(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 844 392 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
04.07.2012 Bulletin 2012/27

(21) Application number: **06704329.9**

(22) Date of filing: **23.01.2006**

(51) Int Cl.:
G06F 7/58 $^{(2006.01)}$ *H04L 9/28* $^{(2006.01)}$

(86) International application number:
PCT/CA2006/000065

(87) International publication number:
WO 2006/076804 (27.07.2006 Gazette 2006/30)

(54) ELLIPTIC CURVE RANDOM NUMBER GENERATION

ELLIPTISCHE KURVE-ZUFALLSZAHLENERZEUGUNG

GENERATION DE NOMBRE ALEATOIRE PAR COURBE ELLIPTIQUE

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI SK TR

(30) Priority: **21.01.2005 US 644982 P**

(43) Date of publication of application:
17.10.2007 Bulletin 2007/42

(73) Proprietor: **Certicom Corp.**
Mississauga, Ontario L4W 0B5 (CA)

(72) Inventors:
- **VANSTONE, Scott A.**
  Campbellville, Ontario L0P 1B0 (CA)
- **BROWN, Daniel R.I.**
  Mississauga, Ontario L5N 1X8 (CA)

(74) Representative: **Finnie, Peter John et al**
Gill Jennings & Every LLP
The Broadgate Tower
20 Primrose Street
London EC2A 2ES (GB)

(56) References cited:
WO-A1-01/13218 **US-A- 6 044 388**
US-A1- 2004 102 242 **US-A1- 2004 102 242**
US-B1- 6 243 467

- **Don B. Johnson: "X9.82 Part 3 - Number Theoretic DRBGs" NIST RNG Workshop 20 July 2004 (2004-07-20), XP002552878 Retrieved from the Internet: URL:http://csrc.nist.gov/groups/ST/ toolkit /documents/rng/ NumberTheoreticDRBG.pdf> [retrieved on 2009-10-29]**
- **LEE ET AL.: 'Elliptic Curve Random Number Generation' ELECTRICAL AND ELECTRONIC TECHNOLOGY, 2001. TENCON. PROCEEDINGS OF IEEE REGION 10 INTERNATIONAL CONFERENCE vol. 1, 19 August 2001 - 22 August 2001, pages 239 - 241, XP010556240**
- **KALISKI B.S.: 'A Pseudo-Random Bit Generator Based On Elliptic Logarithms' ADVANCES IN CRYPTOLOGY, CRYPTO 1986 vol. 63, 1987, pages 84 - 103, XP008120275**

Printed by Jouve, 75001 PARIS (FR)

Description

FIELD OF THE INVENTION:

[0001] The present invention relates to systems and methods for cryptographic random number generation.

DESCRIPTION OF THE PRIOR ART

[0002] Random numbers are utilised in many cryptographic operations to provide underlying security. In public key infrastructures, for example, the private key of a key pair is generated by a random number generator and the corresponding public key mathematically derived therefrom. A new key pair may be generated for each session and the randomness of the generator therefore is critical to the security of the cryptographic system.

[0003] To provide a secure source of random numbers, cryptographically secure pseudorandom bit generators have been developed in which the security of each generator relies on a presumed intractability of the underlying number-theoretical problem. The American National Standards Institute (ANSI) has set up an Accredited Standards Committee (ASC) X9 for the financial services industry, which is preparing a American National Standard (ANS) X9. 82 for cryptographic random number generation (RNG). One of the RNG methods in the draft of X9.82, called Dual_BC_DRBG, uses elliptic curve cryptography (ECC) for its security. Dual_EC_DRBG will hereinafter be referred to as elliptic curve random number generation (ECRNG).

[0004] Elliptic curve cryptography relies on the intractability of the discrete log problem in cyclic subgroups of elliptic curve groups. An elliptic curve E is the set of points (x, y) that satisfy the defining equation of the elliptic curve. The defining equation is a cubic equation, and is non-singular. The coordinates x and y are elements of a field, which is a set of elements that can be added, subtracted and divided, with the exception of zero. Examples of fields include rational numbers and real numbers. There are also finite fields, which are the fields most often used in cryptography An example of a finite field is the set of integers modulo a prime q.

[0005] Without the loss of generality, the defining equation of the elliptic curve can be in the Weierstrass form, which depends on the field of the coordinates. When the field F is integers modulo a prime $q > 3$, then the Weierstrass equation takes the form $y^2 = x^3 + ax +$ b, where a and b are elements of the field F.

[0006] The elliptic curve E includes the points (x, y) and one further point, namely the point O at infinity. The elliptic curve E also has a group structure, which means that the two points P and Q on the curve can be added to form a third point P + Q. The point O is the identity of the group, meaning P + $O = O + P = P$, for all points P. Addition is associative, so that P + $(Q + R) = (P + Q) +$ R, and commutative, so that $P + Q = Q + R$, for all points P, Q and R. Each point P has a negative point $-P$, such that $P + (-P) =$ O. When the curve equation is the Weierstrass equation of the form $y^2 = x^3 + ax + b$, the negative of $P = (x, y)$ is determined easily as $-P = (x, -y)$. The formula for adding points P and Q in terms of their coordinates is only moderately complicated involving just a handful of field operations.

[0007] The ECRNG uses as input two elliptic curve points P and Q that are fixed. These points are not assumed to be secret. Typically, P is the standard generator of the elliptic curve domain parameters, and Q is some other point. In addition a secret seed is inserted into the ECRNG.

[0008] The ECRNG has a state, which may be considered to be an integer s. The state s is updated every time the ECRNG produces an output. The updated state is computed as $u = z(sP)$, where $z()$ is a function that converts an elliptic curve point to an integer. Generally, z consists of taking the x-coordinate of the point, and then converting the resulting field element to an integer. Thus u will typically be an integer derived from the x-coordinate of the point s.

[0009] The output of the ECRNG is computed as follows: $r = t(z(sQ))$, where t is a truncation function. Generally the truncation function removes the leftmost bits of its input. In the ECRNG, the number of bits truncated depends on the choice of elliptic curve, and typically may be in the range of 6 to 19 bits.

[0010] Although P and Q are known, it is believed that the output r is random and cannot be predicted Therefore successive values will have no relationship that can be exploited to obtain private keys and break the cryptographic functions. The applicant has recognised that anybody who knows an integer d such that $Q = dP$, can deduce an integer e such that $ed = 1 \bmod n$, where n is the order of G, and thereby have an integer e such that $P = eQ$. Suppose $U = sP$ and $R = sQ$, which are the precursors to the updated state and the ECRNG output. With the integer e, one can compute U from R as $U =$ eR. Therefore, the output $r = t(z(R))$, and possible values of R can be determined from r. The truncation function means that the truncated bits of R would have to be guessed The z function means that only the x-coordinate is available, so that decompression would have to be applied to obtain the full point R. In the case of the ECRNG, there would be somewhere between about $2^6 = 64$ and 2^{19} (i.e. about half a million) possible points R which correspond to r, with the exact number depending on the curve and the specific value of r.

[0011] The full set of R values is easy to determine from r, and as noted above, determination of the correct value for R determines $U = eR$, if one knows e. The updated state is $u = z(U)$, so it can be determined from the correct value of R. Therefore knowledge of r and e allows one to determine the next state to within a number of possibilities somewhere between 2^6 and 2^{19}. This uncertainty will invariably be eliminated once another output is observed, whether directly or indirectly through a one-way function.

[0012] Once the next state is determined, all future states of ECRNG can be determined, because the ECRNG is a deterministic function. (at least unless additional random entropy is fed into the ECRNG state) All outputs of the ECRNG are determined from the determined states of the ECRNG. Therefore knowledge of *r* and *e*, allows one to determine all future outputs of the ECRNG.

[0013] It has therefore been identified by the applicant that this method potentially possesses a trapdoor, whereby standardizers or implementers of the algorithm may possess a piece of information with which they can use a single output and an instantiation of the RNG to determine all future states and output of the KNG, thereby completely compromising its security. It is therefore an object of the present invention to obviate or mitigate the above mentioned disadvantages.

SUMMARY OF THE INVENTION

[0014] According to a first aspect of the present invention there is provided a method of operating an elliptic curve random number generator including an arithmetic unit to perform elliptic curve operations to compute a random number for use in a cryptographic operation, said method comprising the steps of:

obtaining a pair of inputs, wherein each input is representative of at least one coordinate of respective ones of a pair of elliptic curve points, and wherein at least one of said inputs is obtained in a manner to ensure that one point of the pair of points would not have been chosen as a known multiple of the other point of the pair of points;
providing said pair of inputs as inputs to said arithmetic unit;
performing selected elliptic curve operations on said inputs to obtain an output; and,
utilising said output as a random number in the cryptographic operation.

[0015] In one embodiment, the method of the present invention provides for computing a verifiably random point Q for use with another point P in an elliptic curve random number generator comprising computing a hash including the point P as an input, and deriving the point Q from the hash. In another embodiment, the method of the present invention provides for producing an elliptic curve random number comprising generating an output using an elliptic curve random number generator, and truncating the output to generate the random number.

[0016] In yet another embodiment, the method of the present invention provides for producing an elliptic curve random number comprising generating an output using an elliptic curve random number generator, and applying the output to a one-way function to generate the random number.

[0017] According to a second aspect of the present invention there is provided an elliptic curve random number generator comprising:

means to generate a pair of inputs, wherein each of said inputs is representative of at least one coordinate of respective ones of a pair of elliptic curve points, and wherein at least one of said inputs is generated in a manner to ensure that one point of the pair of points would not have been chosen as a known multiple of the other point of the pair of points; and,
an arithmetic unit to perform elliptic curve operations on said inputs, said arithmetic unit having an output based on the results of said elliptic operations, said output representing a random number for use in a cryptographic operation.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] An embodiment of the invention will now be described by way of example only with reference to the appended drawings wherein:

[0019] Figure 1 is a schematic representation of a cryptographic random number generation scheme.

[0020] Figure 2 is a flow chart illustrating a selection process for choosing elliptic curve points.

[0021] Figure 3 is a block diagram, similar to figure 1 showing a further embodiment

[0022] Figure 4 is flow chart illustrating the process implemented by the apparatus of Figure 3.

[0023] Figure 5 is a block diagram showing a further embodiment.

[0024] Figure 6 is a flow chart illustrating yet another embodiment of the process of Figure 2.

[0025] Figure 7 is schematic representation of an administrated cryptographic random number generation scheme.

[0026] Figure 8 is a flow chart illustrating an escrow key selection process.

[0027] Figure 9 is a flow chart illustrating a method for securely utilizing an escrow key.

DETAILED DESCRIPTION OF THE INVENTION

[0028] Referring therefore to Figure 1, a cryptographic random number generator (ECRNG) 10 includes an arithmetic unit 12 for performing elliptic curve computations. The ECRNG also includes a secure register 14 to retain a state value s and has a pair of inputs 16, 18 to receive a pair of initialisation points *P*, *Q*. The points *P*, *Q* are elliptic curve points that are assumed to be known. An output 20 is provided for communication of the random integer to a cryptographic module 22. The initial contents of the register 14 are provided by a seed input S.

[0029] This input 16 representing the point P is in a first embodiment, selected from a known value published as suitable for such use.

[0030] The input 18 is obtained from the output of a one way function in the form of a hash function 24 typically a cryptographically secure hash function such as SHA1 or SHA2 that receives as inputs the point P. The function 24 operates upon an arbitrary bit string A to produce a hashed output 26. The output 26 is applied to arithmetic unit 12 for further processing to provide the input Q.

[0031] In operation, the ECRNG receives a bit string as a seed, which is stored in the register 14. The seed is maintained secret and is selected to meet pre-established cryptographic criteria, such as randomness and Hamming weight, the criteria being chosen to suit the particular application.

[0032] In order to ensure that d is not likely to be known (e. g. such that $P = dQ$, and ed =1 mod n); one or both of the inputs 16, 18 is chosen so as to be verifiably random. In the embodiment of Figure 1, Q is chosen in a way that is verifiably random by deriving it from the output of a hash-function 24 (preferably one-way) whose input includes the point P. As shown in Figure 2 an arbitrary string A is selected at step 202, a hash H of A is computed at step 204 with P and optionally S as inputs to a hash-based function $F_H()$, and the hash H is then converted by the arithmetic unit 12 to a field element $X$of a desired field F at step 206. P may be pre-computed or fixed, or may also be chosen to be a verifiably random chosen value. The field element X is regarded as the x-coordinate of Q (thus a "compressed" representation of Q). The x-coordinate is then tested for validity on the desired elliptic curve E at step 208, and whether or not $X$is valid, is determined at step 210. If valid, the x-coordinate provided by element $X$is decompressed to provide point Q at step 212. The choice of which of two possible values of the y co-ordinate is generally derived from the hash value.

[0033] The points P and Q are applied at respective inputs 16,18 and the arithmetic unit 12 computes the point sQ where s is the current value stored in the register 14. The arithmetic unit 12 converts the x-coordinate of the point (in this example point sQ) to an integer and truncates the value to obtain $r = t(z(sQ))$. The truncated value r is provided to the output 20.

[0034] The arithmetic unit 12 similarly computes a value to update the register 14 by computing sP, where s is the value of the register 14, and converting the x-coordinate of the point sP to an integer u. The integer u is stored in the register to replace s for the next iteration. {ditto above}

[0035] As noted above, the point P may also be verifiably random, but may also be an established or fixed value. Therefore, the embodiment of Figure 1 may be applied or retrofitted to systems where certain base points (e.g. P) are already implemented in hardware. Typically, the base point P will be some already existing base point, such as those recommended in Federal Information Processing Standard (FIPS) 186-2. In such cases, P is not chosen to be verifiably random.

[0036] In general, inclusion of the point P in the input to the hash function ensures that P was determined before Q is determined, by virtue of the one-way property of the hash function and since Q is derived from an already determined P. Because P was determined before Q, it is clearly understood that P could not have been chosen as a multiple of Q (*e.g.* where $P = eQ$), and therefore finding d is generally as hard as solving a random case of the discrete logarithm problem.

[0037] Thus, having a seed value S provided and a hash-based function $F()$ provided, a verifier can determine that $Q = F(S,P)$, where P may or may not be verifiably random Similarly, one could compute P = $F(S,Q)$ with the same effect, though it is presumed that this is not necessary given that the value of P in the early drafts of X9.82 were identical to the base points specified in FIPS 186-2.

[0038] The generation of Q from a bit string as outlined above may be performed externally of the ECRNG 10, or, preferably, internally using the arithmetic unit 12. Where both P and Q are required to be verifiably random, a second hash function 24 shown in ghosted outline in Figure 1 is incorporated to generate the coordinate of point P from the bit string A. By providing a hash function for at least one of the inputs, a verifiably random input is obtained.

[0039] It will also be noted that the output generated is derived from the x coordinate of the point sP. Accordingly, the inputs 16, 18 may be the x coordinates of P and Q and the corresponding values of sP and sQ obtained by using Montgomery multiplication techniques thereby obviating the need for recovery of the y coordinates.

[0040] An alternative method for choosing Q is to choose Q in some canonical form, such that its bit representation contains some string that would be difficult to produce by generating $Q = dP$ for some known d and P for example a representation of a name. It will be appreciated that intermediate forms between this method and the preferred method may also exist, where Q is partly canonical and partly derived verifiably at random Such selection of Q, whether verifiably random, canonical, or some intermediate, can be called verifiable.

[0041] Another alternative method for preventing a key escrow attack on the output of an ECRNG, shown in Figures 3 and 4 is to add a truncation function 28 to ECRNG 10 to truncate the BCRNG output to approximately half the length of a compressed elliptic curve point. Preferably, this operation is done in addition to the preferred method of Figure 1 and 2, however, it will be appreciated that it may be performed as a primary measure for preventing a key escrow attack. The benefit of truncation is that the list of R values associated with a single ECRNG output r is typically infeasible to search For example, for a 160-bit elliptic curve group, the number of potential points R in the list is about 2^{80}, and searching the list would be about as hard as solving the discrete logarithm problem The cost of this method is that the ECRNG is made half as efficient, because the output length is effectively halved.

[0042] Yet another alternative method shown in Figure

5 and 6 comprises filtering the output of the BCRNG through another one-way function F_{H2},, identified as 34, such as a hash function to generate a new output. Again, preferably, this operation is performed in addition to the preferred method shown in Figure 2, however may be performed as a primary measure to prevent key escrow attacks. The extra hash is relatively cheap compared to the elliptic curve operations performed in the arithmetic unit 12, and does not significantly diminish the security of the ECRNG.

[0043] As discussed above, to effectively prevent the existence of escrow keys, a verifiably random *Q* should be accompanied with either a verifiably random *P* or a pre-established *P.* A pre-established *P* may be a point *P* that has been widely publicized and accepted to have been selected before the notion of the ECRNG 12, which consequently means that *P* could not have been chosen as $P = eQ$ because *Q* was not created at the time when *P* was established.

[0044] Whilst the above techniques ensure the security of the system using the ECRNG by "closing" the trap door, it is also possible to take advantage of the possible interdependence of *P* and *Q,* namely where $P = eQ$, through careful use of the existence of *e*.

[0045] In such a scenario, the value e may be regarded as an escrow key. If *P* and *Q* are established in a security domain controlled by an administrator, and the entity who generates *Q* for the domain does so with knowledge of *e* (or indirectly via knowledge of *d*). The administrator will have an escrow key for every ECRNG that follows that standard.

[0046] Escrow keys are known to have advantages in some contexts. They can provide a backup functionality. If a cryptographic key is lost, then data encrypted under that key is also lost. However, encryption keys are generally the output of random number generators. Therefore, if the ECRNG is used to generate the encryption key *K*, then it may be possible that the escrow key *e* can be used to recover the encryption key *K* Escrow keys can provide other functionality, such as for use in a wiretap. In this case, trusted law enforcement agents may need to decrypt encrypted traffic of criminals, and to do this they may want to be able to use an escrow key to recover an encryption key.

[0047] Figure 7 shows a domain 40 having a number of ECRNG's 10 each associated with a respective member of the domain 40. The domain 40 communicates with other domains 40a, 40b, 40c through a network 42, such as the internet. Each ECRNG of a domain has a pair of identical inputs P,Q. The domain 40 includes an administrator 44 who maintains in a secure manner an escrow key e.

[0048] The administrator 44 chooses the values of *P* and *Q* such that he knows an escrow key *e* such that $Q = eP$. Other members of the domain 40 use the values of *P* and *Q*, thereby giving the administrator 44 an escrow key e that works for all the members of the organization.

[0049] This is most useful in its backup functionality for protecting against the loss of encryption keys. Escrow keys e could also be made member-specific so that each member has its own escrow e' from points selected by the administrator 44.

[0050] As generally denoted as numeral 400 in Figure 8, the administrator initially selects a point P which will generally be chosen as the standard generator *P* for the desired elliptic curve 402. The administrator then selects a value d and the point *Q* will be determined as $Q = dP$ 404, for some random integer d of appropriate size. The escrow key e is computed as $e = d^{-1}$ mod n 406, where *n* is the order of the generator *P* and stored by the administrator.

[0051] . The secure use of such an escrow key 34e is generally denoted by numeral 500 and illustrated in Figure 9. The administrator 44 is first instituted 502 and an escrow keys e would be chosen and stored 504 by the administrator44

[0052] In order for the escrow key to function with full effectiveness, the escrow administrator 44 needs direct access to an ECRNG output value *r* that was generated before the ECRNG output value *k* (i.e. 16) which is to be recovered. It is not sufficient to have indirect access to r via a one-way function or an encryption algorithm. A formalized way to achieve this is to have each member with an ECRNG 12 communicate with the administrator 44 as indicated at 46 in figure 7. and step 506 in figure 9. This may be most useful for encrypted file storage systems or encrypted email accounts. A more seamless method may be applied for cryptographic applications. For example, in the SSL and TLS protocols, which are used for securing web (HTTP) traffic, a client and server perform a handshake in which their first actions are to exchange random values sent in the clear.

[0053] Many other protocols exchange such random values, often called nonces. If the escrow administrator observes these nonces, and keeps a log of them 508, then later it may be able to determine the necessary r value. This allows the administrator to determine the subsequent state of the ECRNG 12 of the client or server 510 (whoever is a member of the domain), and thereby recover the subsequent ECRNG 12 values. In particular, for the client who generally generates a random pre-master secret from which is derived the encryption key for the SSL or TLS session, the escrow key may allow recovery of the session key. Recovery of the session key allows recovery of the whole SSL or TLS session.

[0054] If the session was logged, then it may be recovered. This does not compromise long- term private keys, just session keys obtained from the output of the ECRNG, which should alleviate any concern regarding general suspicions related to escrows.

[0055] Whilst escrow keys are also known to have disadvantages in other contexts, their control within specific security domains may alleviate some of those concerns. For example, with digital signatures for non-repudiation, it is crucial that nobody but the signer has the signing key, otherwise the signer may legitimately argue the repudiation of signatures. The existence of escrow keys means the some other entity has access to the signing key, which enables signers to argue that the escrow key was used to obtain their signing key and subsequently generate their signatures. However, where the domain is limited to a particular organisation or part of an organisation it may be sufficient that the organisation cannot repudiate the signature. Lost signing keys do not imply lost data, unlike encryption keys, so there is little need to backup signing keys.

[0056] Although the invention has been described with reference to certain specific embodiments, various modifications thereof will be apparent to those skilled in the art without departing from the scope of the invention as outlined in the claims appended hereto

Claims

1. A method of operating an elliptic curve random number generator (10) including an arithmetic unit (12) to perform elliptic curve operations to compute a random number for use in a cryptographic operation (22), said method comprising the steps of:

    obtaining a pair of inputs (16,18), wherein each input (16,18) is representative of at least one coordinate of respective ones of a pair of elliptic curve points, and wherein at least one of said inputs (16,18) is obtained in a manner to ensure that one point of the pair of points would not have been chosen as a known multiple of the other point of the pair of points;
    providing said pair of inputs (16,18) as inputs to said arithmetic unit (12);
    performing selected elliptic curve operations on said inputs to obtain an output (20); and,
    utilising said output (20) as a random number in the cryptographic operation (22).

2. The method according to claim 1, wherein said at least one of said inputs (18) is obtained from an output of a hash function (24).

3. The method according to claim 2, wherein the other (16) of said inputs is utilized as an input to said hash function (24).

4. The method according to claim 2, wherein the other (16) of said inputs is obtained from an output of a hash function.

5. The method according to any one of claims 1 to 4, wherein said random number generator (10) has a secret value and said secret value is used to compute scalar multiples of said points represented by said inputs (16,18).

6. The method according to claim 5, wherein one of said scalar multiples is used to derive said random number and the other of said scalar multiples is used to change said secret value for subsequent use.

7. The method according to claim 2 or claim 3, wherein said output of said hash function (24) is validated as a coordinate of a point on an elliptic curve prior to utilization as said input.

8. The method according to claim 7, wherein another coordinate of said point is obtained from said one coordinate for inclusion as said one input.

9. The method according to claim 8, wherein said other input is a representation of an elliptic curve point.

10. The method according to claim 6, wherein said random number is derived from said one scalar multiple by selecting one coordinate of a point represented by said one scalar multiple and truncating said one coordinate to a bit string for use as said random number.

11. The method according to claim 10, wherein said one coordinate is truncated in the order of one half the length of a representation of an elliptic curve point representation.

12. The method according to claim 6, wherein said random number is derived from said one scalar multiple by selecting one coordinate of said point represented by said scalar multiple and hashing said one coordinate to provide a bit string for use as said random number.

13. The method according to claim 1, wherein said at least one of said inputs (16,18) is chosen to be of a canonical form.

14. The method of any one of claims 1 to 9, wherein said output (20) is passed through a one way function to obtain a bit string for use as a random number.

15. The method according to claim 14, wherein said one way function is a hash function.

16. The method of claim 1, wherein obtaining one of the inputs includes obtaining a result of a hash function that is performed on said one input.

17. An elliptic curve random number generator (10) comprising:

    means to generate a pair of inputs (16,18), wherein each of said inputs (16,18) is representative of at least one coordinate of respective ones of a pair of elliptic curve points, and wherein

at least one of said inputs (16,18) is generated in a manner to ensure that one point of the pair of points would not have been chosen as a known multiple of the other point of the pair of points; and,
an arithmetic unit (12) to perform elliptic curve operations on said inputs (16,18), said arithmetic unit (12) having an output (20) based on the results of said elliptic operations, said output (20) representing a random number for use in a cryptographic operation (22).

18. An elliptic curve random number generator (10) according to claim 17, wherein said means to generate said inputs (16,18) includes a one way function and at least one of said inputs (16,18) is derived from an output of a one way function.

19. An elliptic curve random number generator (10) according to claim 18, wherein said one way function is a hash function.

20. An elliptic curve random number generator (10) according to claim 17, wherein one of said inputs (16,18) is obtained from an output of a hash function, and the other of said inputs (16,18) is provided as an input to said hash function.

Patentansprüche

1. Verfahren zum Betreiben eines Zufallszahlengenerators (10) für elliptische Kurven, der eine Recheneinheit (12) einschließt, um elliptische Kurvenoperationen zum Berechnen einer Zufallszahl zur Verwendung in einer kryptografischen Operation (22) auszuführen, wobei das besagte Verfahren folgende Schritte umfasst:

Erlangen eines Eingabepaars (16, 18), wobei jede Eingabe (16, 18) repräsentativ für zumindest eine Koordinate jeweiliger eines Punktepaars einer elliptischen Kurve ist und, wobei zumindest eine der besagten Eingaben (16, 18) auf eine Weise erlangt wird, die sicherstellt, dass ein Punkt des Punktepaars nicht als eine bekannte Vielfache des anderen Punkts des Punktepaars gewählt worden wäre;
Bereitstellen des besagten Eingabepaars (16, 18) als Eingaben in die besagte Recheneinheit (12);
Durchführen selektierter elliptischer Kurvenoperationen an den besagten Eingaben, um eine Ausgabe (20) zu erlangen; und
Verwenden der besagten Ausgabe (20) als eine Zufallszahl bei der kryptografischen Operation (22).

2. Verfahren nach Anspruch 1, wobei die besagte zumindest eine der besagten Eingaben (18) aus einer Ausgabe von einer Hash-Funktion (24) erlangt wird.

3. Verfahren nach Anspruch 2, wobei die andere (16) der besagten Eingaben als eine Eingabe in die besagte Hash-Funktion (24) verwendet wird.

4. Verfahren nach Anspruch 2, wobei die andere (16) der besagten Eingaben aus einer Ausgabe einer Hash-Funktion erlangt wird.

5. Verfahren nach einem beliebigen der Ansprüche 1 bis 4, wobei der besagte Zufallszahlengenerator (10) einen geheimen Wert hat und der besagte geheime Wert verwendet wird, skalare Vielfache der besagten Punkte zu berechnen, die durch die besagten Eingaben (16, 18) repräsentiert sind.

6. Verfahren nach Anspruch 5, wobei eine der besagten skalaren Vielfachen verwendet wird, die besagte Zufallszahl abzuleiten und die andere der besagten Vielfachen verwendet wird, den besagten geheimen Wert für nachfolgende Verwendung zu ändern.

7. Verfahren nach Anspruch 2 oder Anspruch 3, wobei die besagte Ausgabe der besagten Hash-Funktion (24) als eine Koordinate eines Punkts an einer elliptischen Kurve vor Verwendung als besagte Eingabe validiert wird.

8. Verfahren nach Anspruch 7, wobei eine weitere Koordinate des besagten Punkts aus der besagten einen Koordinate zum Einschluss als die besagte eine Eingabe erlangt wird.

9. Verfahren nach Anspruch 8, wobei die besagte andere Eingabe eine Darstellung eines Punkts einer elliptischen Kurve ist.

10. Verfahren nach Anspruch 6, wobei die besagte Zufallszahl von der besagten einen skalaren Vielfachen abgeleitet wird, indem eine Koordinate eines Punkts, der durch die besagte eine skalare Vielfache repräsentiert ist und Kürzen der besagten einen Koordinate auf eine Bitfolge zur Verwendung als die besagte Zufallszahl selektiert wird.

11. Verfahren nach Anspruch 10, wobei die besagte eine Koordinate ungefähr eine Hälfte der Länge einer Darstellung von einer Punktdarstellung einer elliptischen Kurve gekürzt wird.

12. Verfahren nach Anspruch 6, wobei die besagte Zufallszahl von der besagten skalaren Vielfachen abgeleitet wird, indem eine Koordinate des besagten Punkts, der durch die besagte skalare Vielfache repräsentiert ist, selektiert wird und die besagte eine

Koordinate einer Hash-Code-Anwendung unterzogen wird, um eine Bitfolge zur Verwendung als die besagte Zufallszahl bereitzustellen.

13. Verfahren nach Anspruch 1, wobei die besagte zumindest eine der besagten Eingaben (16, 18) ausgewählt wird einer kanonischen Form zu sein.

14. Verfahren nach einem beliebigen der Ansprüche 1 bis 9, wobei die besagte Ausgabe (20) eine Einwegfunktion durchläuft, um eine Bitfolge zur Verwendung als eine Zufallszahl zu erlangen.

15. Verfahren nach Anspruch 14, wobei die besagte Einwegfunktion eine Hash-Funktion ist.

16. Verfahren nach Anspruch 1, wobei das Erlangen einer der Eingaben das Erlangen eines Ergebnisses einer Hash-Funktion einschließt, die an der besagten einen Eingabe ausgeführt wird.

17. Zufallszahlgenerator (10) für elliptische Kurven, umfassend:

> Mittel zum Generieren eines Eingabepaars (16, 18), wobei jede der besagten Eingaben (16, 18) repräsentativ für zumindest eine Koordinate der jeweiligen Punkte eines Punktepaars einer elliptische Kurve ist und, wobei zumindest eine der besagten Eingaben (16, 18) auf eine Weise generiert wird, die sicherstellt, dass ein Punkt des Punktepaars nicht als eine bekannte Vielfache des anderen Punkts des Punktepaars gewählt worden wäre; und,
> eine Recheneinheit (12) zum Ausführen elliptischer Kurvenoperationen an den besagten Eingaben (16, 18), wobei die besagte Recheneinheit (12) eine Ausgabe (20) aufweist, die auf den Ergebnissen der besagten elliptischen Operationen beruht, wobei die besagte Ausgabe (20) eine Zufallszahl zur Verwendung in einer kryptografischen Operation (22) repräsentiert.

18. Zufallszahlgenerator (10) für elliptische Kurven nach Anspruch 17, wobei das besagte Mittel zum Generieren der besagten Eingaben (16, 18) einen Einwegfunktion einschließt und zumindest eine der besagten Eingaben (16, 18) aus einer Ausgabe einer Einwegfunktion abgeleitet ist.

19. Zufallszahlgenerator (10) für elliptische Kurven nach Anspruch 18, wobei die besagte Einwegfunktion eine Hash-Funktion ist.

20. Zufallszahlgenerator (10) für elliptische Kurven nach Anspruch 17, wobei eine der besagten Eingaben (16, 18) aus einer Ausgabe einer Hash-Funktion erlangt wird und die andere der besagten Eingaben (16, 18) der besagten Hash-Funktion als eine Eingabe bereitgestellt wird.

Revendications

1. Procédé d'utilisation d'un générateur de nombre aléatoire par courbe elliptique (10) comprenant une unité arithmétique (12) afin de réaliser des opérations sur une courbe elliptique pour calculer un nombre aléatoire à utiliser dans une opération cryptographique (22), ladite méthode comprenant les étapes qui consistent à :

> obtenir une paire d'entrées (16, 18), chaque entrée (16,18) étant représentative d'au moins une coordonnée des coordonnées respectives d'une paire de points de la courbe elliptique et au moins l'une desdites entrées (16, 18) étant obtenue d'une manière permettant de garantir qu'un point de la paire de points n'aurait pas été choisi comme multiple connu de l'autre point de la paire de points ;
> fournir ladite paire d'entrées (16, 18) en tant qu'entrées à ladite unité arithmétique (12) ;
> réaliser des opérations de courbe elliptique sélectionnées sur lesdites entrées afin d'obtenir une sortie (20) ; et
> utiliser ladite sortie (20) en tant qu'un nombre aléatoire dans l'opération cryptographique (22).

2. Procédé selon la revendication 1, ladite une desdites entrées (18) étant obtenue à partir d'une sortie de la fonction de hachage (24).

3. Procédé selon la revendication 2, l'autre (16) desdites entrées étant utilisée en tant qu'entrée à ladite fonction de hachage (24).

4. Procédé selon la revendication 2, l'autre (16) desdites entrées étant obtenue à partir d'une sortie de la fonction de hachage.

5. Procédé selon l'une quelconque des revendications 1 à 4, ledit générateur de nombre aléatoire (10) ayant une valeur secrète, laquelle est utilisée pour calculer des multiples scalaires desdits points représentés par lesdites entrées (16, 18).

6. Procédé selon la revendication 5, l'un desdits multiples scalaires étant utilisé pour dériver ledit nombre aléatoire et l'autre desdits multiples scalaires étant utilisé pour changer ladite valeur secrète pour un usage ultérieur.

7. Procédé selon la revendication 2 ou la revendication 3, ladite sortie de ladite fonction de hachage (24) étant validée en tant que coordonnée d'un point sur une courbe elliptique avant l'utilisation de ladite entrée.

8. Procédé selon la revendication 7, une autre coordonnée dudit point étant obtenue à partir de ladite une coordonnées en vue d'une inclusion en tant que ladite une entrée.

9. Procédé selon la revendication 8, ladite autre entrée étant une représentation d'un point de la courbe elliptique.

10. Procédé selon la revendication 6, ledit nombre aléatoire étant dérivé dudit un multiple scalaire en sélectionnant une coordonnée d'un point représentée par ledit un multiple scalaire et en tronquant ladite une coordonnée en une chaîne binaire à utiliser en tant que ledit nombre aléatoire.

11. Procédé selon la revendication 10, ladite une coordonnée étant tronquée dans l'ordre d'une moitié de la longueur d'une représentation de la représentation d'un point de la courbe elliptique.

12. Procédé selon la revendication 6, ledit nombre aléatoire étant dérivé dudit un multiple scalaire en sélectionnant une coordonnée dudit point représentée par ledit multiple scalaire et en hachant ladite une coordonnée afin de fournir une chaîne binaire à utiliser en tant que ledit nombre aléatoire.

13. Procédé selon la revendication 1, ladite au moins une desdites entrées (16, 18) étant choisie de sorte à être de forme canonique.

14. Procédé selon l'une quelconque des revendications 1 à 9, ladite sortie (20) étant passée à travers une fonction unidirectionnelle afin d'obtenir une chaîne binaire à utiliser en tant qu'un nombre aléatoire.

15. Procédé selon la revendication 14, ladite une fonction unidirectionnelle étant une fonction de hachage.

16. Procédé selon la revendication 1, l'obtention de l'une des entrées comprenant l'obtention d'un résultat d'une fonction de hachage qui est exécutée sur ladite une entrée.

17. Générateur de nombre aléatoire par courbe elliptique (10) comprenant :

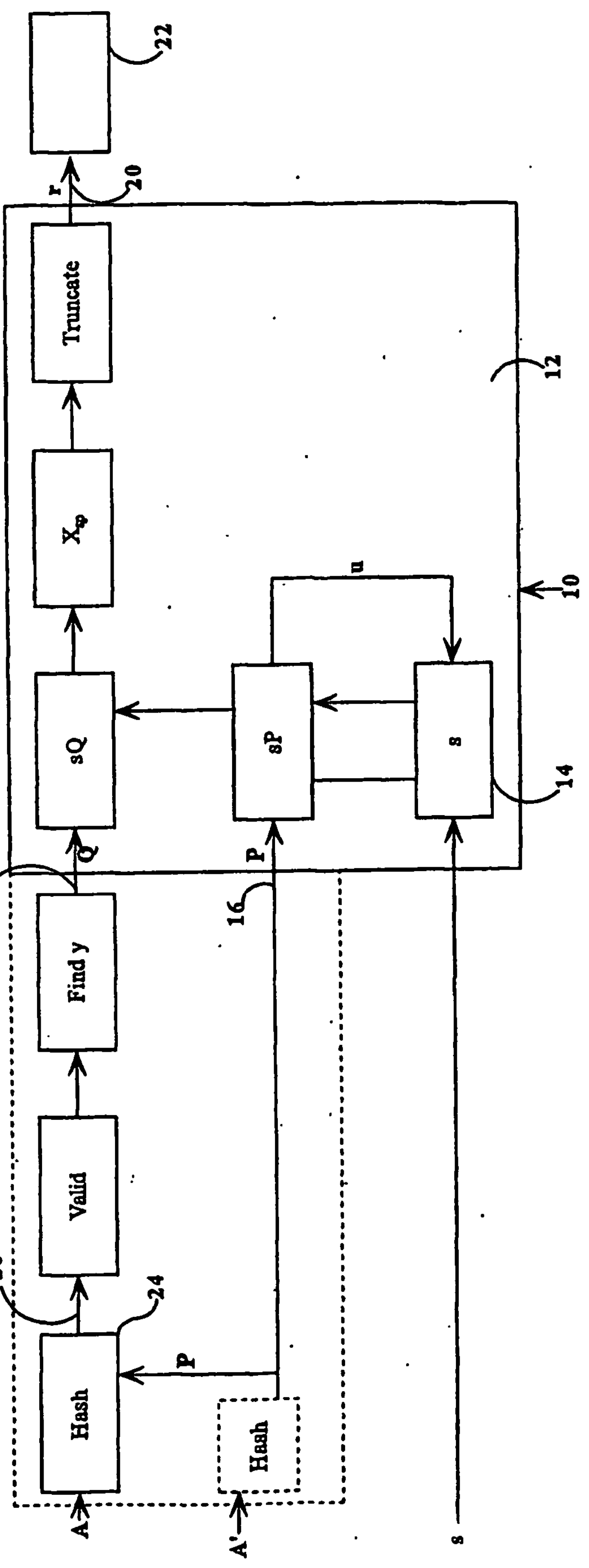
un moyen de génération d'une paire d'entrées (16, 18), chacune des entrées (16, 18) étant représentative d'au moins une coordonnée de coordonnées respectives d'une paire de points de la courbe elliptique, et au moins une desdites entrées (16, 18) étant générée de manière à garantir qu'un point de la paire de points n'aurait pas été choisi en tant que multiple connu de l'autre point de la paire de points ; et,

une unité arithmétique (12) pour l'exécution d'opérations de courbe elliptique sur lesdites entrées (16, 18), ladite unité arithmétique (12) ayant une sortie (20) basée sur les résultats desdites opérations elliptiques, ladite sortie (20) représentant un nombre aléatoire à utiliser dans une opération cryptographique (22).

18. Générateur de nombre aléatoire par courbe elliptique (10) selon la revendication 17, ledit moyen de génération desdites entrées (16, 18) comprenant une fonction unidirectionnelle et au moins une desdites entrées (16, 18) étant dérivée d'une sortie d'une fonction unidirectionnelle.

19. Générateur de nombre aléatoire par courbe elliptique (10) selon la revendication 18, ladite fonction unidirectionnelle étant une fonction de hachage.

20. Générateur de nombre aléatoire par courbe elliptique (10) selon la revendication 17, l'une desdites entrées (16, 18) étant obtenue à partir d'une sortie d'une fonction de hachage et l'autre desdites entrées (16, 18) étant fournie en tant qu'entrée de ladite fonction de hachage.



A
A'
s
Hash
Hash
P
24
26
Valid
Find y
18
16
Q
P
sQ
sP
s
14
u
10
Xsp
Truncate
12
r
20
22

FIGURE 1

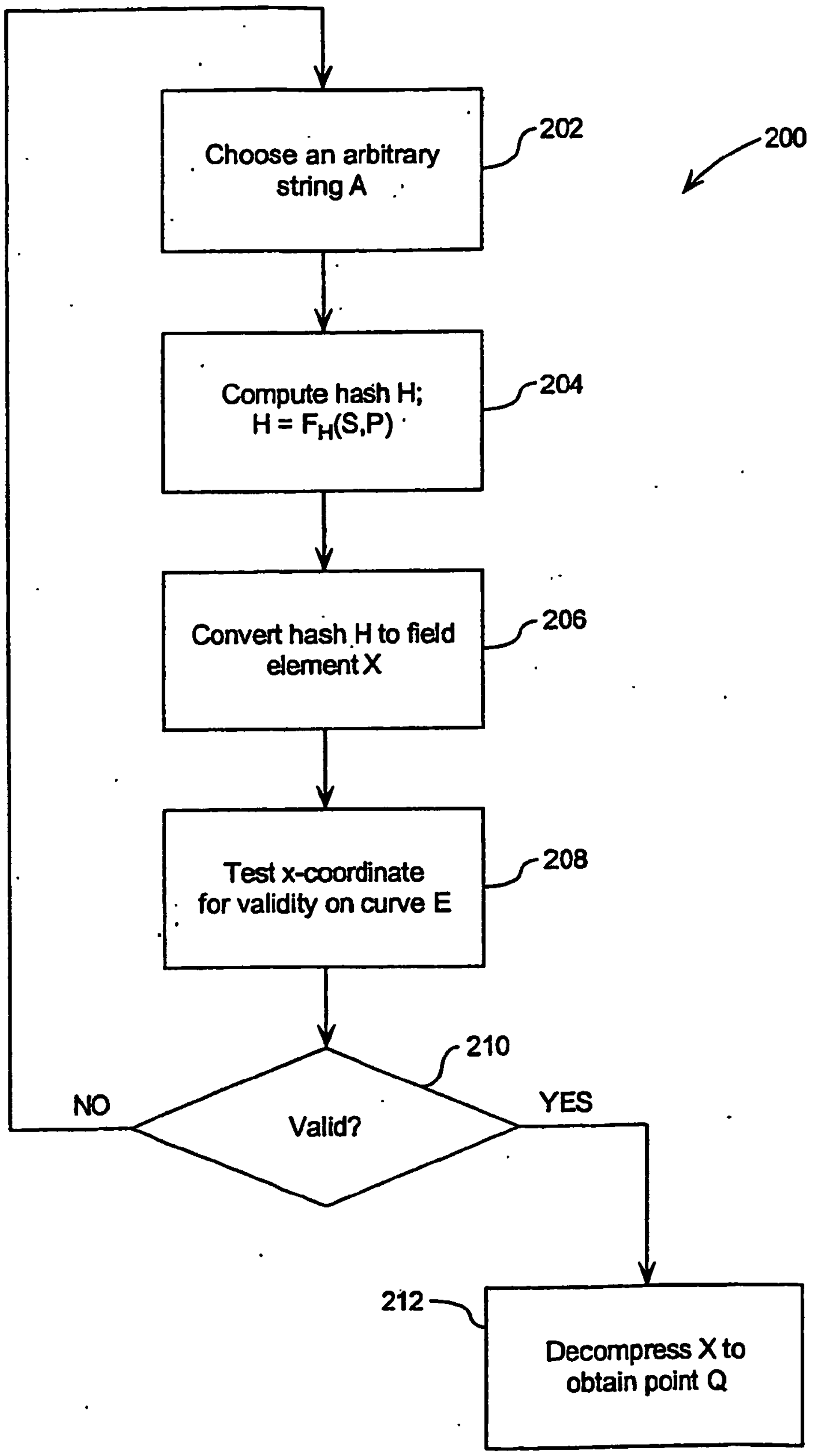

Choose an arbitrary string A
202
200
Compute hash H; $H = F_H(S,P)$
204
Convert hash H to field element X
206
Test x-coordinate for validity on curve E
208
210
NO
Valid?
YES
212
Decompress X to obtain point Q

Figure 2

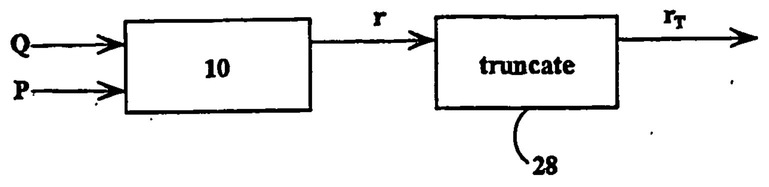


FIGURE 3

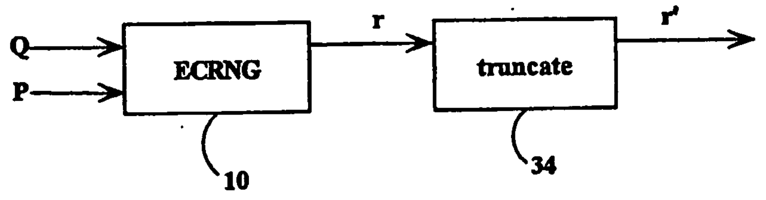


FIGURE 5

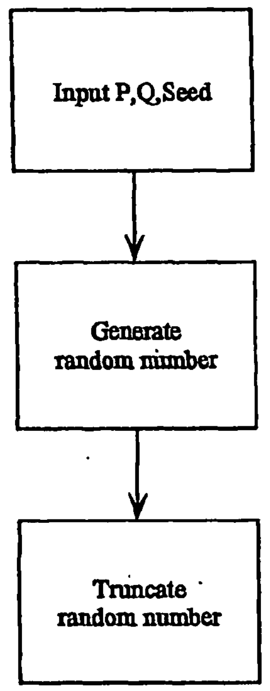


Figure 4

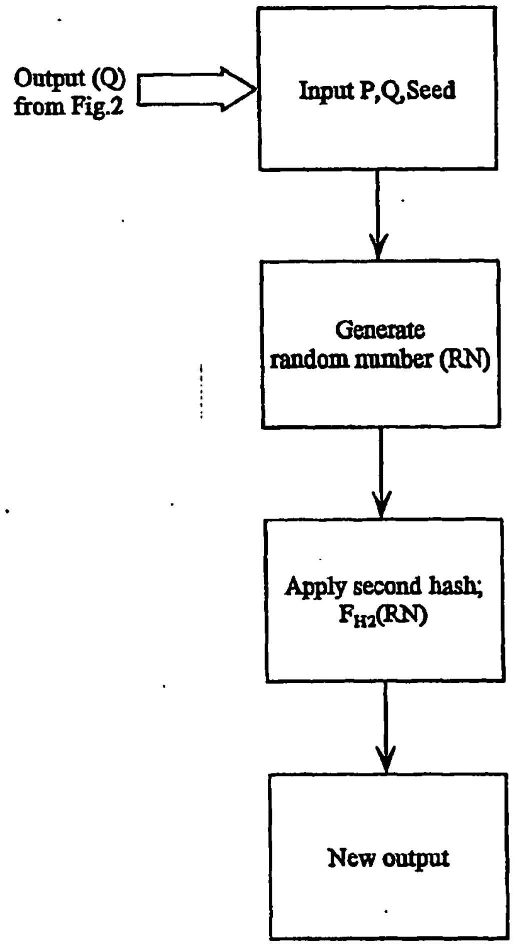


Figure 6

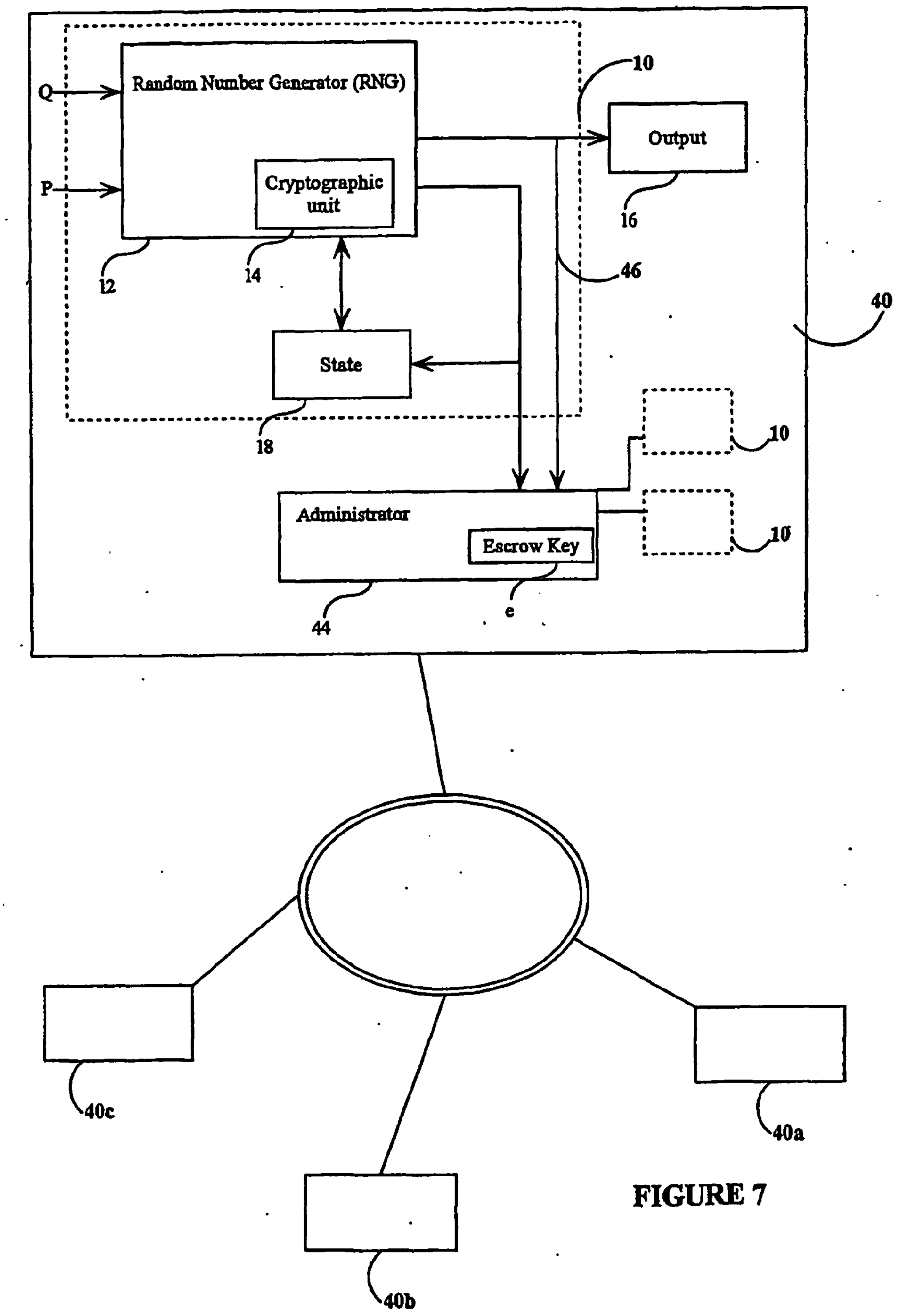

Random Number Generator (RNG)
Q
P
Cryptographic unit
Output
State
Administrator
Escrow Key
10
16
46
12
14
40
18
10
10
44
e
40c
40a
40b

FIGURE 7

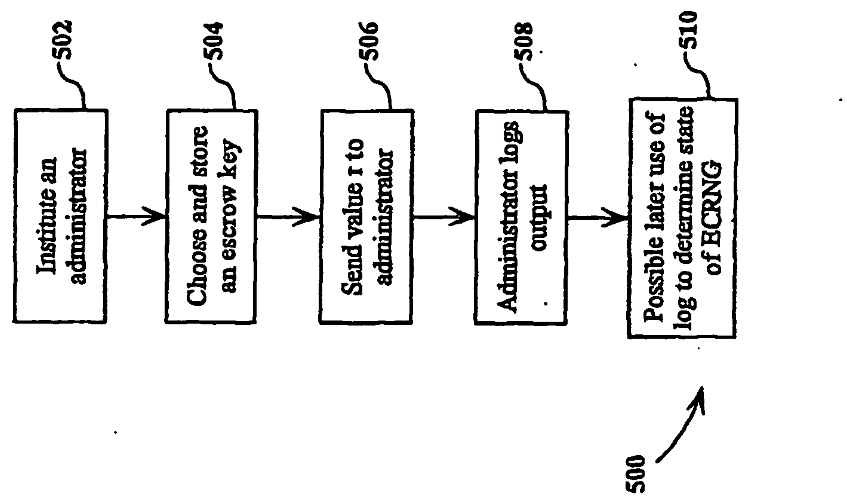

Institute an administrator
502
Choose and store an escrow key
504
Send value r to administrator
506
Administrator logs output
508
Possible later use of log to determine state of ECRNG
510
500

Figure 9

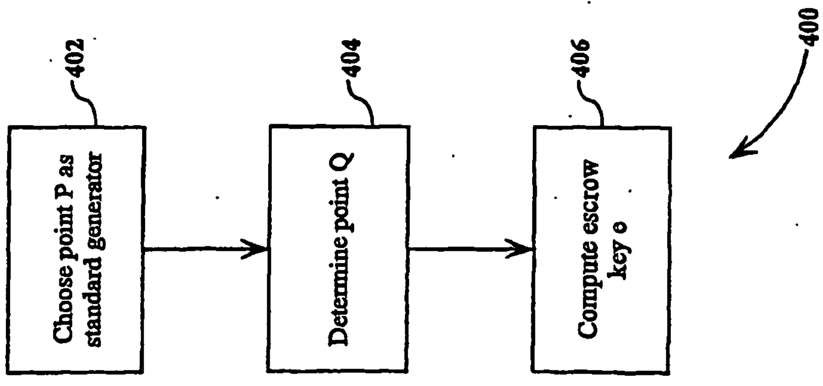


Figure 8