace By Byte Where Bits 1-7 Are All 1's
1360
Obtain Input
1361
Format
1362
Asymmetric Encryption
1363
KEY
1324
Set All Bits Except First Bit of Each Byte to 1
1364
Compare
1365
RESULT

FIG. 14