EXHIBIT A

EXHIBIT A

US007372961B2

(12) **United States Patent**
Vanstone et al.

(10) **Patent No.:** **US 7,372,961 B2**
(45) **Date of Patent:** **May 13, 2008**

(54) **METHOD OF PUBLIC KEY GENERATION**

(75) Inventors: **Scott A. Vanstone**, Campbellville (CA);
Ashok Vadekar, Rockwood (CA);
Robert J. Lambert, Cambridge (CA);
Robert P. Gallant, Mississauga (CA);
Daniel R. Brown, Toronto (CA);
Alfred Menezes, West Canada (CA)

(73) Assignee: **Certicom Corp.**, Mississauga (CA)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 563 days.

(21) Appl. No.: **10/025,924**

(22) Filed: **Dec. 26, 2001**

(65) **Prior Publication Data**

US 2002/0090085 A1 Jul. 11, 2002

(30) **Foreign Application Priority Data**

Dec. 27, 2000 (CA) 2329590

(51) **Int. Cl.**
 H04L 9/08 (2006.01)
 G06F 7/58 (2006.01)
(52) **U.S. Cl.** **380/44**; 380/277; 708/250
(58) **Field of Classification Search** 380/44,
380/277; 708/250

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,073,935	A	*	12/1991	Pastor	380/30
6,088,798	A	*	7/2000	Shimbo	713/176
6,195,433	B1	*	2/2001	Vanstone et al.	380/285
6,219,421	B1	*	4/2001	Backal	380/28
6,307,938	B1	*	10/2001	Matyas et al.	380/44
6,327,660	B1	*	12/2001	Patel	713/193
2002/0116527	A1	*	8/2002	Chen et al.	709/245
2003/0084332	A1	*	5/2003	Krasinski et al.	713/200

OTHER PUBLICATIONS

Schneier, Bruce, "Applied Cryptography", 1996, John Wiley &
Sons, Inc., 2nd editiion, pp. 483-490.*
Nel et al., "Generation of Keys for use with the Digital Signature
Standard (DSS)", 1993, IEEE, pp. 6-10.*

* cited by examiner

Primary Examiner—Emmanuel L. Moise
Assistant Examiner—Michael Pyzocha
(74) *Attorney, Agent, or Firm*—Blake, Cassels & Graydon
LLP; John R. S. Orange; Brett J. Slaney

(57) **ABSTRACT**
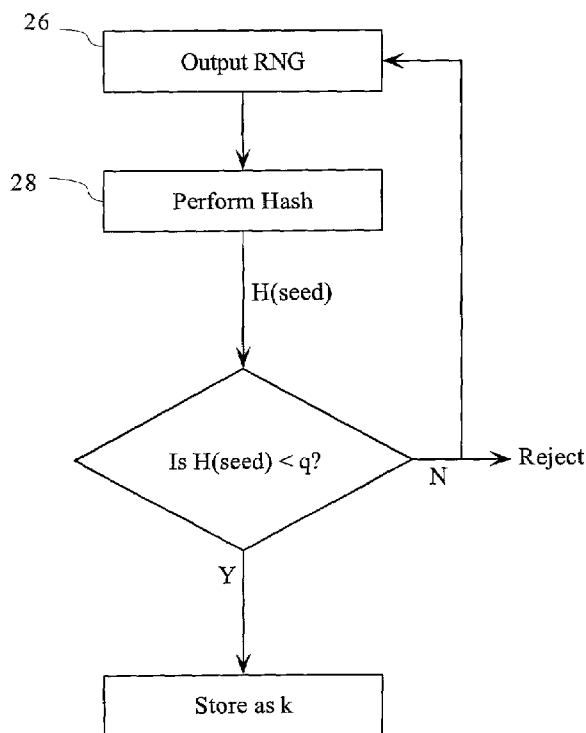
A potential bias in the generation or a private key is avoided
by selecting the key and comparing it against the system
parameters. If a predetermined condition is attained it is
accepted. If not it is rejected and a new key is generated.

30 Claims, 7 Drawing Sheets

Fig. 1

26 ── Output RNG

28 ── Perform Hash

H(seed)

Is H(seed) < q?   N ──→ Reject

Y

Store as k

Fig. 2

Fig. 3

```
        ┌─────────────────────┐
        │     Output RNG      │
        └─────────────────────┘
                  │
                  ▼
        ┌─────────────────────┐
        │     Store Value     │◄──────────┐
        └─────────────────────┘           │
                  │                        │  Increment
                  ▼                        │
        ┌─────────────────────┐           │
        │    Perform Hash     │───────────┘
        └─────────────────────┘
                  │
                  ▼
        ┌─────────────────────┐
        │      Combine        │
        │ H(seed) ‖ H(seed +1)│
        └─────────────────────┘
                  │
                  ▼
        ┌─────────────────────┐
        │  Reject Surplus Bits│
        └─────────────────────┘
                  │
                  ▼
              ╱───────╲
            ╱           ╲
          ╱               ╲
        ╱  Is H(seed) < q?  ╲────── N ──────►  Reject
          ╲               ╱
            ╲           ╱
              ╲───────╱
                  │
                  Y
                  │
                  ▼
        ┌─────────────────────┐
        │     Store as k      │
        └─────────────────────┘
```

Fig. 4

Fig. 5

26 — Output RNG

28 — Apply Hash

Select *l* bits

Increment *l* bit window

Is H(seed) < q?   N → Reject

Y

Store as k
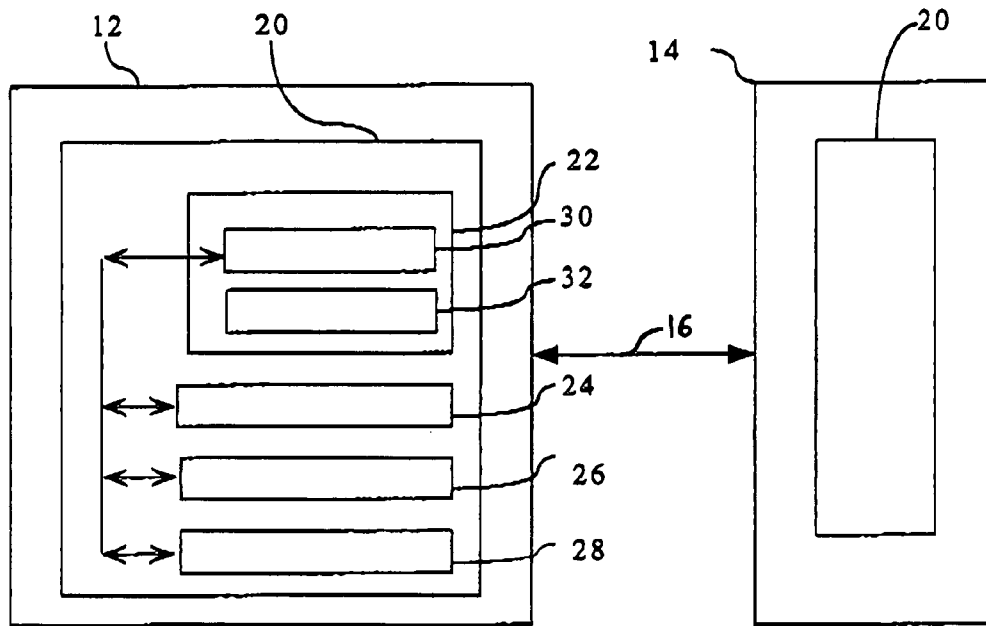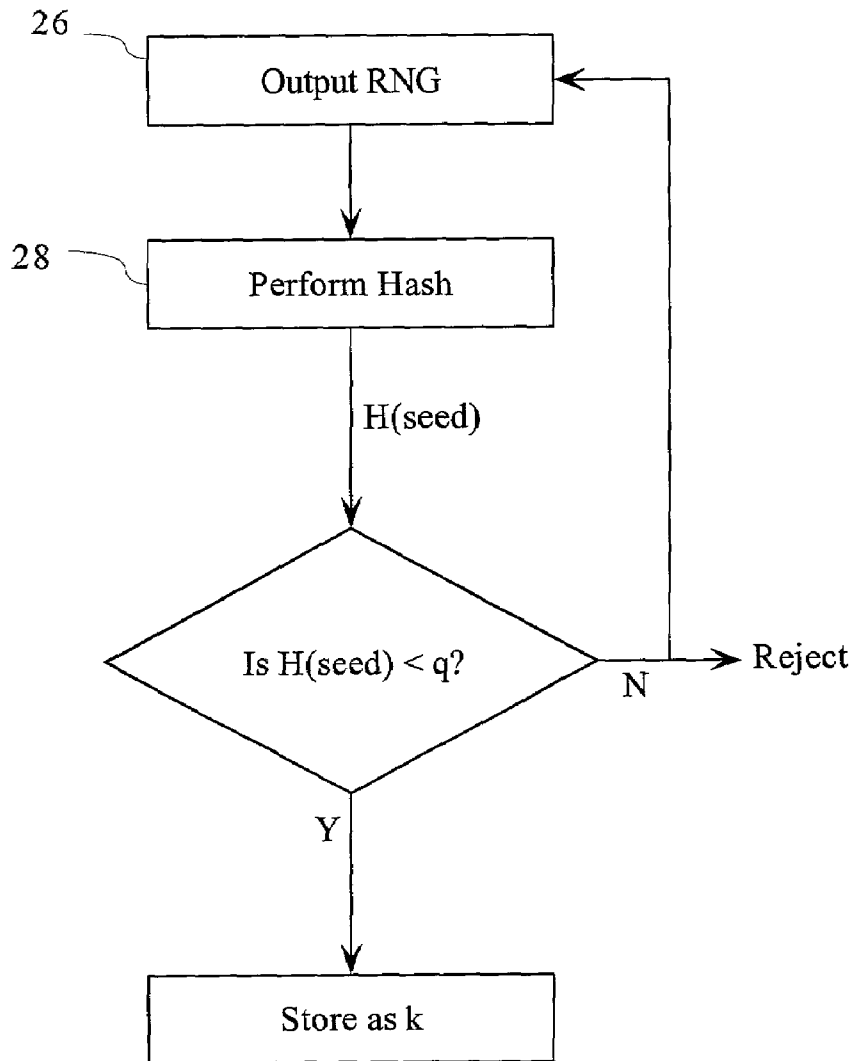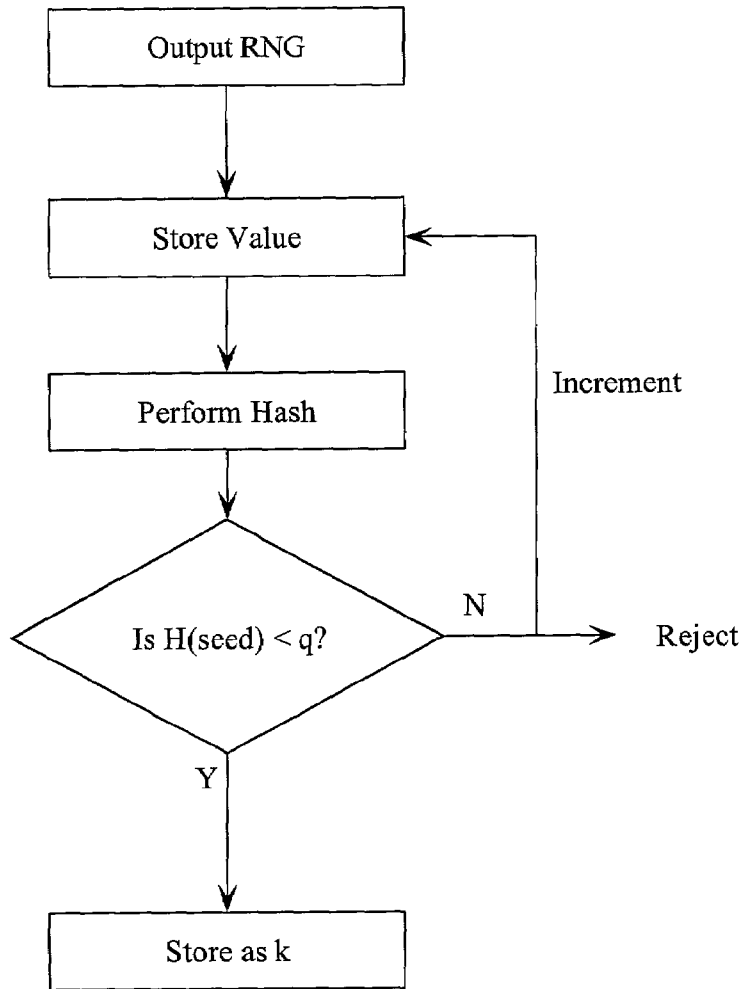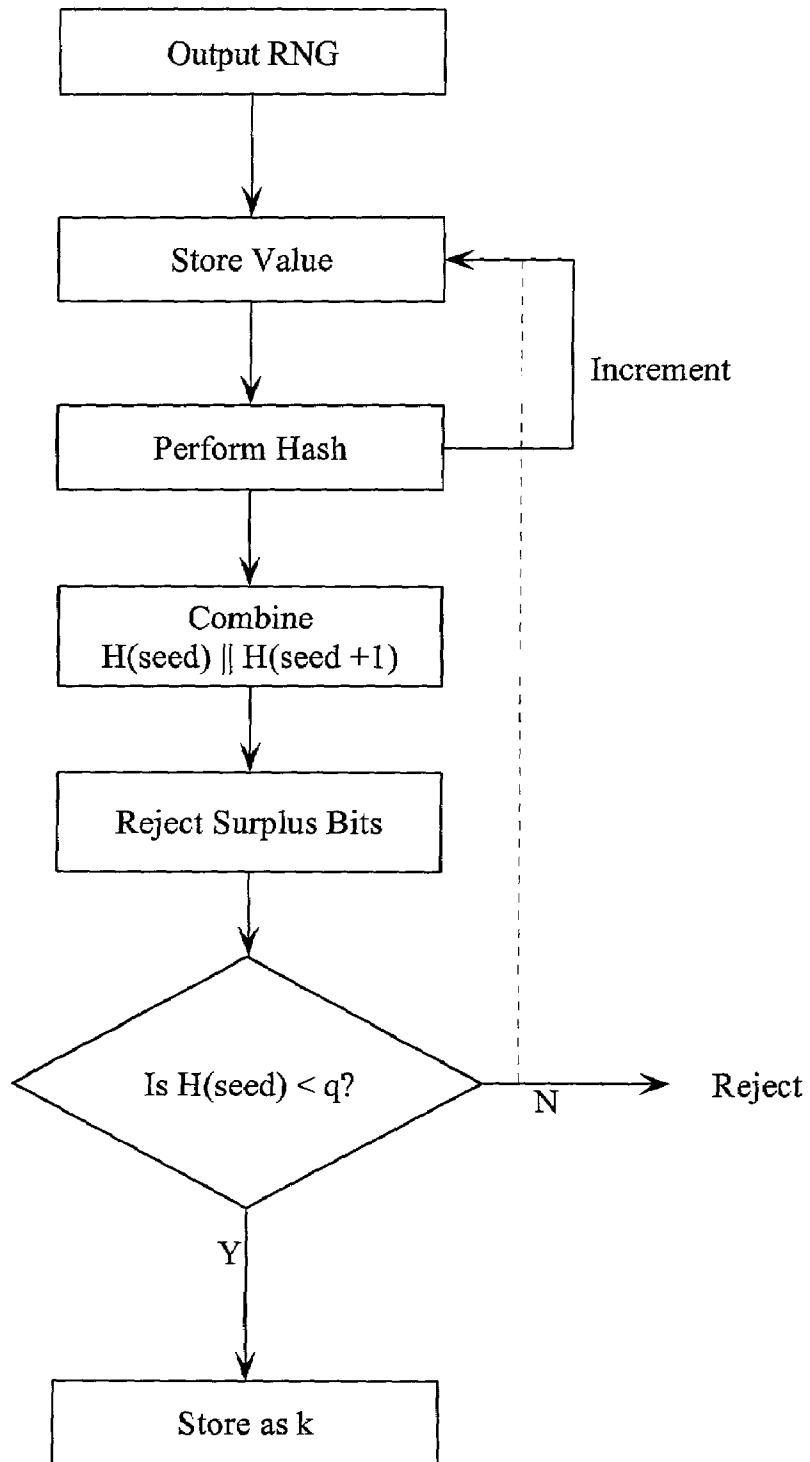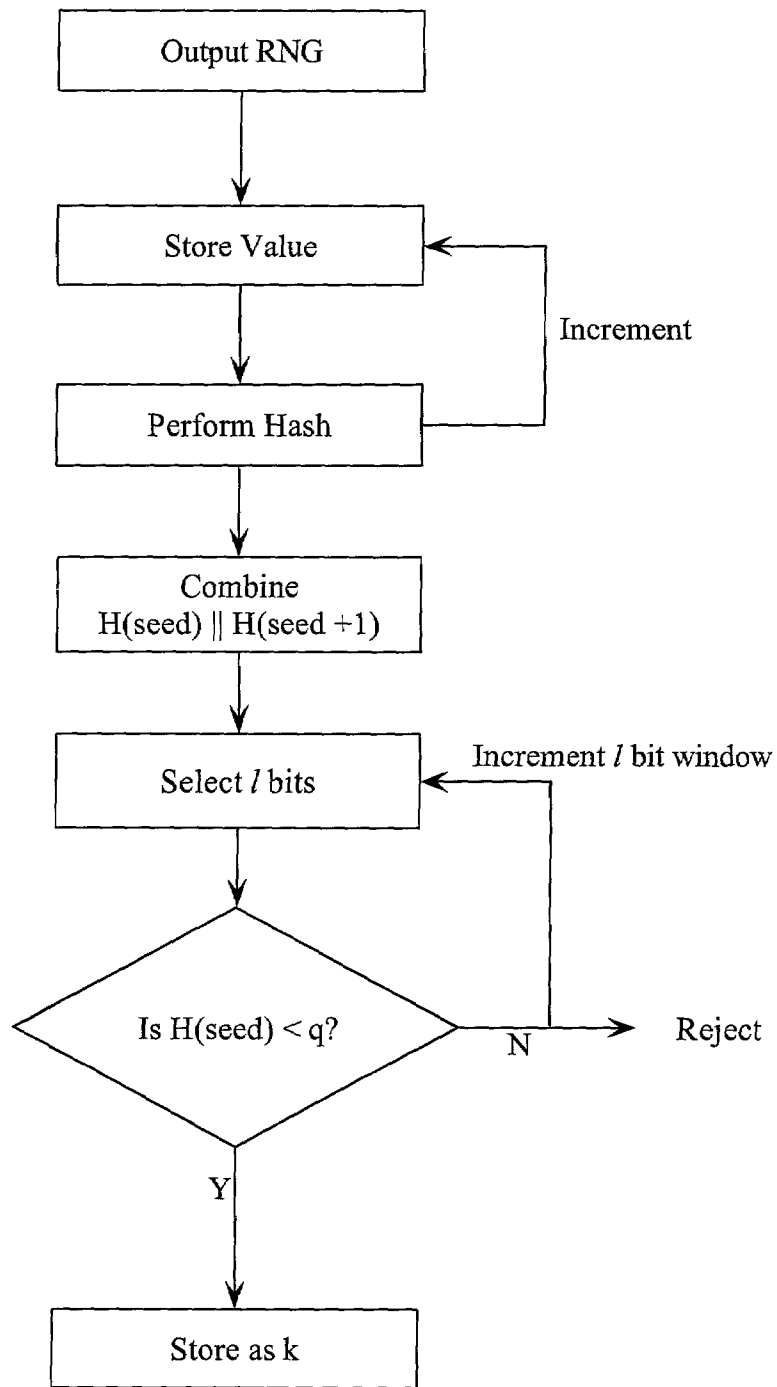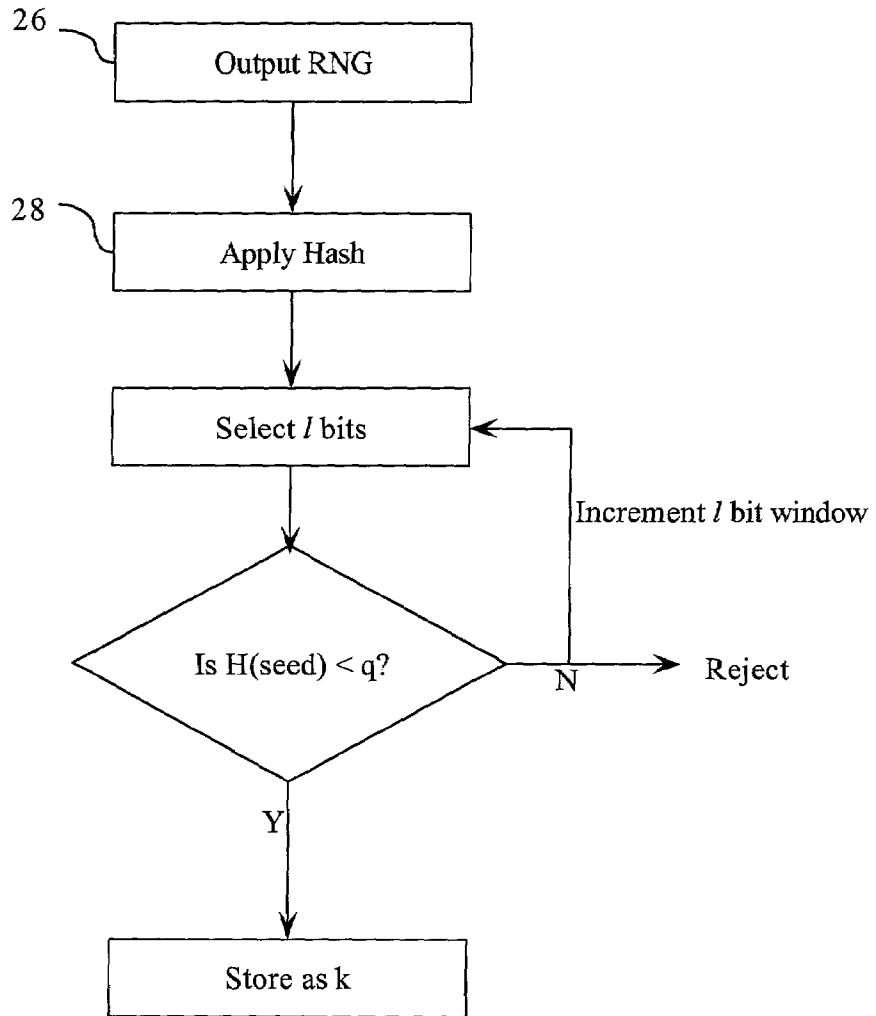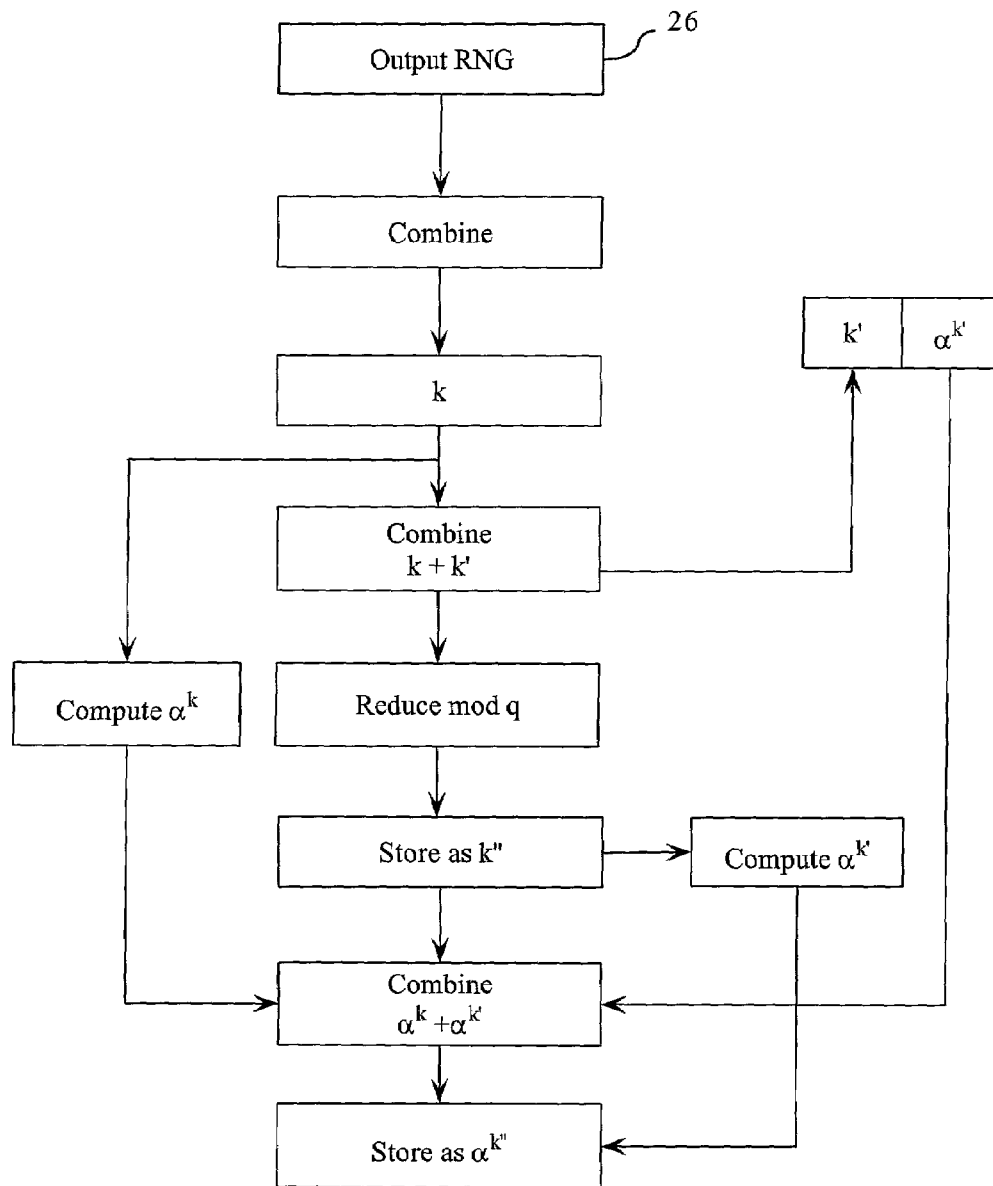
Fig. 6

Fig. 7

METHOD OF PUBLIC KEY GENERATION

FIELD OF THE INVENTION

The present invention relates to public key cryptosystems 5 and more particularly to key generation within such systems.

BACKGROUND OF THE INVENTION

The basic structure of a public key cryptosystem is well 10 known and has become ubiquitous with security in data communication systems. Such systems use a private key k and a corresponding public key α^k where α is a generator of the group. Thus one party may encrypt a message m with the intended recipients public key and the recipient may apply 15 his private key to decrypt it.

Similarly, the cryptosystems may be used for key agreement protocols where each party exponentiates the other party's public key with their own private key. Thus party A will take B's public key α^b and exponentiate it with A's 20 private key a to obtain a session key α^{ab}. Similarly, B will take A's public key α^a and exponentiate it with B's private key b to obtain the same session key α^{ab}. Thereafter data may be transferred using a symmetric key protocol utilizing the common session key. 25

Public key cryptosystems may also be used to sign messages to authenticate the author and/or the contents. In this case the sender will sign a message using his private key and a recipient can verify the message by applying the public key of the sender. If the received message and the recovered 30 message correspond then the authenticity is verified.

The public key cryptosystems rely on the intractability of the discrete log problem in finite field arithmetic, that is even when the generator α and public key is known, it is 35 computationally infeasible to obtain the corresponding private key. The security of such systems does therefore depend on the private key remaining secret. To mitigate the opportunity of disclosing the private key, protocols have been developed that use a pair of private keys and corresponding 40 public keys, referred to as long term and short term or ephemeral key pairs respectively. The ephemeral private key is generated at the start of each session between a pair of correspondents, usually by a random number generator. The corresponding, ephemeral public key is generated and the 45 resultant key pair used in one of the possible operations described above. The long-term public key is utilized to authenticate the correspondent through an appropriate protocol. Once the session is terminated, the ephemeral key is securely discarded and a new ephemeral key generated for 50 a new session.

Some of the more popular protocols for signature are the ElGamal family of signature schemes such as the Digital Signature Algorithm or DSA. The DSA algorithm utilizes both long term and ephemeral keys to generate a signature 55 of the message. The DSA domain parameters are preselected. They consist of a prime number p of a predetermined length, by way of example 1024 bits; a prime number q of a predetermined bit length, by way of example 160 bits, where q divides p–1; a generator α lying between 2 and p–1 60 and which satisfies the condition ($\alpha^q \bmod p$)=1, and; a cryptographic hash function H, such as SHA–1.

The DSA requires the signatory to select an ephemeral key k lying between 1 and q–1. A first signature component r is generated from the generator α such that r=($\alpha^k \bmod p$) 65 mod q, A second signature component s is generated such that s=k^{-1}(H(m)+dr) mod q, and d is the long term private

key of the signatory. The signature on the message m is (r,s). The signature may be verified by computing

$$H(m),$$

$$u_1 = s^{-1} H(m) \bmod q$$

$$u_2 = s^{-1} r \bmod q$$

v=$\alpha^{u_1} \beta^{u_2}$mod p, where $\beta = \alpha^d$ mod p is the long term public key of the signatory and finally verifying that r=v mod q. The use of both the ephemeral and long-term keys in the signature binds the identity of the signatory to the ephemeral key but does not render the long-term key vulnerable.

A similar signature protocol known as ECDSA may be used for elliptic curve cryptosystems. In this protocol k is selected in the interval 1 to n–1 where n is an l bit prime. The signature component r is generated by converting the x coordinate of the public key kP, where P is the seed point on the curve, to an integer mod n, i.e. r=x_{kp} mod n. The component s=k^{-1}(H(m)+dr) mod n and the signature on the message m is (r,s).

It will be apparent in ElGamal signature schemes such as the DSA and ECDSA, that if an ephemeral key k and the associated message m and signature (r,s) is obtained it may be used to yield the long term private key d and thereafter each of the ephemeral keys k can be obtained. Neither the DSA nor the ECDSA inherently disclose any information about the public key k. They both require the selection of k to be performed by a random number generator and it will therefore have a uniform distribution throughout the defined interval. However the implementation of the DSA may be done in such a way as to inadvertently introduce a bias in to the selection of k. This small bias may be exploited to extract a value of the private key d and thereafter render the security of the system vulnerable. One such implementation is the DSS mandated by the National Institute of Standards and Technology (NIST) FIPS 186-2 Standard. The DSS stipulates the manner in which an integer is to be selected for use as a private key. A seed value, SV, is generated from a random number generator which is then hashed by a SHA–1 hash function to yield a bit string of predetermined length, typically 160 bits. The bit string represents an integer between 0 and 2^{160}–1. However this integer could be greater than the prime q and so the DSS requires the reduction of the integer mod q, i.e. k=SHA–1(seed) mod q.

Accordingly the algorithm for selecting k may be expressed as:

if SHA–1(seed)$\geq$q then k$\leftarrow$SHA–1(seed)–q else
    k$\leftarrow$SHA–1(seed).

With this algorithm it is to be expected that more values will lie in the first interval than the second and therefore there is a potential bias in the selection of k.

Recent work by Daniel Bleichenbacher suggests that the modular reduction to obtain k introduces sufficient bias in to the selection of k that an examination of 2^{22} signatures could yield the private key d in 2^{64} steps using 2^{40} memory units. This suggests that there is a need for the careful selection of the ephemeral key k.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to obviate or mitigate the above disadvantages in the generation of a private key.

In general terms the present invention provides a key generation technique in which any bias is eliminated during the selection of the key.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described by way of example only with reference to the accompanying drawings in which:

FIG. **1** is a schematic representation of a data communication system;

FIG. **2** is a flow chart showing a first embodiment of key generation;

FIG. **3** is a flow chart showing a second embodiment;

FIG. **4** is a flow chart showing a third embodiment;

FIG. **5** is a flow chart showing a fourth embodiment;

FIG. **6** is a flow chart showing a fifth embodiment; and

FIG. **7** is a flow chart showing a sixth embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring, therefore to FIG. **1**, a data communication system **10** includes a pair of correspondents **12**, **14** connected by a communication link **16**. The link **16** may be a dedicated link, a multipurpose link such as a telephone connection or a wireless link depending on the particular applications. Similarly, the correspondents **12**, **14** may be computer terminals, point-of-sale devices, automated teller machines, constrained devices such as PDA's, cellphones, pagers or any other device enabled For communication over a link **16**.

Each of the correspondents **12**, **14** includes a secure cryptographic function **20** including a secure memory **22**, an arithmetic processor **24** for performing finite field operations, a random number generator **26** and a cryptographic hash function **28** for performing a secure cryptographic hash such as SHA–1. The output of the function **28** will be a bit string of predetermined length, typically 160 bits although other lengths such as 256, 384 or 512 are being used more frequently. It will be appreciated that each of these functions is controlled by a processor executing instructions to provide functionality and inter-operability as is well known in the art.

The secure memory **22** includes a register **30** for storing a long-term private key, d, and a register **32** for storing an ephemeral private key k. The contents of the registers **30**, **32** may be retrieved for use by the processor **24** for performing signatures, key exchange and key transport functions in accordance with the particular protocols to be executed under control of the processor.

The long term private key, d, is generated and embedded at the time of manufacture or initialization of the cryptographic function and has a corresponding long-term public key α^d. The long-term public key α^d is stored in the memory **22** and is generally made available to other correspondents of the system **10**.

The ephemeral key, k, is generated at each signature or other cryptographic exchange by one of the routines disclosed below with reference to FIGS. **2** to **9**. Once the key, k, and corresponding public key α^k is generated, it is stored in the register **32** for use in the cryptographic protocol, such as the DSA or ECDSA described above.

Referring, therefore, to FIG. **2**, a first method of generating a key, k, originates by obtaining a seed value (SV) from the random number generator **26**. For the purposes of an example, it will be assumed that the cryptographic

function is performed over a group of order q, where q is a prime represented as a bit string of predetermined length L. By way of example only it will be assumed that the length L is 160 bits, although, of course, other orders of the field may be used.

To provide a value of k of the appropriate order, the hash function **28** has an L bit output, e.g. a 160 bit output. The bit string generated by the random number generator **26** is greater than L bits and is therefore hashed by the function **28** to produce an output H(seed) of L bits.

The resultant output H(seed) is tested against the value of q and a decision made based on the relative values. If H(seed)<q then it is accepted for use as k. If not, the value is rejected and the random number generator is conditioned to generate a new value which is again hashed by the function **28** and tested. This loop continues until a satisfactory value is obtained.

A further embodiment is shown in FIG. **3**. In this embodiment, the output of the random number generator **26** is hashed by hash function **28** as before and tested against the value of q. If the H(seed) value is not accepted, the output of the random number generator **26** is incremented by a deterministic function and rehashed by function **28**.

The resultant value H(seed) is again tested and the procedure repeated until a satisfactory value of k is obtained.

The output may be incremented by adding a particular value to the seed value at each iteration, or may be incremented by applying a non-linear deterministic function to the seed value. For example, the output may be incremented by applying the function $f(\text{seed})=a.\text{seed}^2+b \bmod 2^{160}$, where a and b are integer constants.

A further embodiment is shown in FIG. **4** which has particular applicability to an elliptic curve cryptosystem. By way of example it will be assumed that a 163 bit string is required and that the output of the hash function **28** is 160 bits.

The random number generator **26** generates a seed value SV which is processed by the hash function **28** to obtain a first output H(seed).

The seed value SV is incremented by a selected function to provide a seed value SV+ which is further processed by the hash function **28** to provide a second output H(seed+).

The two outputs are then combined, typically by cocatenation, to produce a 320 bit string H(seed)//H(seed+). The excess bits, in this case **157** are rejected and the resultant value tested against the value of q. If the resultant value is less than q, it is accepted as the key k, if not the value is rejected.

Upon rejection, the random number generator may generate a new value as disclosed in FIG. **2** or may increment the seed value as disclosed in FIG. **3**.

A further embodiment is shown in FIG. **5** which is similar to that of FIG. **4**. In the embodiment of FIG. **5**, the selection of the required L bit string is obtained by applying a L-bit wide masking window to the combined bit string.

This is tested against the value of q and if acceptable is used as the value of k. If it is not acceptable it is rejected and the L bit window incremented along the combined bit string to obtain a new value.

The values are tested and the window incremented until a satisfactory value is obtained.

A similar procedure may be used directly on an extended output of the hash function **28** as shown in FIG. **6** by applying a window to obtain the required L bit string. The bit string is tested against q and the window incremented until a satisfactory value of k is obtained.

5

As shown in FIG. **7**, the value of k may be generated by utilizing a low Hamming weight integer obtained by combining the output of the random number generator **26** to facilitate computation of an intermediate public key α^k. The integer is masked by combination with predetermined pre-computed value k' to obtain the requisite Hamming weight for security. Such a procedure is disclosed in copending Canadian application 2,217,925. This procedure is modified to generate the low Hamming weight integer k as a bit string greater than L, for example a 180 bit string. The masking value k' is distributed throughout the 180 bit string and the resultant value reduced mod q to obtain a 163 bit value k". Note that the value $\alpha^{k''}$ can be efficiently computed by combining the precomputed value α with the efficiently computable value α^{k}.

A similar technique may be used by relying on multiplicative masking. In this embodiment the value of k is combined with a value β where $\beta = \alpha^u$. The value of u is a secret value that is used to mask the low Hamming weight of k. Again, the values of u and the low Hamming weight number k can be chosen to have bit lengths greater than L, for example, bit lengths of 180. The resultant value is $k'' = u^k$ mod q. It will be appreciated that $\alpha^{k''}$ can be efficiently computed since $\beta = \alpha^u$ is precomputed, and since k has low Hamming weight.

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 spirit and scope of the invention as outlined in the claims appended hereto.

The embodiment of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A method of generating a key k for use in a crypto-graphic function performed over a group of order q, said method including the steps of:

generating a seed value SV from a random number generator;

performing a hash function H( ) on said seed value SV to provide an output H(SV);

determining whether said output H(SV) is less than said order q prior to reducing mod q;

accepting said output H(SV) for use as said key k if the value of said output H(SV) is less than said order q;

rejecting said output H(SV) as said key if said value is not less than said order q;

if said output H(SV) is rejected, repeating said method; and

if said output H(SV) is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output H(SV).

2. The method of claim **1** wherein another seed value is generated by said random number generator if said output is rejected.

3. The method of claim **1** wherein the step of accepting said output as a key includes a further step of storing said key.

4. The method of claim **1** wherein said key is used for generation of a public key.

5. The method of claim **1** wherein said order q is a prime number represented by a bit string of predetermined length L.

6. The method of claim **5** wherein said output from said hash function is a bit string of predetermined length L.

7. The method of claim **1** wherein if said output is rejected, said output is incremented by a deterministic function and a hash function is performed on said incre-

6

mented output to produce a new output; a determination being made as to whether said new output is acceptable as a key.

8. The method of claim **7**, wherein said step of incrementing includes a further step of adding a particular value to said seed value.

9. A method of generating a key k for use in a crypto-graphic function performed over a group of order q, said method including the steps of:

generating a seed value SV from a random number generator;

performing a hash function H( ) on said seed number SV to provide a first output H(SV);

incrementing said seed value SV by a predetermined function f( ) and performing said hash function H( ) on said incremented seed value to provide a second output H(f(SV));

combining said first output H(SV) and second output H(f(SV)) to produce a new output;

determining whether said new output has a value less than said order q prior to reducing mod q;

accepting said new output for use as said key k if said new output has a value less than said order q;

rejecting said new output as said key k if said value is not less than said order q;

if said new output is rejected, repeating said method, and

if said new output is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said new output.

10. The method of claim **9** wherein upon rejection of said new output a new seed value is generated by said random number generator.

11. The method of claim **9** wherein upon rejection of said new output said seed value is incremented by a predetermined function and revised values for said first output and said second output are obtained.

12. The method of claim **9** wherein a bit string greater than a predetermined length L is obtained and an L bit string selected therefrom for comparison with said order q.

13. The method of claim **12** wherein upon rejection of said bit string of predetermined length L, a further L bit string is selected.

14. The method of claim **9** wherein said step of combining said first and second outputs includes a further step of rejecting excess bits such that said new output is a bit string of length L.

15. A computer readable medium comprising computer executable instructions for generating a key k for use in a cryptographic function performed over a group of order q, said instructions including instructions for:

generating a seed value SV from a random number generator;

performing a hash function H( ) on said seed value SV to provide an output H(SV);

determining whether said output H(SV) is less than said order q prior to reducing mod q;

accepting said output H(SV) for use as said key k if the value of said output H(SV) is less than said order q;

rejecting said output H(SV) as said key if said value is not less than said order q;

if said output H(SV) is rejected, repeating said method; and

if said output H(SV) is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output H(SV).

16. The computer readable medium of claim **15** wherein another seed value is generated by said random number generator if said output is rejected.

17. The computer readable medium of claim **15** wherein the step of accepting said output as a key includes a further step of storing said key.

18. The computer readable medium of claim **15** wherein said key is used for generation of a public key.

19. The computer readable medium of claim **15** wherein said order q is a prime number represented by a bit string of predetermined length L.

20. The computer readable medium of claim **19** wherein said output from said hash function is a bit string of predetermined length L.

21. The computer readable medium of claim **15** wherein if said output is rejected, said output is incremented by a deterministic function and a hash function is performed on said incremented output to produce a new output; a determination being made as to whether said new output is acceptable as a key.

22. The computer readable medium of claim **21**, wherein said step of incrementing includes a further step of adding a particular value to said seed value.

23. A cryptographic unit configured for generating a key k for use in a cryptographic function performed over a group of order q, said cryptographic unit having access to computer readable instructions and comprises:

an arithmetic processor for:

generating a seed value SV from a random number generator;

performing a hash function H( ) on said seed value SV to provide an output H(SV);

determining whether said output H(SV) is less than said order q prior to reducing mod q;

accepting said output H(SV) for use as said key k if the value of said output H(SV) is less than said order q;

rejecting said output H(SV) as said key if said value is not less than said order q;

if said output H(SV) is rejected, repeating said method; and

if said output H(SV) is accepted, providing said key k for use in performing said cryptographic function, wherein said key k is equal to said output H(SV).

24. The cryptographic unit of claim **23** wherein another seed value is generated by said random number generator if said output is rejected.

25. The cryptographic unit of claim **23** wherein the step of accepting said output as a key includes a further step of storing said key.

26. The cryptographic unit of claim **23** wherein said key is used for generation of a public key.

27. The cryptographic unit of claim **23** wherein said order q is a prime number represented by a bit string of predetermined length L.

28. The cryptographic unit of claim **27** wherein said output from said hash function is a bit string of predetermined length L.

29. The cryptographic unit of claim **23** wherein if said output is rejected, said output is incremented by a deterministic function and a hash function is performed on said incremented output to produce a new output; a determination being made as to whether said new output is acceptable as a key.

30. The cryptographic unit of claim **29**, wherein said step of incrementing includes a further step of adding a particular value to said seed value.

* * * * *

CERTIFICATE OF CORRECTION

PATENT NO.	: 7,372,961 B2	Page 1 of 1
APPLICATION NO.	: 10/025924	
DATED	: May 13, 2008	
INVENTOR(S)	: Scott A. Vanstone et al.	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the drawings, Sheet 7, Fig. 7, reverse the arrow between box "k'" and box "Combine k + k'", delete the box identified as "Combine" and replace with a box identified as "Comb".

In column 1, line 35, delete "public key is known" and replace with "public key are known".

In column 2, line 21, delete "r=x_{kp}" and replace with "r=x_{kP}".

In column 2, line 50, delete "else".

In column 2, line 51, add "else" at the beginning of the line to read "else $k \leftarrow$ SHA-1(seed)."

In column 3, line 61, delete "is generated" and replace with "are generated".

In column 3, line 61, delete "it is stored" and replace with "k is stored".

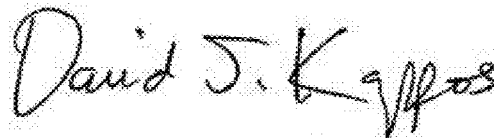In column 4, line 44, delete "cocatenation" and replace with "concatenation".

In column 4, line 44, delete "//" and replace with "11".

In column 4, line 45, delete "**157**" and replace with "157".

In column 5, line 2 and 3, delete "combining" and replace with "combing".

In column 5, line 14, delete "α" and replace with "α^k".

Signed and Sealed this
Twelfth Day of July, 2011

David J. Kappos
Director of the United States Patent and Trademark Office