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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 January 2018 (25.01.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/015464 A1

(51) International Patent Classification:
H04L 9/08 (2006.01) *H04L 9/00* (2006.01)
H04L 9/30 (2006.01)

(21) International Application Number:
PCT/EP2017/068287

(22) International Filing Date:
20 July 2017 (20.07.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16180633.6 21 July 2016 (21.07.2016) EP

(71) Applicant: **NAGRAVISION S.A.** [CH/CH]; Route de Genève 22-24, 1033 Cheseaux-sur-Lausanne (CH).

(72) Inventor: **PELLETIER, Hervé**; chemin du Verger 2, 1053 Cugy (CH).

(74) Agent: **LEMAN CONSULTING S.A. 284**; Chemin de Précossy 31, 1260 Nyon (CH).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

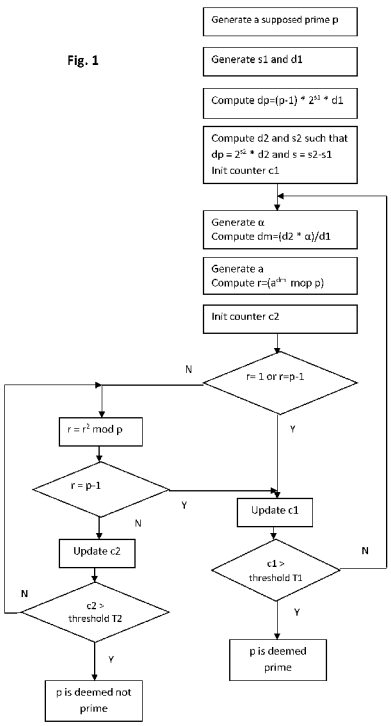
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

WO 2018/015464 A1

(54) Title: METHOD AND DEVICE TO VERIFY IF A NUMBER IS PRIME

Fig. 1



(57) Abstract: The present disclosure proposes a method to determine if a randomly generated number n is prime. This method comprises : a. generating randomly a value s1 and a value d1, b. calculating a value dp = (n-1) * 2^s1 * d1, c. calculating a value d2 and a value s2 such as dp = 2^{s2} * d2, and computing a value s = s2-s1, d. initializing a first counter c1, d. generating randomly a value α, e. calculating a value dm=d2*α/d1, f. generating randomly a value A, g. initializing a second counter c2, h. calculating a value r= A^{dm} mod n, i. verifying that the value r=1 or r=n-1, in the positive event : j. updating the first counter c1, k. if the first counter is below a first threshold T1, continuing with the step d.; I. otherwise, considering that the number n is prime; m. in the negative event : n. updating the value r = r^2 mod n; o. verifying that the value r=1, in the positive event, continuing with the step j; p. in the negative event : q. updating the second counter c2, r. if the second counter c2 is below a second threshold T2, continuing with the step n; s. otherwise, considering the number n is not prime.

METHOD AND DEVICE TO VERIFY IF A NUMBER IS PRIME

Introduction

Prime numbers play an important role in cryptography. Prime numbers, or simply primes, are integers greater than 1 that are not divisible by any integer other than 1 and themselves. Primes are important because the security of many encryption algorithms is based, on the fact that it is possible to multiply two large prime numbers very efficiently and get the result, while it is extremely costly in terms of computational needs to perform the reverse operation. Given a number known to be the product of two primes, recovering these two prime numbers out of their product is infeasible when the given number is large enough. This problem is called integers factorization and finding an algorithm which does it fast is one of the unsolved problems of computer science. A second example of a hard problem relying on primes and exploited by cryptographic algorithms is the computation of discrete logarithms modulo a prime number p. Given two numbers g and a, the operation consisting in raising to the power a the number g, where all the operations are performed modulo a prime p, i.e., computing A=g^a mod p can be performed very efficiently. The inverse operation, i.e., recovering the exponent a from A, an operation known as computing the discrete logarithm of A with respect to base g modulo p, is infeasible in practice provided the integer p is sufficiently large.

The Rivest-Shamir-Adleman (RSA) algorithm is one of the first practical public-key cryptosystems and is widely used for securing data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. For the RSA algorithm, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers.

A further example is the Diffie-Hellman protocol. It is one of the first public-key key exchange protocol and is widely used for securing data transmission. In such a protocol, the exchanged messages are public while a private part is kept secret. It is well-known that the security of the Diffie-Hellman protocol relies on the difficulty to compute discrete logarithms modulo a large prime p.

A further example requiring the use of a large prime number is the Digital Signature Algorithm (DSA).

The purpose of the present disclosure is the secure generation in presence of side-channel attacks of an integer and the verification that this integer is prime.

Brief description of the drawings

The present disclosure will be better understood with the attached figures, given as a not limiting example, in which:

- figure 1 illustrates the flowchart of the method of the present disclosure,

- figure 2 illustrates one example of a device executing the method.

Detailed description

Some important cryptographic algorithms, such as RSA or Diffie-Hellman, for instance, critically depend on hard problems involving large prime numbers. These large prime numbers are used in various applications, for example hashing, public-key cryptography, and search of prime factors in large numbers. In many examples, one or a plurality of primes form the secret part of a public-key cryptosystem. For instance, considering the RSA encryption algorithm, the "public key" consists of the product of two large primes and is used to encrypt a message, while the corresponding "secret key" consists of those two primes and they are necessary to decrypt the message. A main device can share the public key public, and everyone can use it to encrypt messages to the main device, but only the main device knows the prime factors and can decrypt the messages. Everyone else would have to factor the number, which takes too long to be practical, given the current state of the art of computational number theory. In the frame of the present disclosure, an example of a main device can be a service provider delivering audio/video services to one or a plurality of target devices. A target device will hold the public key of the service provider and will use this public key to encrypt the messages sent to the service provider so that only the service provider, with the corresponding secret key, can decrypt the message. In the same manner, the target device can comprises its own set of public and private keys, the public key being known by the service provider (for example by publishing the public key to a key repository and transmitting the public key to the service provider).

The standard way to generate big prime numbers is to take a preselected random number of the desired length, apply a Fermat test (best with the base 2 as it can be optimized for speed) and then to apply a certain number of Miller-Rabin tests (depending on the length and the allowed error rate, this number is typically comprised between 2 and 100) to get a number which is very probably a prime number.

The pre-selection is done either by test divisions by small prime numbers (up to few hundreds) or by sieving out primes up to 10,000 - 1,000,000 considering many prime candidates of the form b+2i (b big, i up to few thousands).

The Miller–Rabin primality test, or Rabin–Miller primality test, is an algorithm that determines whether a given number is prime, similar to the Fermat primality test and the Solovay–Strassen primality test.

Given an integer p, how can we check if p is prime? The most obvious way is to look for factors of p, but no efficient factoring algorithm is known.

A first assumption is the fact that p is odd, since deciding the primality of an even number is trivial.

By Fermat's Theorem, if p is prime, then for any a we have $a^{p-1}=1$ (mod n). This suggests the Fermat test for a prime: one generates a random base $a \in \{1,..., p-1\}$ and check whether $a^{p-1}=1$(mod n). If it is not the case, then p must be composite (not prime). However, one weakness of the Fermat test is that one may still get equality even when p is not prime for a certain class of numbers, known as the Carmichael numbers. If a is not coprime to p then the Fermat test fails, but then we can easily recover a factor of p by computing a greatest common devisor for gcd(a, p).

The Miller- Rabin test improves the Fermat test in that one can prove that it works on all numbers. It relies on the fact that there are no non-trivial square roots of 1 modulo a prime p, i.e, square roots that are different from 1 and p-1. One test is to first check $a^{p-1}=1$, then check that $a^{(p-1)/2}=\pm 1$, because $a^{(p-1)/2}$ is a square root of 1.

The Miller-Rabin test works by picking a random integer a between 1 and p-1, then check that the above sequence has the correct form. If the sequence does not begin with 1, or the first member of the sequence that is not 1 is also not −1, then p is not prime.

It turns out that for any composite number n to test, the worst-case probability that n passes the Miller-Rabin test is at most 1/4. On average, it is even significantly less. Thus, by iterating the procedure many times, the probability that a composite number n passes several tests decreases exponentially fast. If n fails the Miller-Rabin test with a sequence starting with 1, then we have a nontrivial square root of 1 modulo n. Again, under that scenario, it means that one can factor the number n.

In practice, a single iteration of the Miller-Rabin test is implemented as follows:

1. Given n, find s so that $n-1=2^s d$ for some odd integer d

2. Choose a random base $a \in \{1,..., n-1\}$

3. If $a^d=1$ mod n or $a^d=n-1$ mod n then n passes (and exit).

4. For i=0,...,s−1, check if $a^{(2\wedge i\ d)} = 1$ then the number is not prime and if not check whether $a^{(2\wedge i\ d)} = n-1$. If so, n is maybe prime (and exit).

5. Otherwise n is not prime.

In practice, this process is repeated several times with freshly generated random bases until the error probability is as low as desired.

Since the apparition of side-channel attacks in 1999, many attacks have been published against the implementation of cryptographic algorithms. A side-channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis). For example, timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system. Some side-channel attacks require technical knowledge of the internal operation of the system on which the cryptography is implemented, although others such as differential power analysis are effective as black-box attacks. Today, embedded devices implement more and more cryptographic features. One of these features is the capability to generate prime numbers for RSA or DSA algorithms. Until now, such complex operation was not really attacked since such operation is supposed to be performed in a secure environment, like a secure building, for example. But with the development of mobility and the Internet of Things (IoT), such assumption has unfortunately become void. Moreover, the development of new stochastic side-channel attacks allows to recovery a secret with only little information.

The one or more client devices facing the generation of prime numbers include, but are not limited to, set-top boxes, mobile phones, personal digital/data assistants ("PDAs"), computers, two-way pagers, and other types of client devices (e.g., Internet appliances, cable modems, etc.). The one or more client devices may include "thin-client" devices.

Thin-client devices typically include devices with small physical size, light weights, limited device resources, and corresponding small display screen sizes, if they have any screens at all. These thin-client devices have a number of constraints including limited processing power, limited memory, limited battery life, a limited user interface if any, etc.

A thin-client typically performs limited application processing, or none at all. Thin-clients typically rely on a "fat" server to perform application processing such as encryption. However, the present disclosure is not limited to "thin-client" devices. The one or more client devices may also include "fat-client" devices, such as desktop computers (not illustrated), laptop computers or other computing devices and computing devices capable of performing all or most of the application processing, and may have as many device resources as the server device.

To check if a number n is prime, one of the most widely used probabilistic test is the Miller-Rabin test as explained above. This test requires to decompose $n-1=2^s * d$ where d is an odd integer. Then, based on different random bases a, the value $a^d \bmod n$ is computed and tested. In this context, d is a very sensitive value and since the operations $a^d \bmod n$ is repeated for different values of a, this operation could leak information by side-channel. Indeed, the modular exponentiation $a^d \bmod n$ is very sensitive to side-channel attacks. As it is known, modular arithmetic is deeply dependent of the operands value. Indeed modular arithmetic operations are complex and imbricated. For example, even a simple timing difference can reveal the secret value of the operands (value a in the case of modular exponentiation $a^d \bmod n$).

Moreover, the classical known countermeasures are not easy to implement. The primality test operation takes a long time and so hardware countermeasures are too penalizing to be applied. The other algebraic masking methods cannot be used since n is maybe not prime.

In the present disclosure, it is proposed a way to solve this problem, by masking the sensitive value d during all the computation. The main idea is to take another random distribution for the primality test.

1. To begin with, two random integer values s1 and d1 are generated, preferably using a cryptographically secure pseudo-random generator.

2. A value dp is computed such as $dp=(p-1)*2^{s1}*d1$.

3. The values s2 and d2 are determined such that $dp=2^{s2}*d2$. To obtain s2 and d2, the value dp is divided by 2 a number of time corresponding to s2 until it is no longer possible to divide by 2 (odd value). This odd value is then d2. Since d1 and s1 are known, one can calculate a value s = s2-s1.

4. A first loop counter c1 is initialized. A counter, during the loop function, can be incremented or decremented. The initialization comprises setting the counter to a threshold value T1 and decrementing the counter until it is equal to zero. Alternatively, the initialization comprises setting the counter to zero and incrementing the counter until the threshold value T1 is reached.

5. A value α is randomly generated, preferably using a cryptographically secure random number generator.

6. A value dm is computed as $dm=d2*\alpha/d1$. dm is the masked value of d and it corresponds to $\alpha*d$ value.

7. An integer value A is randomly generated, preferably using a cryptographically secure pseudo-random number generator.

8. A second loop counter c2 is initialized. Concerning the way the counter is initialized, the same remarks concerning the counter c1 can be applied to c2.

9. A value r is computed such as $r = A^{dm} \bmod n$

10. This value r is compared to 1 or n-1. In case that the comparison is positive :

10.1 updating the first loop counter c1 and

10.2 if the loop counter c1 has not reached the first threshold T1, then restarting with the step 5, otherwise, using the number n as prime number.

11. and if the comparison of r with 1 or n-1 is negative (i.e. $r \neq 1$ and $r \neq (p-1)$) then

12. updating the value r such as $r = r^2 \bmod n$

13. the updated value r is compared to 1, and in case that the comparison is positive, continuing with the step 10.1., otherwise :

14. updating the second loop counter c2, if the second loop counter c2 has not reached the second threshold T2 continuing with the step 12 and if the second loop counter c2 has reached the second threshold T2, discarding the number n as not being a prime number.

Once is has been established that the number n is prime, the number n is used to further cryptographic operations requiring a prime number.

With the above described method the sensitive element d is masked completely during all the operations needed for the Miller-Rabin test. By this way a side-channel attack against the element d is not possible, since the value d is never present in memory in clear and its value is statistically uncorrelated with the numbers processed by the protected Miller-Rabin test.

According to one embodiment, the second counter c2 is defined such that c2 = s2-s1.

According to one particular embodiment, the step of generating a random value α, further comprises a verification that the generated value α is within the range of d2/4 and d2/2, in the contrary, generating randomly a new value α.

In the frame of the present disclosure, the random numbers are determined by cryptographically secure pseudo-random number generator. A cryptographically secure pseudo-random number

generator can satisfy the next-bit test, which establishes that given the first n bits of a random sequence, there is no polynomial-time algorithm able to predict the (n+1)th bit with probability of success greater than 50%. A cryptographically secure pseudo-random number generator should also resist to the "state compromise extensions". Even if part of all of the state being compromised, it does not allow for reconstruction of the prior stream of random numbers. Several certifications exist such as NIST SP800-22 Compliance, METAS Certification, CTL Certification, BSI AIS 31-compliance Certificate.

In order to hide the counters and avoid exposing the loops controlled by the counter, according to one embodiment, the counters are replaced by a LFSR. The characteristic of a Linear-feedback shift register LFSR is to produce an unpredictable output. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current or previous state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits that appears random and has a very long cycle. A LFSR can be shifted forward or backward. A random value is selected and used as the seed to load the initial state of the LFSR. For a given threshold T1, the LFSR is then shifted backward a number of time equal to the threshold T1. The LFSR is then ready to play the role of a counter. The update of the counter is replaced by a shift forward of the LFSR. The output of the LFSR is then compared to the seed in order to determine if the number of maximal loops is reached. This way each time that a LFSR-counter is used, the output of the LFSR will be different, thus giving no indication to an outside observer about the number of loops executed. In the frame of the present disclosure, the first counter c1 or the second counter c2, or both can be replaced by LFSR-counters.

Figure 2 illustrates a block diagram of one implementation of a computing device 200 within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative implementations, the machine may be connected (e.g., networked) to other machines in a Local Area Network (LAN), an intranet, an extranet, or the Internet. The machine may operate in the capacity of a server or a client machine in a client-server network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet computer, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term “machine” shall also be taken to include any collection of machines (e.g., computers) that

individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

The example computing device 200 includes a processing device 202, a main memory 204 (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM) or Rambus DRAM (RDRAM), etc.), a static memory 206 (e.g., flash memory, static random access memory (SRAM), etc.), and a secondary memory (e.g., a data storage device 216), which communicate with each other via a bus 208.

Processing device 202 represents one or more general-purpose processors such as a microprocessor, central processing unit, or the like. More particularly, the processing device 202 may be a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, processor implementing other instruction sets, or processors implementing a combination of instruction sets. Processing device 202 may also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. Processing device 202 is configured to execute the processing logic (instructions 224) for performing the operations and steps discussed herein. Frr the purpose of the verification that a number is prime, the processing device 202 may have access to a cryptographic module 223 executing the method steps of the claim 1. The input of this module is the number n and the output of this module is a result 0 or 1, 0 if the number is considered as non prime and 1 is the number is considered as prime.

The computing device 200 may further include a network interface device 220. The computing device 200 also may include a graphic display unit 210 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)), an alphanumeric input device 212 (e.g., a keyboard), a cursor control device 214 (e.g., a mouse), and an audio generation device 218 (e.g., a speaker).

The data storage device 216 may include a machine-readable storage medium (or more specifically a non-transitory computer-readable storage medium) 222 on which is stored one or more sets of instructions 224 embodying any one or more of the methodologies or functions described herein. The instructions 224 may also reside, completely or at least partially, within the main memory 204 and/or within the processing device 202 during execution thereof by the computer system 200, the main memory 204 and the processing device 202 also constituting computer-readable storage media.

The computer-readable storage medium 222 may also be used to store a service to determine the primality of a number (as described with reference to Figure 1), and/or a software library containing methods that call a service to determine the primality of a number. The computing device 200 further comprises a random generator module 226 in charge of producing cryptographically secure random number. While the computer-readable storage medium 222 is shown in an example implementation to be a single medium, the term "computer-readable storage medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "computer-readable storage medium" shall also be taken to include any medium other than a carrier wave that is capable of storing or encoding a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies described herein. The term "computer-readable storage medium" shall accordingly be taken to include, but not be limited to, solid-state memories, and optical and magnetic media.

In an implementation, the modules, components and other features described herein, designed to execute the method illustrated by the figure 1, can be implemented as discrete hardware components or integrated in the functionality of hardware components such as ASICS, FPGAs, DSPs or similar devices as part of an individualization server. In addition, the modules can be implemented as firmware or functional circuitry within hardware devices. Further, the modules can be implemented in any combination of hardware devices and software components, or only in software.

In an implementation a computer-implemented cryptographic method that includes performing, by a processing device of a first computer, at least part of a verification operation in which a number n is verified as a prime number. The processing device receives the number n from the first computer. The processing device generates a first random integer value s1 and a first random value d1. The processing device calculates a value dp = (n-1) * 2^s1 * d1, decomposes dp to extract a second value d2 and a second integer s2 so that dp = 2^{s2} * d2, and computes a sensitive value s = s2-s1.

The processing device initializes a first counter c1, generates a third random value α and calculates a value dm=d2*α/d1. The processing device generates a fourth random value A, initializes a second counter c2, and calculates a comparison value r= A^{dm} mod n to verify if that the comparison value satisfies r=1 or r=n-1.

In response to determining that the comparison value satisfies r=1 or r=n-1, the processing device updates the first counter c1 and determines whether the first counter is below a first thr eshold T1.

In an implementation, the first threshold T1 is a preconfigured number. In response to determining the first counter is below a first threshold T1, the processing device repeats to generating new value for the third random value α. In response to determining the first counter is not below a first threshold T1, the processing device considers that the number n is prime. For example, the processing device can record that the value is prime and transmit the prime number n to a cryptographic operation carried by the processing device.

In response to determining that the comparison value satisfies that the value $r \neq 1$ and $r \neq n-1$, the processing device updates the value $r = r^2 \bmod n$. The processing device verifies if the value r=1, updates the first counter c1 and determines whether the first counter is below a first threshold T1. In an implementation, the first threshold T1 is a preconfigured number. In response to determining the first counter is below a first threshold T1, the processing device repeats to generating a new value for the third random value α. In response to determining the first counter is below a first threshold T1, the processing device considers that the number n is prime. In an implementation, the prime number n is used to further cryptographic operations requiring a prime number.

If after updating value $r = r^2 \bmod n$, the processing device determines that value $r \neq 1$and updates the second counter c2.

In response to determining that the second counter c2 is below a second threshold T2, the processing device returns to updating the value $r = r^2 \bmod n$.

In response to determining that the second counter c2 is not below a second threshold T2, the processing, considers the number n is not prime.

Some portions of the detailed description have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless

specifically stated otherwise, as apparent from the following specification, it is appreciated that utilizing terms such as "generating", "comparing", "calculating", "verifying", "updating" or the like, refer to the actions and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

Implementations of the present disclosure also relate to an apparatus for performing the operations herein. This apparatus may be specially constructed for the discussed purposes, or it may comprise a general purpose computer system selectively programmed by a computer program stored in the computer system. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic disk storage media, optical storage media, flash memory devices, other type of machine-accessible storage media, or any type of media suitable for storing electronic instructions, each coupled to a computer system bus.

It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other implementations will be apparent to those of skill in the art upon reading and understanding the above description. Although the present disclosure has been described with reference to specific example implementations, it will be recognized that the disclosure is not limited to the implementations described, but can be practiced with modification and alteration within the spirit and scope of the appended claims. Accordingly, the specification and drawings are to be regarded in an illustrative sense rather than a restrictive sense. The scope of the disclosure should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

Claims

1. Method, executed by one or a plurality of processors, to verify and use the primality of a number n, said method comprising :

a. generating randomly a value s1 and a value d1,

b. calculating a value dp = (n-1) * 2^s1 * d1,

c. calculating a value d2 and a value s2 such as dp = 2^{s2} * d2, and computing a value s = s2-s1,

d. initializing a first counter c1,

e. generating randomly a value α,

f. calculating a value dm, wherein dm=d2*α/d1,

g. generating randomly a value A,

h. initializing a second counter c2,

i. calculating a value r= A^{dm} mod n,

j. in response to a verification that the value r=1 or r=n-1, then:

k. updating the first counter c1,

l. if the first counter c1 is below a first threshold T1, continuing with the step e,

m. otherwise, using the number n as prime number in a cryptographic operation.

2. The method of claim 1, wherein:

n. in response to a verification that the value r≠1 and r≠n-1, then:

o. updating the value r = r^2 mod n,

p. in response to a verification that the value r=1, then continuing with the step k.

q. in response to a verification that the value r≠1, then:

r. updating the second counter c2,

s. if the second counter c2 is below a second threshold T2, continuing with the step o.

t. otherwise, discarding the number n as prime number.

3. The method of claim 1 or 2, wherein the second counter c2 is defined such that c2 = s2-s1.

4. The method of any of the claims 1 to 3, further comprising:

verifying that the value α is within the range of d2/4 and d2/2, in the contrary, generating randomly a new value α.

5. The method of any of the claims 1 to 4, wherein the random numbers are generated using a cryptographically secure pseudo-random generator.

6. A non-transitory machine-readable storage medium comprising instructions that, when executed by one or more processors of a computing device, cause the computing device to:

a. generate randomly a value s1 and a value d1,

b. calculate a value dp = (n-1) * 2^s1 * d1,

c. calculate a value d2 and a value s2 such as dp = 2^{s2} * d2, and computing a value s = s2-s1,

d. initialize a first counter c1,

e. generate randomly a value α,

f. calculate a value dm, wherein dm=d2*α/d1,

g. generate randomly a value A,

h. initialize a second counter c2,

i. calculate a value r= A^{dm} mod n,

j. in response to a verification that the value r=1 or r=n-1, then:

k. update the first counter c1,

l. if the first counter is below a first threshold T1, continue with the step e.

m. otherwise, use the number n as a prime number in a cryptographic operation.

7. The non-transitory machine-readable storage medium of claim 6, where the one or more processors are further to:

n. in response to a verification that the value r≠1 and r≠n-1, then:

o. update the value r = r^2 mod n,

p. In response to a verification that the value r=1, then continue with the step k.

q. in response to a verification that the value r≠1, then:

r. update the second counter c2,

s. if the second counter c2 is below a second threshold T2, continue with the step o.

t. otherwise, discard the number n is not prime.

8. The non-transitory machine-readable storage medium of any of the claims 6 to 7, comprising instructions that, when executed by one or more processors of a computing device, cause the processor to:

define the second counter c2 such that c2 = s2-s1.

9. The non-transitory machine-readable storage medium of any of the claims 6, 7, or 8, comprising instructions that, when executed by one or more processors of a computing device, cause the computing device to:

verify that the value α is within the range of d2/4 and d2/2, in the contrary, generating randomly a new value α.

10. Computing device, comprising one or a plurality of processors, further comprising a random number generator (226) and a cryptographic module (223), said cryptographic module being configured to, while receiving a number n:

a. generate randomly a value s1 and a value d1,

b. calculate a value dp = (n-1) * 2^s1 * d1,

c. calculate a value d2 and a value s2 such as dp = 2^{s2} * d2, and computing a value s = s2-s1,

d. initialize a first counter c1,

e. generate randomly a value α,

f. calculate a value dm,

g. generate randomly a value A,

h. initialize a second counter c2,

i. calculate a value r= A^{dm} mod n,

j. verify that the value r=1 or r=n-1, in the positive event:

k. update the first counter c1,

l. if the first counter c1 is below a first threshold T1, continue with the step e.

m. otherwise, use the number n as a prime number in a cryptographic operation.

11. The computing device of claim 10, wherein the cryptographic module is further configured to:

n. in the negative event:

o. update the value $r = r^2 \bmod n$

p. verify that the value r=1, in the positive event, continue with the step k .

q. in the negative event:

r. update the second counter c2,

s. if the second counter c2 is below a second threshold T2, continue with the step o.

t. otherwise, outputting a result representing that the number n is not prime.

Fig. 1

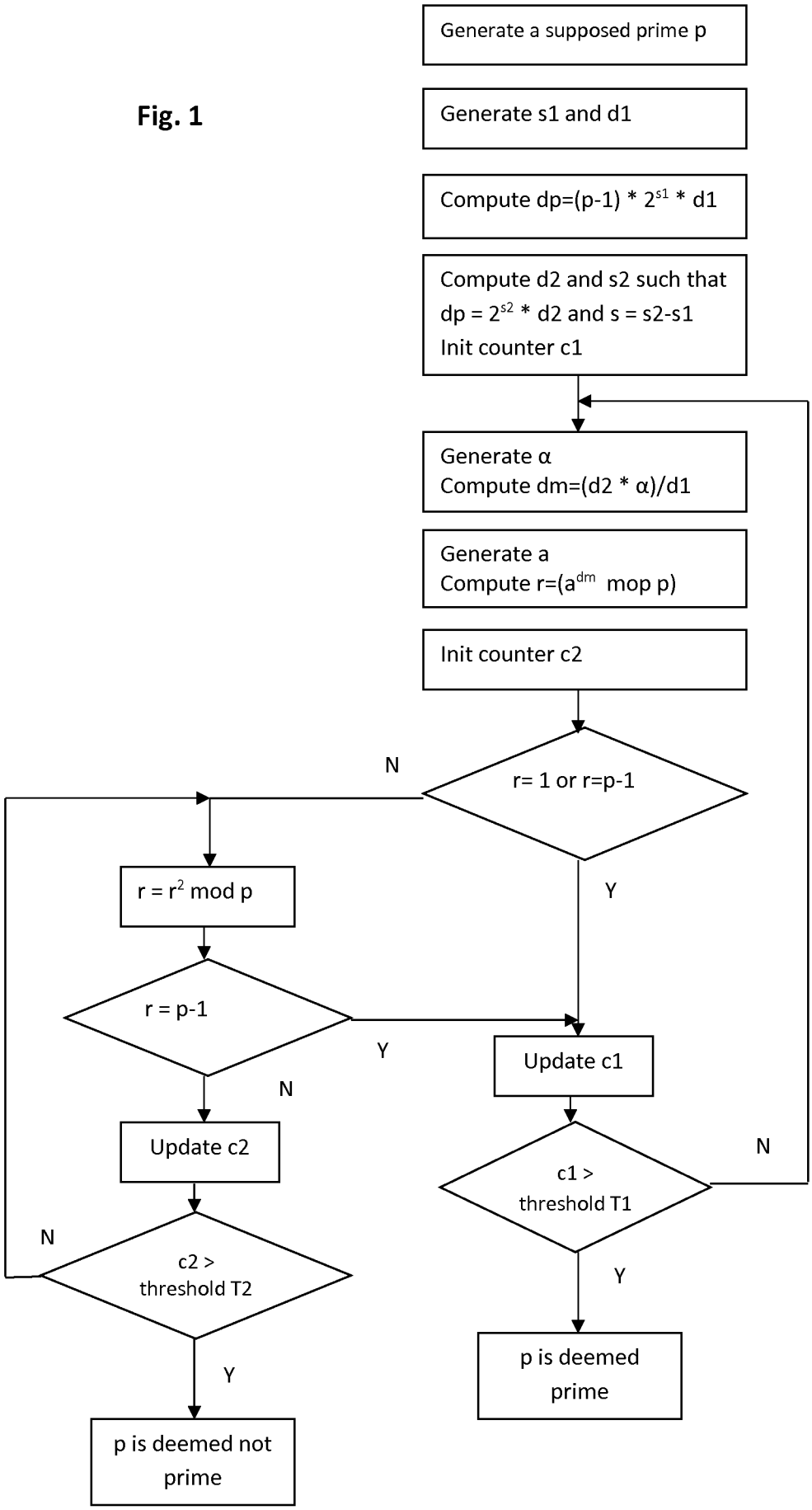
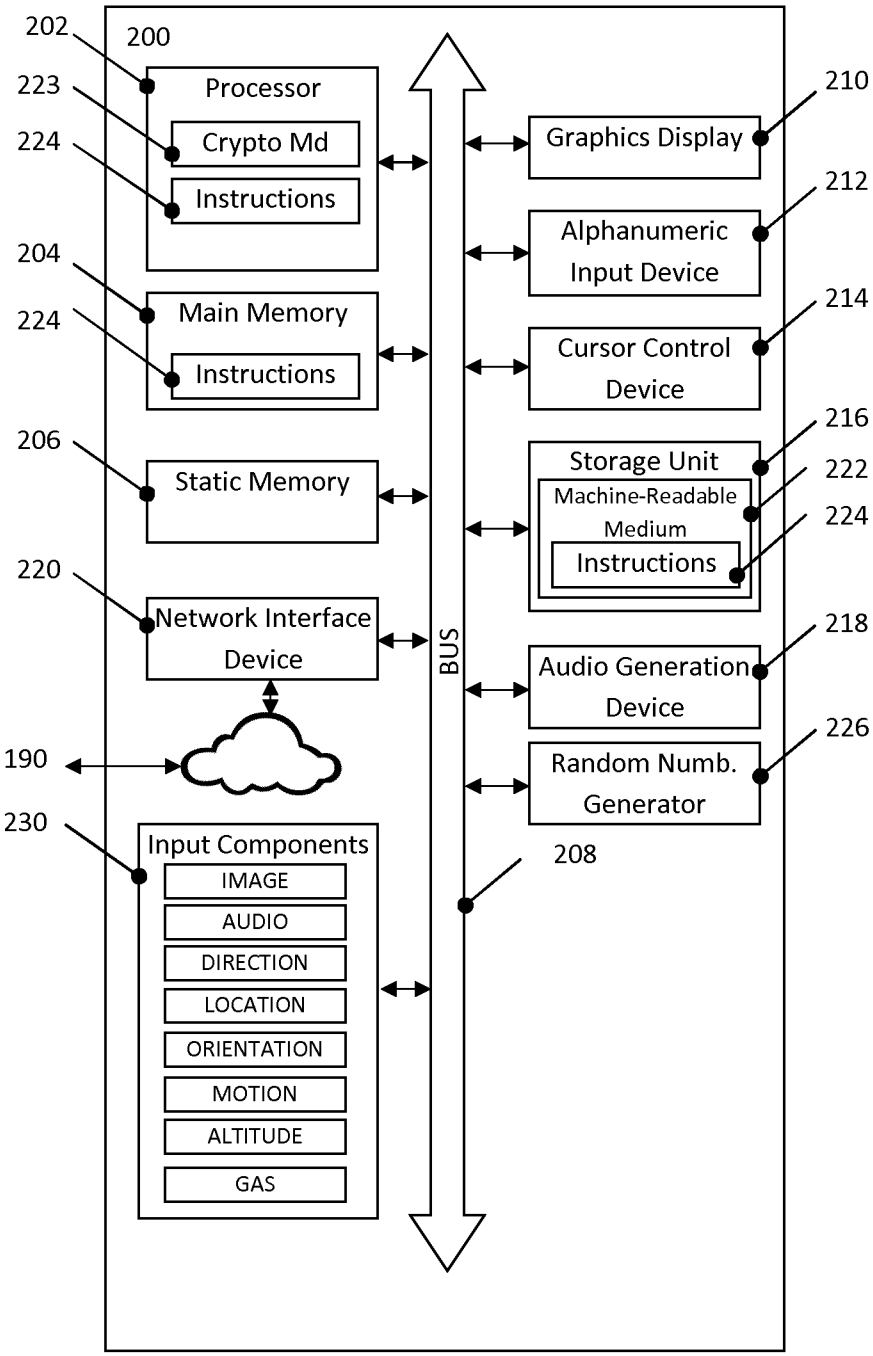

Generate a supposed prime p
Generate s1 and d1
Compute dp=(p-1) * 2^s1 * d1
Compute d2 and s2 such that dp = 2^s2 * d2 and s = s2-s1
Init counter c1
Generate α
Compute dm=(d2 * α)/d1
Generate a
Compute r=(a^dm mop p)
Init counter c2
r= 1 or r=p-1
N
Y
r = r^2 mod p
r = p-1
Y
N
Update c1
Update c2
c1 > threshold T1
N
Y
c2 > threshold T2
N
Y
p is deemed prime
p is deemed not prime

Fig. 2


202
200
Processor
223
Crypto Md
224
Instructions
204
Main Memory
224
Instructions
206
Static Memory
220
Network Interface Device
190
230
Input Components
IMAGE
AUDIO
DIRECTION
LOCATION
ORIENTATION
MOTION
ALTITUDE
GAS
BUS
210
Graphics Display
212
Alphanumeric Input Device
214
Cursor Control Device
216
Storage Unit
222
Machine-Readable Medium
224
Instructions
218
Audio Generation Device
226
Random Numb. Generator
208

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2017/068287

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/08 H04L9/30 H04L9/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DE 103 26 057 A1 (CV CRYPTOVISION GMBH [DE]) 13 January 2005 (2005-01-13) the whole document	1-11
A	US 2013/182839 A1 (VUILLAUME CAMILLE [JP]) 18 July 2013 (2013-07-18) paragraphs [0218] - [0240]	1-11
A	GEBHARD BÖCKLE: "The Miller-Rabin test with randomized exponents", JOURNAL OF MATHEMATICAL CRYPTOLOGY, vol. 3, no. 4, 1 January 2009 (2009-01-01) , pages 1-13, XP055335996, ISSN: 1862-2976, DOI: 10.1515/JMC.2009.019 pages 1-2	1-11

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 October 2017	27/10/2017

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Horbach, Christian

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	"Chapter 4: Public-Key Parameters ED - Menezes A J; Van Oorschot P C; Vanstone S A", HANDBOOK OF APPLIED CRYPTOGRAPHY; [CRC PRESS SERIES ON DISCRETE MATHEMATICES AND ITS APPLICATIONS], CRC PRESS, BOCA RATON, FL, US, PAGE(S) 133 - 168 , 1 October 1996 (1996-10-01), XP001525004, ISBN: 978-0-8493-8523-0 Retrieved from the Internet: URL:http://www.cacr.math.uwaterloo.ca/hac/ pages 135-141 -----	1-11
A	"Chapter 2: Mathematical Background ED - Menezes A J; Van Oorschot P C; Vanstone S A", HANDBOOK OF APPLIED CRYPTOGRAPHY; [CRC PRESS SERIES ON DISCRETE MATHEMATICES AND ITS APPLICATIONS], CRC PRESS, BOCA RATON, FL, US, PAGE(S) 49 - 86 , 1 October 1996 (1996-10-01), XP001525002, ISBN: 978-0-8493-8523-0 Retrieved from the Internet: URL:http://www.cacr.math.uwaterloo.ca/hac/ pages 63-75 -----	1-11

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2017/068287

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
DE 10326057 A1	13-01-2005	NONE	
US 2013182839 A1	18-07-2013	JP 5848106 B2	27-01-2016
		JP 2013113978 A	10-06-2013
		US 2013182839 A1	18-07-2013