



- (51) **International Patent Classification:**
H04L 9/06 (2006.01)
- (21) **International Application Number:**
PCT/EP2012/056685
- (22) **International Filing Date:**
12 April 2012 (12.04.2012)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
P 201130574 12 April 2011 (12.04.2011) ES
- (71) **Applicant (for all designated States except US):** **TELEFONICA, S.A.** [ES/ES]; Gran Vía, 28, E-28013 Madrid (ES).
- (72) **Inventors; and**
- (75) **Inventors/Applicants (for US only):** **MONTOJA VITINI, Fausto** [ES/ES]; Consejo Superior de Investigaciones Científicas (CSIC), C/ Serrano, 117, E-28010 Madrid (ES). **ORÚE LOPEZ, Amalia, Beatriz** [ES/ES]; Consejo Su-

perior de Investigaciones Científicas (CSIC), C/ Serrano, 117, E-28010 Madrid (ES). **GUERRA ESTEVEZ, Alberto** [ES/ES]; Consejo Superior de Investigaciones Científicas (CSIC), C/ Serrano, 117, E-28010 Madrid (ES). **HERNANDEZ ENCINAS, Luis** [ES/ES]; Consejo Superior de Investigaciones Científicas (CSIC), C/ Serrano, 117, E-28010 Madrid (ES). **MARTIN MUÑOZ, Agustín** [ES/ES]; Consejo Superior de Investigaciones Científicas (CSIC), C/ Serrano, 117, E-28010 Madrid (ES). **SOTO RODRIGUEZ, Mercedes** [ES/ES]; Consejo Superior de Investigaciones Científicas (CSIC), C/ Serrano, 117, E-28010 Madrid (ES).

(74) **Agent:** **CARPINTERO LOPEZ, Francisco**; Herrero & Asociados, S.L., C/ Alcalá, 35, E-28014 Madrid (ES).

(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, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME,

[Continued on next page]

(54) **Title:** METHOD AND SYSTEM FOR IMPROVING THE SYNCHRONIZATION OF STREAM CIPHERS

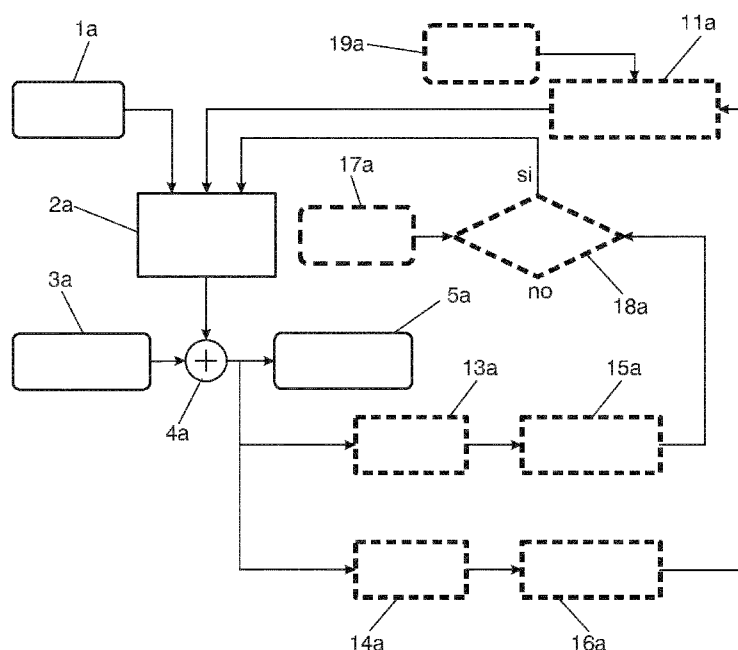


FIG. 3

(57) **Abstract:** Method and system for improving stream cipher encryption of data, where the data to be encrypted, called plaintext, is combined with a random sequence of digits, called key stream, to obtain the encrypted plain text called cipher text. The invention proposed a new type of synchronism of stream ciphers, where the synchronization action is produced at random intervals in an unpredictable time schedule.



MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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.

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, ML, MR, NE, SN, TD, TG).

Published:

(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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT,

- with international search report (Art. 21(3))
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))

METHOD AND SYSTEM FOR IMPROVING THE SYNCHRONIZATION OF
STREAM CIPHERS

DESCRIPTION

5

TECHNICAL FIELD

The present invention relates generally to the encryption of data and more particularly to a method and system for improving the synchronization and, therefore, the behaviour of stream cipher encryption procedures.

10

DESCRIPTION OF THE PRIOR ART

Encryption is the process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information (in cryptography, referred to as cipher text). In many contexts, the word encryption also implicitly refers to the reverse process, decryption, to make the encrypted information readable again (i.e. to make it unencrypted).

15

20

Encryption is now commonly used in protecting information within many kinds of systems. Encryption can be used to protect data "at rest", such as files on computers and storage devices. In recent years there have been numerous reports of confidential data such as customers' personal records being exposed through loss or theft of laptops or backup drives. Encrypting such files at rest helps protect them should physical security measures fail. Digital rights management systems which prevent unauthorized use or reproduction of copyrighted material and protect software against reverse engineering are another somewhat different example of using encryption on data at rest. Encryption is also used to protect data in transit, for example data being transferred via networks (e.g. the Internet, e-commerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in

25

30

transit being intercepted in recent years. Encrypting data in transit also helps to secure it as it is often difficult to physically secure all access to networks.

5 A stream cipher is a type of symmetric encryption algorithm. The encryption is accomplished by combining the plaintext (the message to be encrypted) with a key stream, usually with the exclusive or (XOR) operation but other operations are also possible. The combination is done digit by digit; usually the digits consist of one or several bits. The transformation of successive plaintext bits varies during the encryption, equal plaintext bits are encrypted into different cipher text bits depending
10 of its position on the message.

The key stream consists of a random sequence of bits, that can be generated in advance or in the moment of use. This sequence of bits can represent any data, whose confidentiality should be protected, both for its storing and transmission: text,
15 images, programs, audio and video. To guarantee the unbreakability of the cipher, the length of the key stream should be equal to the length of the plaintext, and each key stream should not be reused for more than one operation of encryption. Since the entire key stream is random, even an opponent with infinite computational resources can only guess the plaintext if he sees the cipher text. Such a cipher is
20 said to offer perfect secrecy.

The historic precedents of stream cipher are the Caesar Cipher, whose key stream had only one digit consisting of an alphabet character, and the Blaise de Vigenere Cipher, whose key had a limited number of digits and was replayed as many times
25 as necessary to reach the length of the plaintext. The first modern stream cipher was the Vernam Cipher—also known as the one-time pad— which is the only mathematically demonstrable unbreakable cipher. The one-time pad was used during wartime over diplomatic channels requiring exceptionally high security. The fact that the secret key (which can be used only once) is as long as the message
30 introduces severe key management problems. While perfectly secure, the one-time pad is in general impractical. Today's stream ciphers were developed as an approximation to the action of the one-time pad. While contemporary stream ciphers are unable to provide the satisfying theoretical security of the one-time pad, they are at least practical.

The generation of the key stream is accomplished by a Pseudo Random Number Generator (PRNG) or by certain modes of operation of block ciphers, which effectively transform them into a key stream generator; in this way, any block cipher can be used as a stream cipher; as in TDEA (Triple Data Encryption Algorithm) or AES (Advanced Encryption System) in OFB (open feedback) or CRT (counter) modes of operation. However, stream ciphers with a dedicated design are typically much faster. PRNGs or block ciphers must have a key K that may determine the next state function and the output function as well as the initial state. The basic problem of key stream generator design in the context of finite state machines is to find next state functions and output functions which are guaranteed to produce a key stream that satisfies the basic requirements of large period, large linear complexity and uniform distribution properties. The determination of these basic performance criteria, of course, implies the analyzability of the key stream generator. The construction of secure running key generators necessarily implies the introduction of nonlinear transformations, which greatly complicates the indispensable analysis. There exist implicit and explicit methods for introducing nonlinear effects.

The generation of the key stream can be independent of the plaintext and cipher text, yielding what is termed a synchronous stream cipher, or it can depend on the data and its encryption, in which case the stream cipher is said to be self-synchronizing. The Vernam Cipher was of the first type.

A key stream generator can be described by a 6-tuple $(S, R, f, g, K1, i)$, more precisely:

S , set of internal states;

R , output alphabet;

$f: f(S, K1) \rightarrow S$, the state transition function;

$g: g(S, K1) \rightarrow R$, the output function;

$K1$, the key of the functions f and g ;

i , the seed that determines the initial state.

The number of internal states S is much bigger than the output alphabet R ; each state is a function f of the key $K1$ and the previous state (the key $K1$ use to be a

sequence of digits); the output in each moment is a function g of the key $K1$ and the corresponding state; the initial state is determined by the constant i . Each time that the internal state changes a new output digit is generated, which is a function of the new state and the key $K1$; the moment of the internal state changing is usually controlled by an internal clock or a synchronism mechanism between a transmitter and a receiver.

There are mainly two types of stream cipher, the synchronous stream ciphers and the self-synchronizing stream ciphers.

In a **synchronous stream cipher** a stream of pseudo-random digits is generated independently of the plaintext and cipher text messages, and then combined with the plaintext (to encrypt) or with the cipher text (to decrypt). In the most common form, binary digits (bits) are used, and the key stream is combined with the plaintext using the exclusive or (XOR) (other logic functions can be used as well). This is termed a binary additive stream cipher.

In a synchronous stream cipher, the sender and receiver must be exactly in step for decryption to be successful. If digits are added or removed from the message during transmission, synchronization is lost. To restore synchronization, various offsets can be tried systematically to obtain the correct decryption. Another approach is to tag the cipher text with markers at regular points in the output. If, however, a bit is corrupted in transmission, rather than added or lost, only a single bit in the plaintext is affected and the error does not propagate to other parts of the message. This property is useful when the transmission error rate is high; however, it makes less likely the error would be detected without further mechanisms.

In synchronous stream ciphers, the next state depends only on the previous state and not on the input so that the succession of states is independent of the sequence of characters received. Consequently, the enciphering transformation is memoryless, though time-varying. But the device itself is not memoryless, it needs the internal memory to generate the necessary state sequence. It is natural therefore, in a synchronous stream cipher, to separate the enciphering

transformation from the generation process of the time-varying parameter that controls the enciphering transformation.

Figure 1 illustrates the complete block diagram of a synchronous stream cipher and decipher system. The plaintext (3) is encrypted into the cipher text (5), by the modulo-2 addition (XOR) (4) of the plaintext bits with the key stream bits generated by the PRNG (2); this PRNG generates a key stream which is a complicated function of the Key, which is stored in the register (1). If both the plaintext and the key stream are composed of independent bits, the XOR operation is accomplished bit by bit; but if both the plaintext and the key stream are composed of digits formed by blocks of several bits, the XOR operation is made bitwise, that is bit a bit. For example, if the plaintext digit pd is formed by the 4 bits $pd = (pb1, pb2, pb3, pb4)$ and the key digit kd is formed by the 4 bits $kd = (kb1, kb2, kb3, kb4)$ the XOR operation will be: $pd \text{ XOR } kd = (pb1 \text{ XOR } kb1, pb2 \text{ XOR } kb2, pb3 \text{ XOR } kb3, pb4 \text{ XOR } kb4)$.

The cipher text is transmitted to the receiving end, where it is decrypted into the retrieved plaintext (10), by the modulo-2 subtraction (9) of the key stream bits, generated by the PRNG (7), from cipher text bits (8) —the modulo-2 addition and subtraction operations are identical, hence, both are implemented by a unique mechanism: the XOR combination of bits—; the PRNG (7) generates a key stream which is identical to the generated by the PRNG (2), as a function of the Key (6), whose exact replica is stored in the register (1).

In the **self-synchronizing stream ciphers** (also known as asynchronous stream ciphers and also as ciphertext autokey), the key stream is not independent of the plaintext and cipher text messages because the previous N cipher text digits to compute the key stream. The idea of self-synchronization is known since in 1946, and has the advantage that the receiver will automatically synchronize with the key stream generator after receiving N cipher text digits, making it easier to recover the plaintext, if digits are dropped or added to the message stream. Single-digit errors are limited in their effect, affecting only up to N plaintext digits.

Self-synchronizing stream ciphers have the facility to resume correct decryption if the key stream generated by the decrypting unit falls out of synchronization with the encrypting key stream. Suppose the encryption of a bit depends on k previous cipher text bits. The system demonstrates limited error propagation; if one bit is received incorrectly then decryption of the following k bits may be incorrect. Additionally however, the system is able to resynchronize itself and produce a correct decryption after k bits have been received correctly. This makes such ciphers suitable for applications where synchronization is difficult to maintain. In contrast with synchronous stream ciphers, in self-synchronizing stream ciphers, the deciphering transformation has finite memory with respect to the influence of past characters so that an erroneous or lost cipher text character causes only a fixed number of errors in the deciphered plaintext, after which again correct plaintext is produced. For these stream ciphers the function that defines the next state of the cryptosystem takes as input some of the previously digits of the generated cipher text. The most common example of this is provided by some block cipher in what is termed cipher-feedback (CFB) mode.

Figure 2 illustrates the complete block diagram of a self-synchronizing stream cipher and decipher system. The plaintext (3) is encrypted into the cipher text (5), by the XOR (4) of the plaintext bits with the key stream bits generated by the PRNG (2); this PRNG generates a key stream which is a complicated function of the Key (1) and of the last N digits of the cipher text, stored in the register (11). If both the plaintext and the key stream are composed of independent bits, the XOR operation is accomplished bit by bit; but if both the plaintext and the key stream are composed of digits formed by blocks of several bits, the XOR operation is made bitwise. The cipher text is transmitted to the receiving end, where it is decrypted into the retrieved plaintext (10), by the modulo-2 subtraction (9) of the key stream bits, generated by the PRNG (7), from cipher text bits (8); the PRNG (7) generates a key stream which is identical to the generated by the PRNG (2), as a function of the Key (6) and of the last N digits of the received cipher text, stored in the register (12).

The main problems of existing stream ciphers are the following:

Synchronous stream ciphers require, perfect synchronism between the encrypting and the decrypting device. When some bits are lost or intentionally added during transmission, then the key stream produced at the receiving site will be combined with some lag or advance to the cipher text, causing a new effective encryption of the cipher text a second time. To re-establish synchronism at the receiver usually involves searching over all possible offsets that the key stream could have experienced, or notifying the sender that the cryptosystem should be reinitialized. In both cases, large portions of the transmitted data may be lost. This difficulty of setting up again the synchronism is considered to be the major disadvantage of synchronous stream ciphers.

In most stream ciphers, if a bit of cipher text is lost, or an extra bit is somehow inserted, synchronization will be lost and subsequent deciphered data will be incorrect. However, garbling from the error position through the end of the message can also occur from a cipher text value error when using a synchronous stream cipher which has forward data diffusion.

On the other hand, self-synchronizing stream ciphers protect against cipher text searching, but because of their self-synchronizing property they do not (or only slightly) protect against injections, deletions and replay of cipher text. In general, self synchronizing stream ciphers offer a lower level of security since arguments and function values of the key stream generating process are known to the cryptanalyst. The real disadvantage of self-synchronizing stream ciphers, however, is their limited analyzability because of the dependence of the key stream on the message stream.

Self-synchronizing stream ciphers have some limited error propagation which may or may not be viewed as an advantage. Certainly any changes made by an attacker to the cipher text will have additional consequences on other parts of the decrypted plaintext.

There are two main drawbacks of self-synchronizing stream ciphers: 1) Self-synchronizing stream ciphers are also vulnerable to a playback attack. First the attacker records some cipher text bits. Then, at a later time, he substitutes this recording into current traffic. After some initial garbage while the receiving end

resynchronizes, the old cipher text will decrypt as normal. The receiving end has no way of knowing that this is not current data, but old data being replayed. Unless timestamps are used, the attacker can convince a bank to credit his account again and again, by replaying the same message (assuming the key hasn't been changed, of course). Other weaknesses in this type of scheme could be exploited in the cases of very frequent resynchronization. 2) An opponent willing to intercept and decrypt an enciphered message may find easier to guess the next state of the PRNG since its input is taken partially from the ciphertext.

However, self-synchronizing stream ciphers, when realized with nonlinear key stream functions, have the advantage of combining secrecy with the ease of synchronization. They are resistant against bit slips (insertion or deletion of digits). Single bit errors in the cipher text result in error bursts in the plaintext, which is a disadvantage for communication applications. On the other hand, this prevents active eavesdroppers from selective changes of single plaintext digits via the cipher text and thus provides a basic level of message authenticity; because any change of a ciphertext digit will not only induce the change of the corresponding retrieved plaintext digit (after legitimate decoding), but will cause an error burst of the succeeding digits, revealing to the legitimate receiver that the decoded message is not faithful because an opponent or just a fortuitous transmission error has corrupted it.

SUMMARY OF THE INVENTION

The present invention uses a new method and system that will reduce or eliminate the deficiencies of current tools.

The object of the present invention is to propose a new type of synchronism of stream ciphers. The synchronization action is produced at random intervals, in an unpredictable time schedule; being impossible to detect by any unauthorized observer not possessing the key. The invention, advantageously, provides a manner to maintain the sender and receiver, of an encrypted transmission, exactly in step, avoiding the lost of information when a disruption of the stream of information

between them is produced, either by noise or by intentional perturbation of an opponent.

5 The occurrence of the synchronism action is induced by the spontaneous occurrence of a succession of several predetermined digits in the cipher text sequence; when this succession of digits is observed the synchronization is performed, by changing the state of the PRNG that generated the key stream, being the new state a non linear function of a set of previously transmitted digits. Moreover, the scheme may increase the security of the encrypted transmission by
10 increasing the size of the system key.

In a first aspect, it is presented a method

A method for improving stream cipher encryption of data, where the data to be encrypted, called plaintext, is combined with a random sequence of digits, called key
15 stream, to obtain the encrypted plain text called cipher text, being the key stream generated by a pseudo random number generator, PRNG as a function of a certain key sequence and the current state of the generator, the digits being blocks of s bits, being s a design parameter, the method comprising the following steps:

20 a) Determining the initial state of the pseudo random number generator and, resetting to the zero state the contents of a first and a second delay line (13a, 14a) of the cipher text of a first and a second register

b) Obtaining the first key, K_1 , which will be used as the key of the PRNG

c) Obtaining a second key K_2 , which will be a sequence of l digits, being l a design
25 parameter.

d) Delaying in the first delay line the current sequence of the cipher text k digits and storing in the first register, the last l digits of the delayed cipher text, being k a design parameter, being $k \geq 0$.

e) Comparing K_2 with the sequence of l bits of the delayed cipher text stored in step

30 d)

f) If both sequences do not match, generating the next state of the PRNG as a function of the first key and the previous state and going to step j)

g) If both sequences match, delaying in the second delay line the cipher text j digits and storing in the second register the last m digits of the delayed cipher text, being j and m design parameters, being $j \geq 0$

h) Applying a transfer function to the sequence of m digits stored in step g) to obtain a transformed sequence

i) Generating a load order to the PRNG to change the state of the PRNG to a forced value given by the transformed sequence

j) The PRNG generating one output key stream digit as a function of the first key and the current state of the PRNG

k) Reading 1 digit of the plain text

l) Combining the digit of plaintext digit and the key stream digit generating as a result one digit of the cipher text, the combination being a XOR operation of the plaintext digit and the key stream digit

m) Storing the obtained cipher text digit and adding it to the previous cipher text sequence, the obtained cipher text sequence will be the current cipher text sequence

n) Going to step d)

In a second aspect, it is presented a system

A system for improving stream cipher encryption and decryption of data, the system comprising a ciphering device and a deciphering device, where in the ciphering device the data to be encrypted, called plaintext, is received and combined with a random sequence of digits, called key stream, to obtain the encrypted plain text called cipher text, and in the deciphering device the text to be decrypted, cipher text, is received and combined with the key stream to obtain the decrypted text, plaintext, the digits being blocks of s bits;

The ciphering device comprising:

- A pseudo random number generator, PRNG being the key stream generated by the PRNG as a function of a key $K1$ and the current state of the generator and being the next state of the generator determined by $K1$ and the previous state if there is not a forced value for the next state

- A combiner which reads a digit of the plain text, reads a digit of the key stream, combines both digits and generates as a result one digit of the cipher text, the combination being a XOR operation of the plaintext digit and the key stream digit
- A first delay line which delays the cipher text sequence k digits, being k a design parameter, $k \geq 0$.
- A second delay line which delays the cipher text sequence j digits, being j a design parameter, $j \geq 0$.
- A first memory register which stores the last l digits of the cipher text sequences delayed by the first delay line, being l a design parameter, the register being updated each time a new digit of cipher text is generated
- A second memory register which stores the last m digits of the cipher text sequences delayed by the second delay line, being m a design parameter, the register being updated each time a new digit of cipher text is generated
- A transfer module which applies a transfer function to the sequence of m digits stored in the second memory register, generating a transformed sequence.
- A comparator which, after each new generated digit of the cipher text, compares the sequence of l digits stored in the first memory register with a second key K2 which is a sequence of l digits; and if both sequences match, generates a load order to the PRNG to change the state of the PRNG to a forced value given by the transformed sequence generated by the transfer module

The deciphering device comprising:

- A pseudo random number generator, PRNG being the key stream generated by the PRNG as a function of the key K1 and the current state of the generator and being the next state of the generator determined by K1 and the previous state if there is not a forced value for the next state
- A combiner which reads a digit of the cipher text, reads a digit of the key stream, combines both digits and generates as a result one digit of the decrypted cipher text, that is the plaintext, the combination being a XOR operation of the cipher text digit and the key stream digit
- A first delay line which delays the cipher text sequence k digits.
- A second delay line which delays the cipher text sequence j digits

- A first memory register which stores the last l digits of the cipher text sequences delayed by the first delay line, the register being updated each time a new digit of cipher text is received
 - A second memory register which stores the last m digits of the cipher text sequences delayed by the second delay line, being m a design parameter, the register being updated each time a new digit of cipher text is received
 - A transfer module which applies a transfer function to the sequence of m digits stored in the second memory register, generating a transformed sequence.
 - A comparator which, after each new received digit of the cipher text, compares the sequence of l digits stored in the first memory register with the second key K_2 ; and if both sequences match, generates a load order to the PRNG to change the state of the PRNG to a forced value given by the transformed sequence generated by the transfer module
- Finally, a computer program comprising computer program code means adapted to perform the above-described method is presented.

For a more complete understanding of the invention, its objects and advantages, reference may be had to the following specification and to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

To complete the description and in order to provide for a better understanding of the invention, a set of drawings is provided. Said drawings form an integral part of the description and illustrate a preferred embodiment of the invention, which should not be interpreted as restricting the scope of the invention, but rather as an example of how the invention can be embodied. The drawings comprise the following figures:

Figure 1 represents a block diagram of a synchronous stream cipher and decipher system.

Figure 2 represents a block diagram of a self-synchronizing stream cipher and decipher system.

Figure 3 represents a block diagram of the cipher end of the mechanism of random-synchronism of stream ciphers.

5 Figure 4 represents a block diagram of the decipher end of the mechanism of random-synchronism of stream ciphers.

Figure 5 represents a flowchart of the mechanism of random-synchronism of stream ciphers.

10

Figure 6 represents a block diagram of an embodiment of the invention using a Linear Feedback Shift Register as pseudorandom number generator.

15

Figure 7 represents a block diagram of an embodiment of the invention using block cipher TDEA in Counter Mode as pseudorandom number generator.

Corresponding numerals and symbols in the different figures refer to corresponding parts unless otherwise indicated.

20

DETAILED DESCRIPTION OF THE INVENTION

25

A new method and system for synchronizing stream ciphers is proposed, called random-synchronism of stream ciphers. Oppositely to the synchronous stream ciphers, which are synchronized only once at the beginning of the transmission and to the self-synchronizing stream ciphers, which are synchronized whenever a digit is encrypted; the new method perform a synchronization operation at random intervals, in an unpredictable time schedule.

30

In the proposed invention the decision to carry out the synchronization of ciphering and deciphering machines is taken upon occurrence of a predetermined string of cipher text digits. Such string of digits may be pre-selected at random, and will form part of the system key. But its length must be selected in function of the desired mean frequency of synchronization. Moreover, the scheme may increase the security of the encrypted transmission by increasing the size of the system key.

A necessary condition for the security of an encryption procedure is that the cipher text sequence is completely random. Hence, the occurrence of any predetermined string of bits in the cipher text sequence is entirely possible. The mean frequency of the occurrence of such string depends on its bit length, for statistical reasons, being its probability of occurrence inversely proportional to its length: strings of few bits will occur frequently and strings of many bits will occur rarely. A string of l bits of length happens every 2^l bits, in average, in any random bit sequence; hence the mean probability of the spontaneous occurrence of any predetermined string of l bits of length, inside a random sequence of bits, is equal to:

$$P_l = 1 / 2^l \quad (1)$$

Not all the spontaneous occurrences of a predetermined string will happen with equal frequency inside a random sequence of bits; the distance k , in bit places, between their occurrences may vary from a distance value of $k = 0$, to big distances, being the growing distances progressively less probable. The probability of occurrence of two equal strings of length l with a separation of k bit places is:

$$P_{l,k} = (1 / 2^l) \cdot (1 / 2^k) \cdot (1 - 1 / 2^l)^k. \quad (2)$$

Very big distances between equal strings seldom occur. The total joint probability of occurrences of equal strings of length l , with distances $k > d$ is:

$$\sum P_{l,d} = (1 / 2^l) \cdot (1 - 1 / 2^l)^d = P_l \cdot (1 - 1 / 2^l)^d. \quad (3)$$

As an example, suppose that $l = 8$, hence $2^l = 256$, the probability of occurrences of strings with a separation $k > 256$ is: $\sum P_{8,256} < 0.0014$.

The synchronism is executed each time that the spontaneous occurrence of a string of a length of l predetermined digits is observed; when this succession of cipher text digits happens the synchronization is performed, by changing the state of the PRNG that generated the key stream, being the new state a non linear function of a set of previous transmitted digits.

Figure 3 depicts the block diagram of the mechanism of random-synchronism of stream ciphers, at the sender location; i.e. the cipher end; new blocks specific of this invention, not used in synchronous stream ciphers nor in self-synchronizing stream ciphers, are distinguished with a discontinuous border. As in any standard stream

cipher, there is a plaintext (3a) that must be encrypted into the cipher text (5a), by the combination (4a) of the plaintext bits with the key stream bits generated by the PRNG (2a); this PRNG generates a key stream which is a complicated function of the system key $K1$, stored in the register (1a) and the sequence of N digits stored in the register (11a). The combination (4a) consists of an XOR, if both the plaintext and the key stream are composed of independent bits, the XOR operation is accomplished bit by bit; but if both the plaintext and the key stream are composed of digits formed by blocks of several bits, the XOR operation is made bitwise. The synchronization mechanism is achieved by the rest of elements of the Fig. 3, in the following way:

The cipher text (5a), besides being sent to the receiver end, is also routed in parallel to the two delay lines (13a) and (14a), whose delay lengths k and j can be comprised between 0 and any value. The cipher text, after passing through the delay lines (13a) and (14a), arrives at two memory registers (15a) and (16a), with a capacity of l digits and m digits, respectively. The delay length values k and j must be selected in such a way that the stored digits in the register (15a) and the register (16a) do not coincide.

The contents of registers (15a) and (16a) are updated each time a new digit of cipher text is generated. This cipher text digits enters into the delay lines (13a) and (14a) substituting the previous entered digit. The last digit of each delay line (13a) and (14a) comes out and enters into the registers (15a) and (16a) respectively.

After each new generated digits of cipher text, the contents L of the register (15a) are compared with the key $K2$, stored in the register (17a), the comparison is made by the comparator (18a).

If the $L \neq K2$, i. e. the contents of both registers (17a) and (15a) do not match, no action is taken; hence, the PRNG (2a) continues unperturbed generating new key digits. But if the contents of registers (17a) and (15a) do agree, $L = K2$, the comparator (18a) sends a load order, to the PRNG (2a), of jumping forcefully to a new state. The value of the new state of the PRNG (2a) is loaded from the output of the transfer function state (11a), which is a transformation $q(M)$ of the string of cipher text M , stored contents in the register (16a). Such transformation can be any

function, it can be fully defined or may be dependent of a third key $K3$ (19a); and such function can also be an identity.

Figure 4 depicts the block diagram of the mechanism of random-synchronism of stream ciphers, at the receiver location, i.e. the decipher end; new blocks specific of this invention, not used in synchronous stream ciphers nor in self-synchronizing stream ciphers, are distinguished with a non discontinuous border. The block diagram is similar to the diagram of the cipher end, but there are three differences. First, the block (3b) consists of the incoming cipher text. Second, the block (5b) consists of the recovered plaintext, which is the result of the combination (4b) of the cipher text digits with the key stream digits generated by the PRNG (2b). Third, the input connection of both delay lines (13b) and (14b) is made before the combination (4b), by means of a XOR, of the cipher text (3b) digits and the key stream bits generated by the PRNG (2b). The rest of the elements of block diagram are equal to the ones in the cipher end, i.e. delay lines 13b and 14b which delay the cipher text k and j digits respectively; registers 15b and 16b to store the delay signals; a comparator 18b which compares the first delay signal with $K2$ stored in register 17b and sends a load order of jumping to a new state to the PRNG if the two signals agree. Said new state is loaded from the output of the transfer function state 11b and may be dependent of a third key $K3$ stored in a register (13b).

Figure 5 illustrates the flowchart of the mechanism of random-synchronism of stream ciphers; steps 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29, are the normal steps of any synchronous stream cipher. The rest are the new steps which are added to convert it in a stream cipher with random-synchronism.

Step 20 consists of initializing the PRNG, determining an initial state with the seed i . Step 21 consists of reading from the store 22 the key $K1$. Step 23 consists of advancing undisturbed to the next state of the PRNG each time that key stream digit is going to be generated according to the state transition function $f(K1, S)$ which controls it. Step 24 consists of generating one output key stream digit applying the function g to the key $K1$ and the state generated. Step 25 consists of reading one digit of plaintext from the store 26. Step 27 consists of combining by means of the XOR operation the plaintext digit read in step 25 with the key stream digit generated in the step 24. In the step 28 the cipher text is obtained and stored in the store 29.

The steps pertaining to the mechanism of random-synchronism of stream ciphers are described next and are shown in figure 5 with a discontinuous border.

In order to initially synchronize the cipher and decipher mechanisms, the step 20 besides initializing the PRNG in an initial state with the seed i , it resets to the zero state the contents of the delay lines (13a and 14a) and correspondent registers (15a and 16a) of the cipher end as well as the delay lines (13b, 14b) and registers (15b and 16b) of the decipher.

Step 30 consists of reading the key $K2$, of l digits of size, from the store 31. Step 32 consists of reading from the store 29 a string of l digits of cipher text L , (after a delay of k digits). Step 33 consists of comparing the key $K2$ with the delayed cipher text L ; here a decision is taken: if both quantities are different, the system continues to step 23 and following; but if both quantities do agree the system continues to step 34.

Step 34 consists of reading a string of m digits of cipher text M , from the store 29, after a delay of j digits. Step 35 consists of transform the m digits of cipher text M to obtain the future next state $q(M)$ of the PRNG. Step 36 consists of forcing the next state of the PRNG to adopt the value of the transformed function $q(M)$. Then the system continues to step 24 and following.

There are many possible embodiments of the invention, depending on the PRNG used and the transfer function of the mechanism of random-synchronism of stream cipher. As an example two distinct embodiments are presented.

Stream cipher using a Linear Feedback Shift Register as PRNG

The block diagram of the proposed embodiment is illustrated in Fig. 6, which is similar to the block diagram depicted in Fig. 3. The numbering of blocks is the same in both figures, but instead of the suffix "a" used in Fig. 3, in Fig. 6 the suffix "c" is used.

The PRNG (2c) consists now of a Linear Feedback Shift Register (LFSR), with a length 127, and characteristic polynomial $x^{127} + x^{63} + 1$ (the bigger exponent is called

the degree of the polynomial). The key $K1$ consists of the initial state value of the shift register. The value of the exponents of the characteristic polynomial can be interpreted as a structural key, which is not changed frequently; it should be selected as to guarantee the maximum allowable repetition period, what is obtained selecting a primitive irreducible polynomial.. In the present embodiment, to obtain the maximum computing efficiency, the selected polynomial is just a trinomial, precisely a primitive one whose degree p is a Mersenne–prime (all polynomials of Mersenne–prime degree are irreducible).

The selected length of the delay line equivalent to (13a) is $k = 0$, i.e. it does not exist. The length of the register (15c), of the comparator (18c) and of the key $K2$ is $l = 6$ bits. The length of the delay line (14c) is $j = 6$ bits, in this way it is assured that the contents L and M of the registers (15c) and (16c) correspond to different parts of the cipher text. The length of the register (16c) is 128 bits. Each time that the contents L of the register (15c) and the key $K2$ coincide, the comparator (18c) generates a new state load order for the PRNG.

The transfer function (11c) consists of an AES block cipher —whose block length is 128 bits—, and its key $K3$, stored in the register (19c), is selected with a length of 256 bits. The output of the transfer function $AES_{K3}(M)$ is loaded into the LFSR of the PRNG, as a new state forced value, each time that the comparator (18c) generates a new state load order. As the output of the AES is a block of 128 bits and the LFSR is 127 bits in length, the most significant bit of the output block of the AES is discarded, prior to loading it into the LFSR.

A PRNG made with a LFSR has perfect randomness properties but unfortunately is not secure, as it should be. The form of breaking a stream cipher, which uses a LFSR as PRNG, consists of determining the value of the characteristic polynomial of the LFSR and its present state. To determine them, it suffices to know a fragment of the generated key stream, of length double than the length of the shift register. Such fragment of the key stream can be learned by an opponent by means of a “known plaintext attack”. But, when adding the mechanism of random synchronism, object of this invention, this kind of attack is fruitless, because an opponent cannot have access to the undisturbed string of 254 bits, necessary to mount the attack. Due to

the random occurrence of the coincidences between the key $K2$ and the contents L of the register (15c), the mean distance between occurrences is $2^l = 64$ bits; but according to Eq. (3) the proportion of occurrences with a separation greater than 254 bits is only $1 \cdot 10^{-3}$ of the total occurrences. Hence, an opponent will have a very low chance of finding an undisturbed string of 254 bits, necessary to determine the characteristic polynomial and the state of the LFSR. Moreover, if she could succeed in finding the characteristic polynomial and the state LFSR, she cannot profit of it, because the state of the LFSR will change very soon to a new unexpected forced value.

The LFSR would generate, if undisturbed, an ideal key stream of $2^{127}-1$ bits of length. The net effect of the invention is equivalent to extracting different segments, at random, of arbitrary length, from the aforesaid ideal key stream, to build another key stream, chaining them together arranged in a completely new random order. The proposed embodiment has a strength much higher than the original stream cipher without random synchronism, because besides knowing the initial state of the shift register and characteristic polynomial, the attacker should know the keys $K2$ and $K3$ to break it.

Stream cipher using the block cipher TDEA in Counter Mode as PRNG

The block diagram of the proposed embodiment is illustrated in Fig. 7, which is similar to the block diagram depicted in Fig. 3. The numbering of blocks is the same in both figures, but instead of the suffix "a" used in Fig. 3, in Fig. 6 the suffix "d" is used.

The PRNG (2d) consists now of a block cipher TDEA in Counter Mode, being its block length 64. The key (1d) is composed by the Counter initial state i and by the $K1$ of the TDEA block cipher. The Counter is a simple 64 bit binary counter. The selected length of the delay line equivalent to (13a) is $k = 0$, i.e. it does not exist. The length of the register (15c), of the comparator (18c) and of the key $K2$ is $l = 8$ bits. The length of the delay line (14c) is $j = 8$ bits and the length of the register $m=64$ bits, in this way it is assured that the contents L and M of the registers (15c) and (16c) correspond to different parts of the cipher text.

Each time that the contents L of the register (15c) and the key $K2$ coincide, the comparator (18c) generates a new state load order for the Counter of the PRNG, which causes that the Counter changes its count, jumping to a new count of completely random value, higher or lower than the original one. The transfer function equivalent to (11a) is just an identity, i.e. it does not exist. The content of the register (16d) is loaded into the counter of the PRNG, as a new state forced value, each time that the comparator (18c) generates a new state load order. As a consequence of no existence of the transfer function there is neither any $K3$ key.

A PRNG made with a TDEA in Counter Mode generates a random permutation, quite different from a pseudorandom sequence, because pseudorandom sequences show a great deal of repeated numbers (collisions), while random permutations have no collisions, i.e. all the generated numbers are different. To avoid a distinguishing attack against a PRNG made with a TDEA in Counter Mode, the length of the sequence generated should be limited to a size much smaller than the "birthday bound" —which in the case of a 64 bits block cipher is just 2^{32} bits—. But, when adding the mechanism of random synchronism, object of this invention, a PRNG made with a TDEA in Counter Mode do generate authentic pseudorandom sequences. The reason is that the net effect of the invention is equivalent to extract different segments, at random, of arbitrary length, from the aforesaid random permutation, to build another different key stream, chaining them together arranged in a completely new random order; this new key stream will have now collisions, due to the possibility of having selected twice (or more times) some portions of the original random permutation. Hence, the length of the usable pseudorandom sequence can be as large as desired, because there is no more birthday bound limit. The length of the generated sequence can be even larger than the period of repetition of the undisturbed counter, i. e. in the present embodiment can be larger than 2^{64} bits.

The proposed embodiment has strength much higher than the original stream cipher without random synchronism, because besides knowing the initial state i of the Counter and the key $K1$, the attacker should know the key $K2$ to break it.

Summarizing, the advantages of the present invention are:

- 5 • The mechanism of random–synchronism of stream ciphers allows the synchronism of a stream cipher to take place at random points of time, hence an opponent cannot deduce the moment when the synchronism takes place, enhancing the system security.
- Adding a new key $K2$, the security of the system is enhanced, because an opponent, when performing a brute force attack (trying all possible key values), will need to try 2^l times more key values than were necessary without the random–synchronism mechanism.
- 10 • The transfer function $q(M)$ may incorporate a third key $K3$ in its design, allowing a supplementary increase of security, because of the increase in difficulty of a brute force attack. In such a case, an opponent will need to try $2^{(l+m)}$ times more key values than were necessary without the random–synchronism mechanism.
- The transfer function $q(M)$ may convert an insecure stream cipher, which makes use
15 of a predictable PRNG (as a Feedback Linear Shift Register), in a secure stream cipher because the resulting key stream is no longer predictable due to the random forced jumps of the generator.
- The PRNGs of some stream ciphers do not generate an authentic pseudorandom key stream, but simply a key–depending permutation of all the possible states of the
20 generator. This is the case of a block cipher working in the *Counter Mode* or a Linear Congruential Generator. It is possible to recover the key stream of such stream ciphers by means of a known plaintext attack, and then to mount a distinction attack against the PRNG, observing that the generated key stream sequence shows no collisions (repetition of values), as a true pseudorandom
25 sequence must show. The combination of these two attacks may allow an opponent to guess, with some probability of success, future values of the key stream.
- The use of the mechanism of random–synchronism converts the key stream of any stream cipher, built with a PRNG which generates a key depending permutation of all the possible states, in a true pseudorandom key stream, with collisions; hence,
30 preventing the previously described attack.
- Playback attacks are noticeable and easily detected. With the use of the random–synchronism mechanism, when an opponent records some cipher text bits and, at a later time, he substitutes this recording into current traffic, he induces much longer noticeable garbage in the recovered plaintext (while the receiving end

resynchronizes) than in the case of using a self-synchronizing stream cipher, allowing the detection of the attack. This is because the random-synchronism does not take place every cipher text digits (as in a self-synchronizing system) but each 2^l cipher text digits.

5

Although the present invention has been described with reference to specific embodiments, it should be understood by those skilled in the art that the foregoing and various other changes, omissions and additions in the form and detail thereof may be made therein without departing from the spirit and scope of the invention as defined by the following claims.

10

CLAIMS

1. A method for improving stream cipher encryption of data, where the data to be encrypted, called plaintext, is combined with a random sequence of digits, called key stream, to obtain the encrypted plain text called cipher text, being the key stream generated by a pseudo random number generator, PRNG as a function of a certain key sequence and the current state of the generator, the digits being blocks of s bits, being s a design parameter, the method comprising the following steps:
- 5
- 10 a) Determining the initial state of the pseudo random number generator and, resetting to the zero state the contents of a first and a second delay line (13a, 14a) of the cipher text of a first and a second register
- b) Obtaining the first key, $K1$, which will be used as the key of the PRNG
- c) Obtaining a second key $K2$, which will be a sequence of l digits, being l a design parameter.
- 15
- d) Delaying in the first delay line the current sequence of the cipher text k digits and storing in the first register, the last l digits of the delayed cipher text, being k a design parameter, being $k \geq 0$.
- e) Comparing $K2$ with the sequence of l bits of the delayed cipher text stored in step
- 20 d)
- f) If both sequences do not match, generating the next state of the PRNG as a function of the first key and the previous state and going to step j)
- g) If both sequences match, delaying in the second delay line the cipher text j digits and storing in the second register the last m digits of the delayed cipher text, being j and m design parameters, being $j \geq 0$
- 25
- h) Applying a transfer function to the sequence of m digits stored in step g) to obtain a transformed sequence
- i) Generating a load order to the PRNG to change the state of the PRNG to a forced value given by the transformed sequence
- 30 j) The PRNG generating one output key stream digit as a function of the first key and the current state of the PRNG
- k) Reading 1 digit of the plain text

l) Combining the digit of plaintext digit and the key stream digit generating as a result one digit of the cipher text, the combination being a XOR operation of the plaintext digit and the key stream digit

m) Storing the obtained cipher text digit and adding it to the previous cipher text sequence, the obtained cipher text sequence will be the current cipher text sequence

n) Going to step d)

2. A method according to any of the previous claims where the initial state of the PRNG is given by an initial seed, i , which is a design parameter.

3. A method according to any of the previous claims where $s=1$, that is, the digits are independent bits.

4. A method according to any of the previous claims where if $s=1$ the XOR operation of step l) is made bit by bit and if $s>1$ the XOR operation is made bit by bit.

5. A method according to any of the previous claims where the steps of obtaining the first and second keys comprises the steps of reading the correspondent registers where the keys are stored.

6. A method according to any of the previous claims where the transfer function is dependent of a third key K_3 , stored in a register.

7. A method according to claims 1-5 where the transfer function is an identity.

8. A method according to any of the previous claims, wherein $k > j + m$.

9. A method according to claims 1-7 wherein $j > k + l$

10. A system for improving stream cipher encryption and decryption of data, the system comprising a ciphering device and a deciphering device, where in the ciphering device the data to be encrypted, called plaintext, is received and combined with a random sequence of digits, called key stream, to obtain the encrypted plain

text called cipher text, and in the deciphering device the text to be decrypted, cipher text, is received and combined with the key stream to obtain the decrypted text, plaintext, the digits being blocks of s bits;

5 The ciphering device comprising:

- A pseudo random number generator, PRNG being the key stream generated by the PRNG as a function of a key $K1$ and the current state of the generator and being the next state of the generator determined by $K1$ and the previous state if
10 there is not a forced value for the next state
- A combiner which reads a digit of the plain text, reads a digit of the key stream, combines both digits and generates as a result one digit of the cipher text, the combination being a XOR operation of the plaintext digit and the key stream digit
- A first delay line which delays the cipher text sequence k digits, being k a
15 design parameter, $k \geq 0$.
- A second delay line which delays the cipher text sequence j digits, being j a design parameter, $j \geq 0$.
- A first memory register which stores the last l digits of the cipher text sequences delayed by the first delay line, being l a design parameter, the register
20 being updated each time a new digit of cipher text is generated
- A second memory register which stores the last m digits of the cipher text sequences delayed by the second delay line, being m a design parameter, the register being updated each time a new digit of cipher text is generated
- A transfer module which applies a transfer function to the sequence of m
25 digits stored in the second memory register, generating a transformed sequence.
- A comparator which, after each new generated digit of the cipher text, compares the sequence of l digits stored in the first memory register with a second key $K2$ which is a sequence of l digits; and if both sequences match, generates a load order to the PRNG to change the state of the PRNG to a forced value given by
30 the transformed sequence generated by the transfer module

The deciphering device comprising:

- A pseudo random number generator, PRNG being the key stream generated by the PRNG as a function of the key K1 and the current state of the generator and being the next state of the generator determined by K1 and the previous state if there is not a forced value for the next state

5 - A combiner which reads a digit of the cipher text, reads a digit of the key stream, combines both digits and generates as a result one digit of the decrypted cipher text, that is the plaintext, the combination being a XOR operation of the cipher text digit and the key stream digit

- A first delay line which delays the cipher text sequence k digits.

10 - A second delay line which delays the cipher text sequence j digits

- A first memory register which stores the last l digits of the cipher text sequences delayed by the first delay line, the register being updated each time a new digit of cipher text is received

15 - A second memory register which stores the last m digits of the cipher text sequences delayed by the second delay line, being m a design parameter, the register being updated each time a new digit of cipher text is received

- A transfer module which applies a transfer function to the sequence of m digits stored in the second memory register, generating a transformed sequence.

20 - A comparator which, after each new received digit of the cipher text, compares the sequence of l digits stored in the first memory register with the second key K2; and if both sequences match, generates a load order to the PRNG to change the state of the PRNG to a forced value given by the transformed sequence generated by the transfer module

25 11. A computer program comprising computer program code means adapted to perform the method according to any claims from 1 to 9 when said program is run on a computer, a digital signal processor, a field-programmable gate array, an application-specific integrated circuit, a micro-processor, a micro-controller, or any other form of programmable hardware.

30

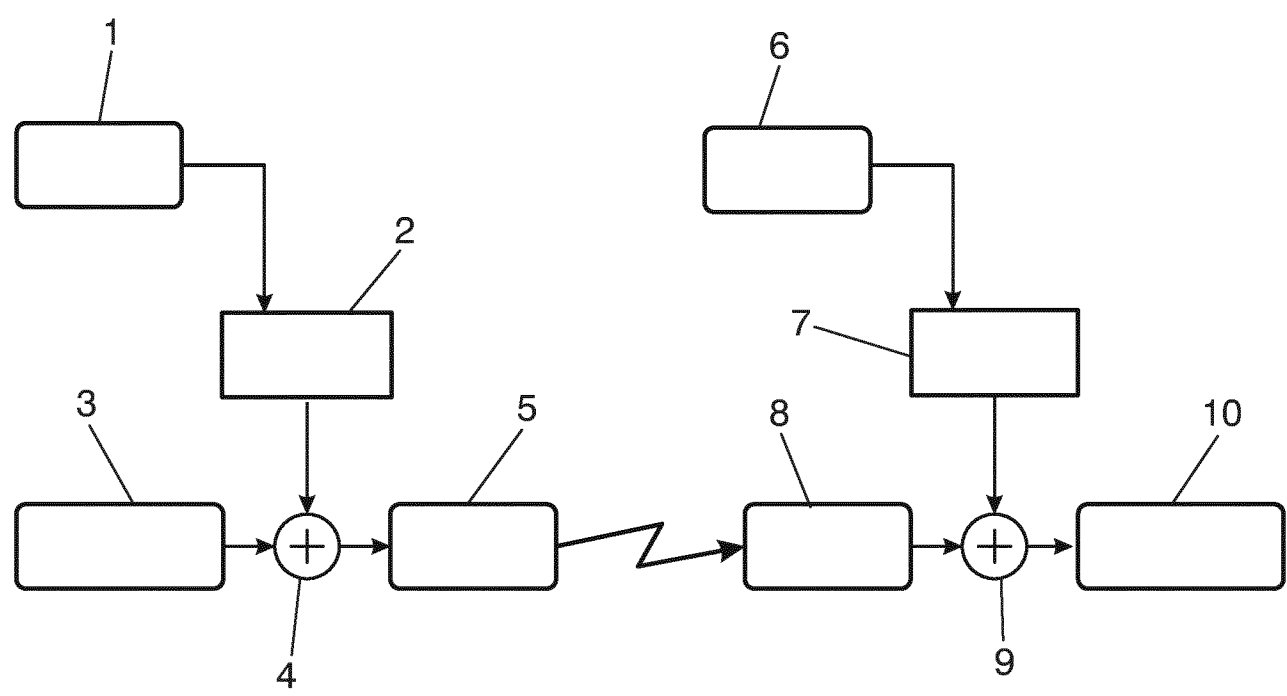
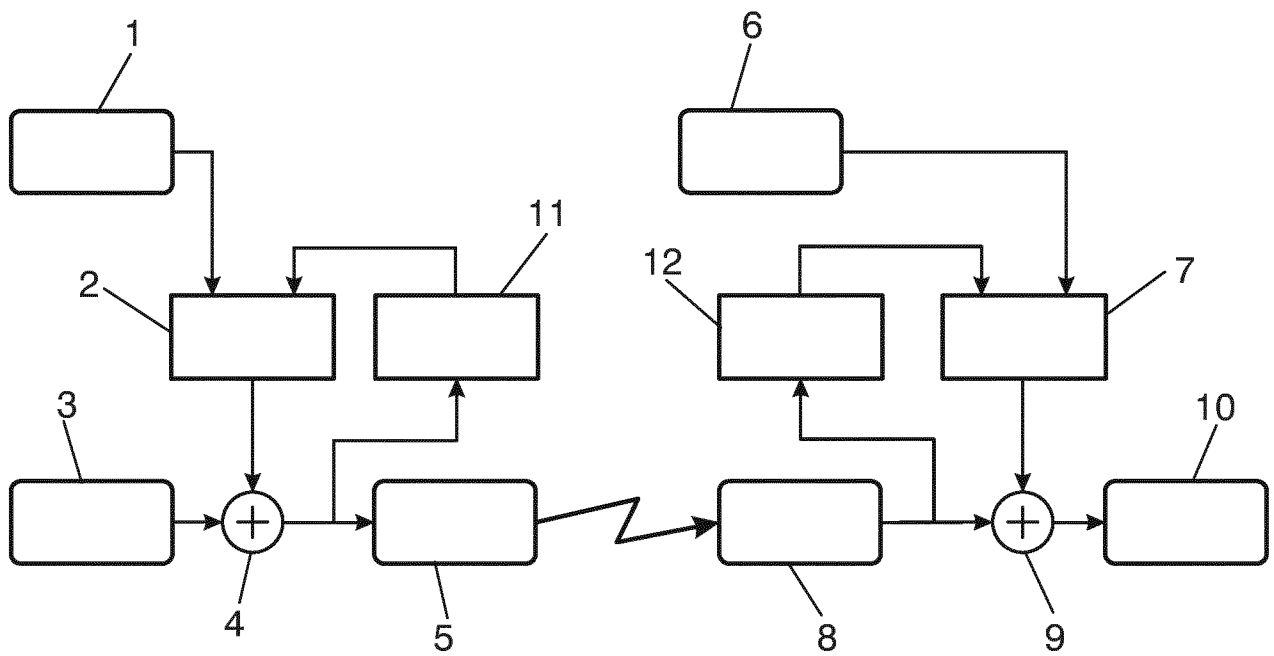
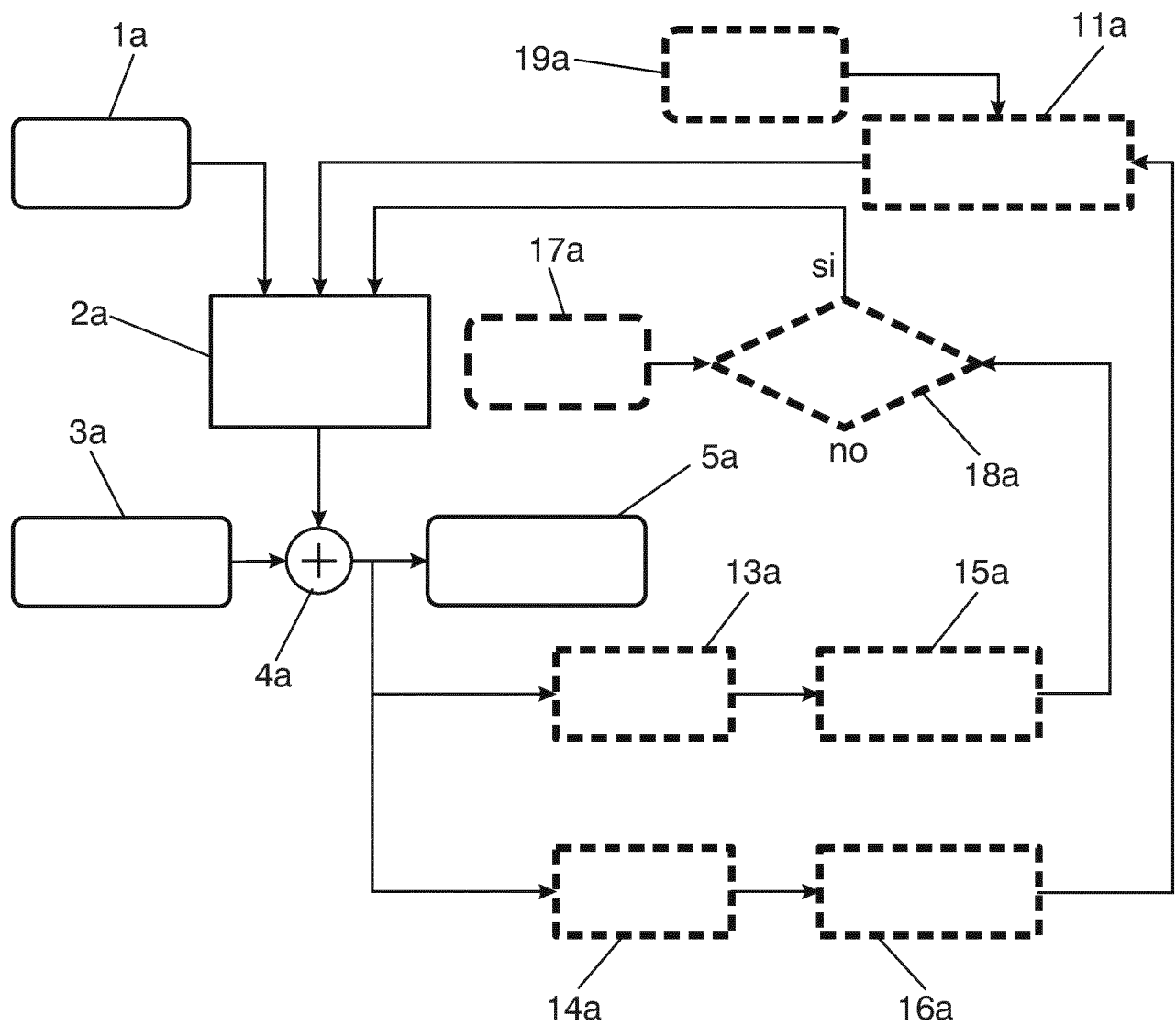


FIG. 1

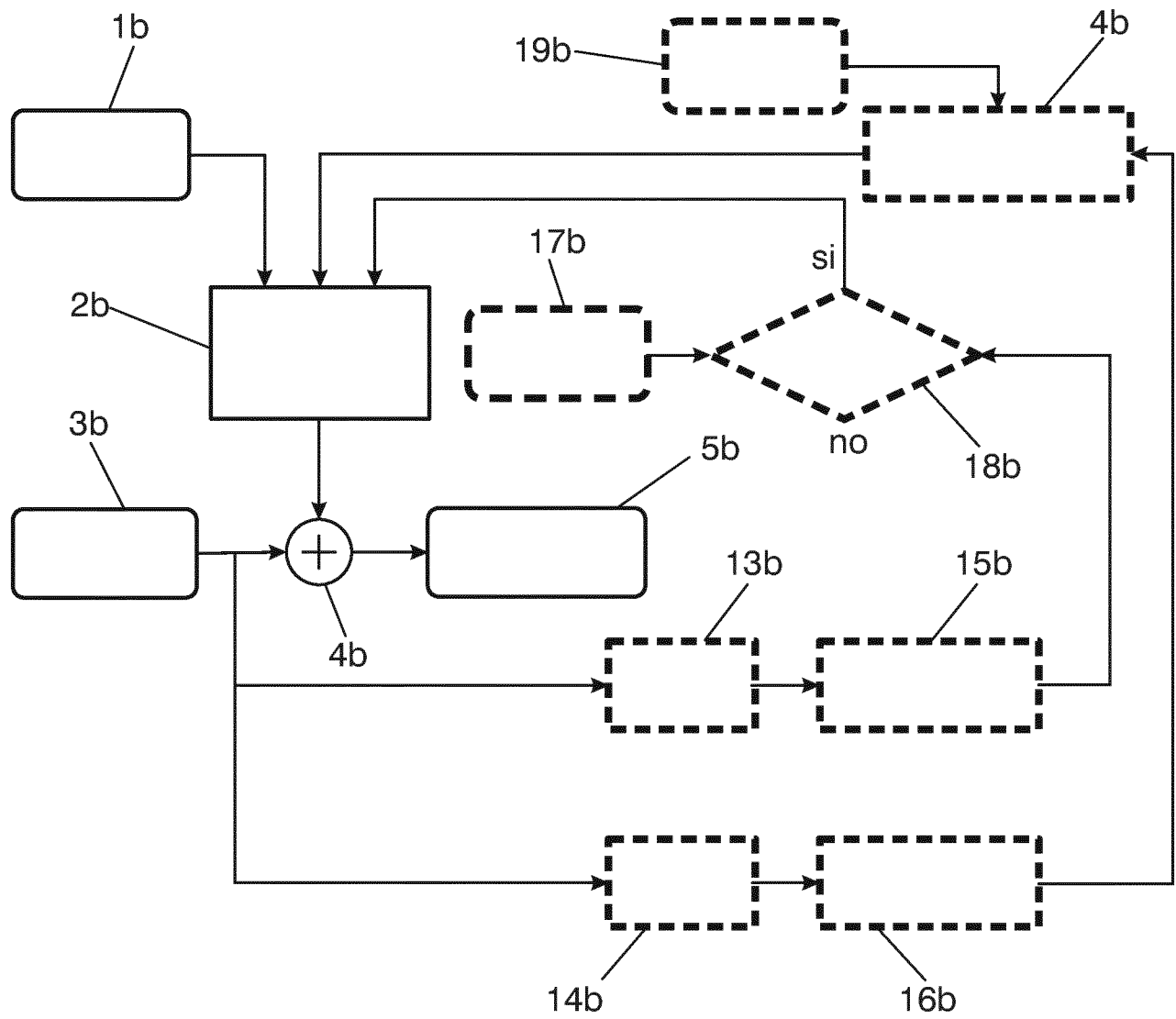
2/7

**FIG. 2**

3/7

**FIG. 3**

4/7

**FIG. 4**

5/7

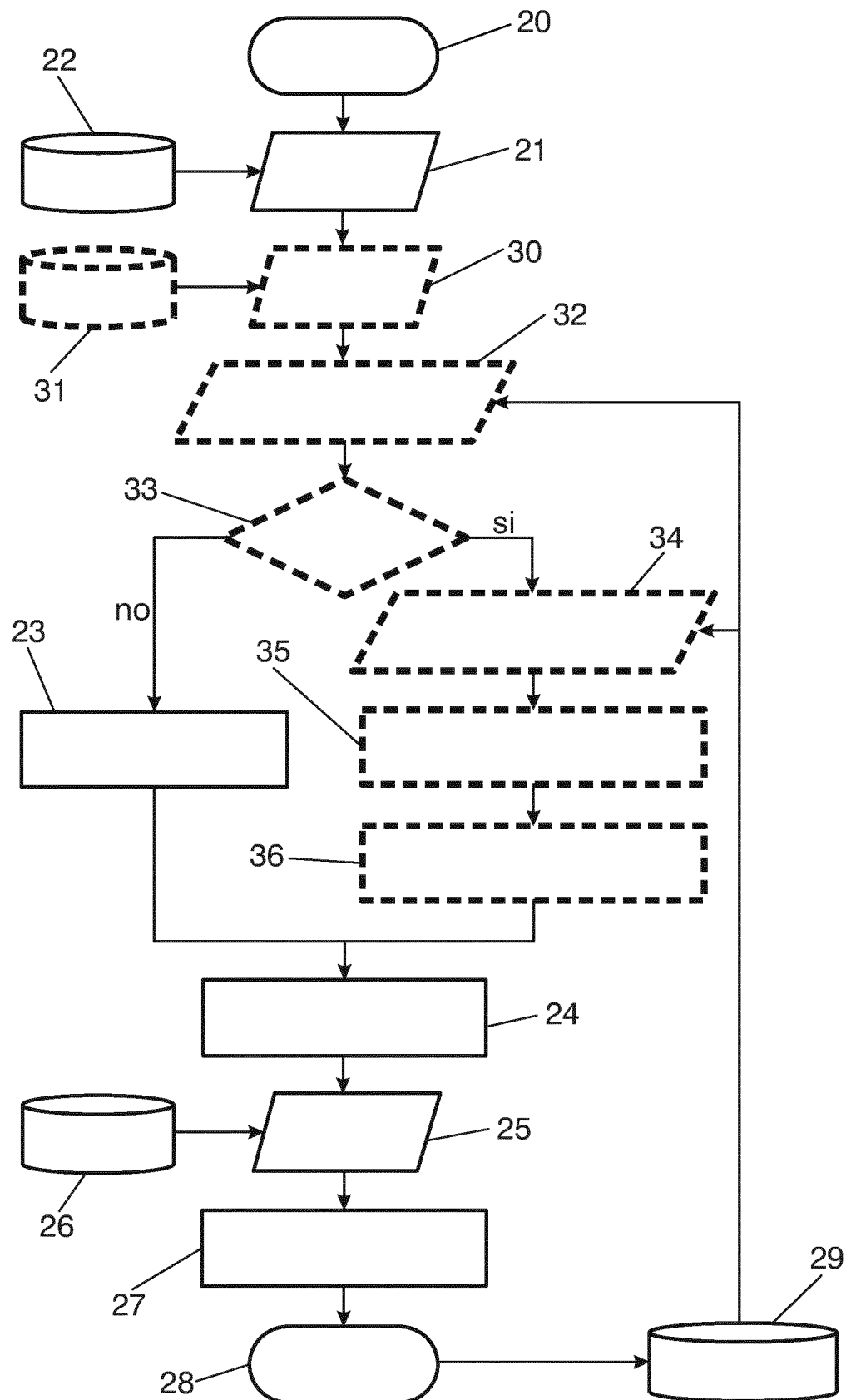
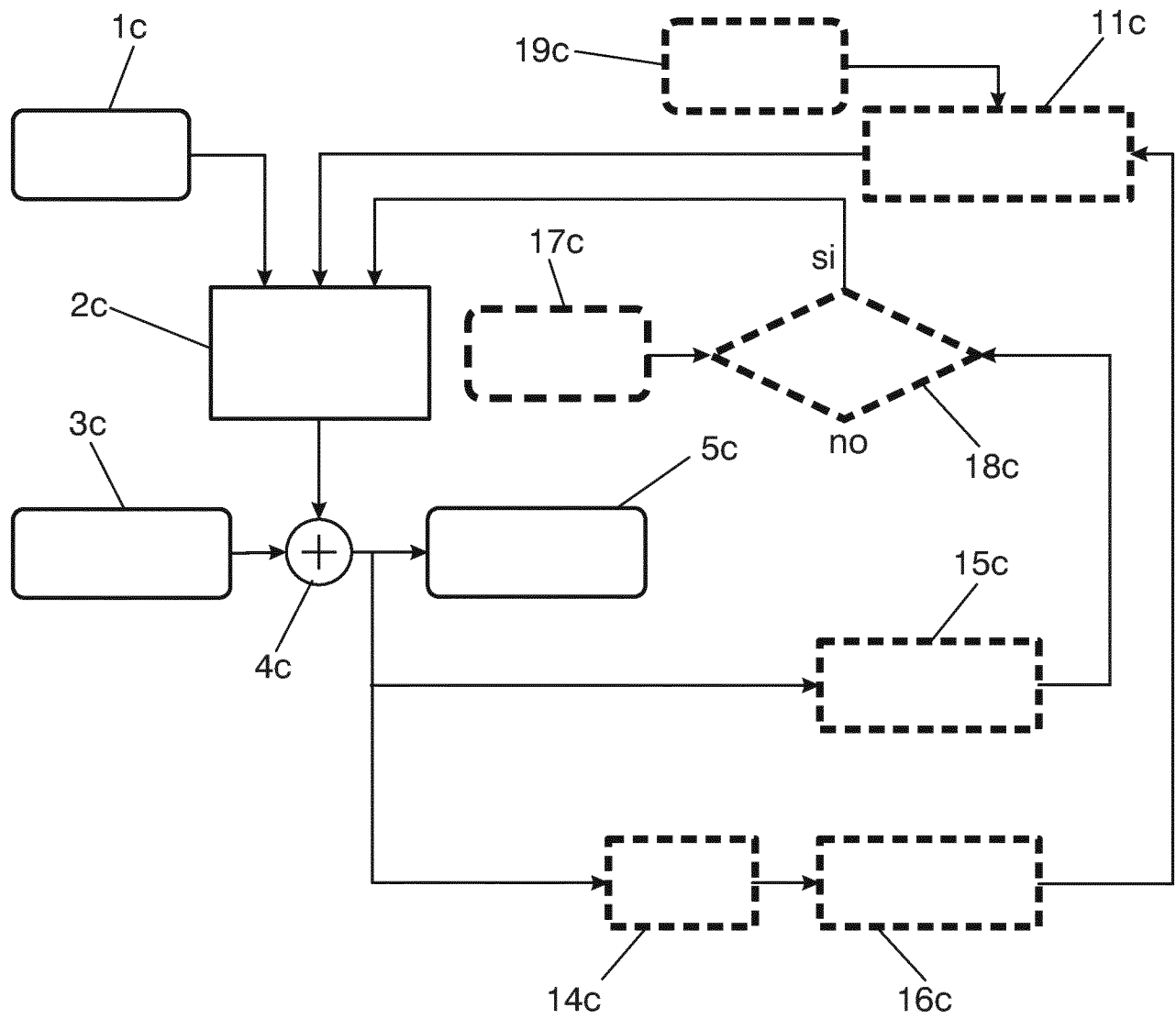
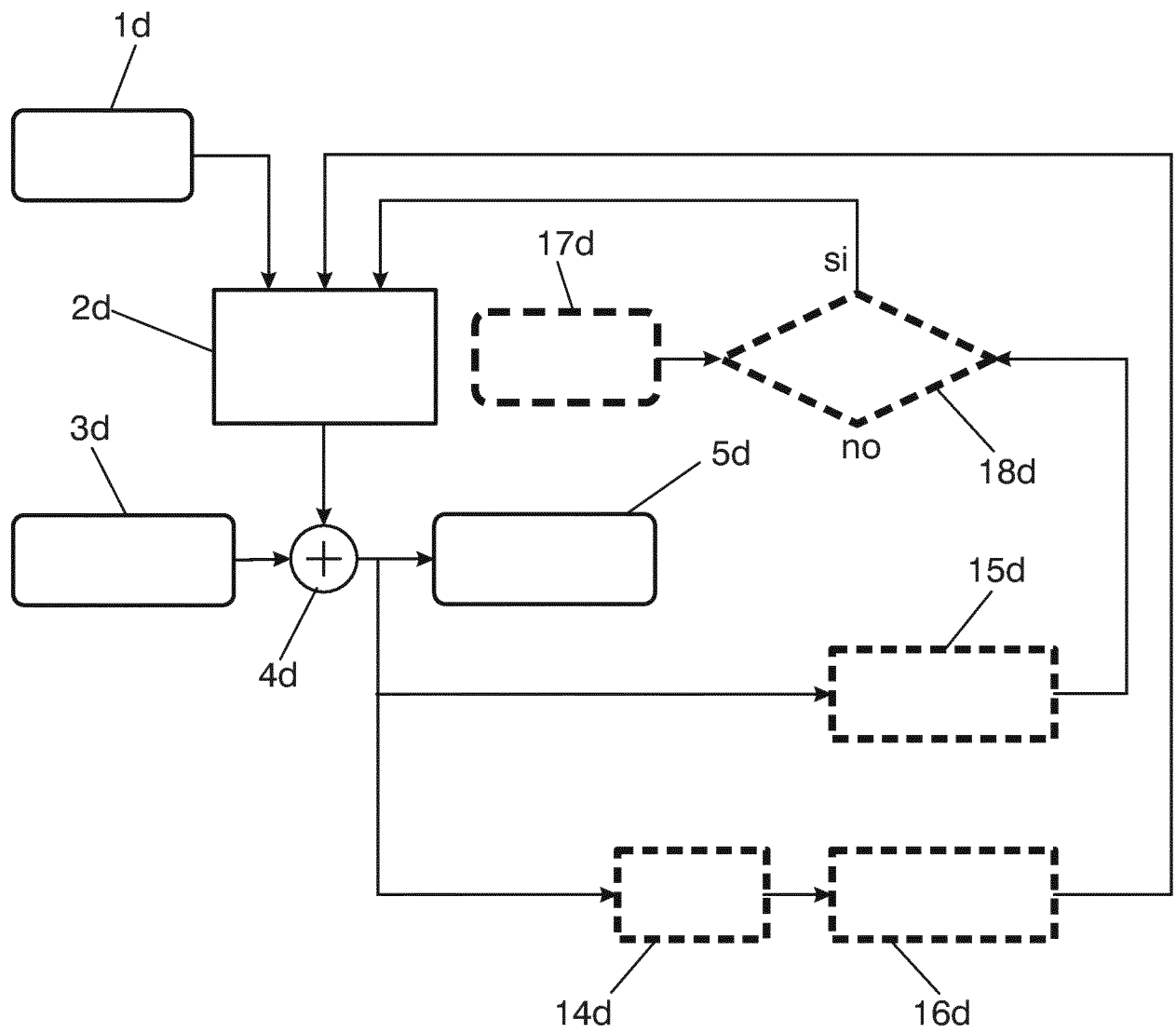


FIG. 5

6/7

**FIG. 6**

7/7

**FIG. 7**

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2012/056685

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/06
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)
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.
X	HEYS H M: "Analysis of the statistical cipher feedback mode of block ciphers", IEEE TRANSACTIONS ON COMPUTERS, IEEE SERVICE CENTER, LOS ALAMITOS, CA, US, vol. 52, no. 1, 1 January 2003 (2003-01-01), pages 77-92, XP011095420, ISSN: 0018-9340, DOI: 10.1109/TC.2003.1159755	1-5,7-11
Y	sections 3 and 7; figures 3,4 ----- -/--	6



Further documents are listed in the continuation of Box C.



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

17 September 2012

Date of mailing of the international search report

26/09/2012

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Billet, Olivier

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2012/056685

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	STIG F MJÄ LSNES ED - JEAN-JACQUES QUISQUATER ET AL: "A Simple Technique for Diffusing Cryptoperiods", 10 April 1989 (1989-04-10), ADVANCES IN CRYPTOLOGY - EUROCRYPT '89 : WORKSHOP ON THE THEORY AND APPLICATION OF CRYPTOGRAPHIC TECHNIQUES, HOUTHALEN, BELGIUM, APRIL 10 - 13, 1989; [LECTURE NOTES IN COMPUTER SCIENCE], BERLIN [U.A.] : SPRINGER, DE, PAGE(S) 110 - 120, XP019194424, ISBN: 978-3-540-53433-4 section 2 -----	1-5,7-11
Y	UELI M MAURER ED - DONALD W DAVIES: "New Approaches to the Design of Self-Synchronizing Stream Ciphers", 8 April 1991 (1991-04-08), ADVANCES IN CRYPTOLOGY EUROCRYPT 91, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 458 - 471, XP019194235, ISBN: 978-3-540-54620-7 sections 2 and 3; figures 1,2,3 -----	6
X	Ammar Alkassar ET AL: "Optimized Self-Synchronizing Mode of Operation" In: "Fast Software Encryption", 1 January 2002 (2002-01-01), Springer Berlin Heidelberg, Berlin, Heidelberg, XP55038265, ISBN: 978-3-54-043869-4 vol. 2355, pages 78-91,	1-5,7-11
Y	sections 1 and 3 -----	6