



- (51) **International Patent Classification:**
H04L 9/00 (2006.01) *G06F 12/14* (2006.01)
G06F 21/72 (2013.01)
- (21) **International Application Number:**
PCT/AU2014/000884
- (22) **International Filing Date:**
9 September 2014 (09.09.2014)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
2013903447 9 September 2013 (09.09.2013) AU
2013904459 19 November 2013 (19.11.2013) AU
- (71) **Applicant:** **NEWSOUTH INNOVATIONS PTY LIMITED** [AU/AU]; Rupert Myers Building, Gate 14, Barker Street, The University Of New South Wales, New South Wales 2052 (AU).
- (72) **Inventors:** **IGNJATOVIC, Aleksander**; 22/106 Joynton Avenue, Zetland, New South Wales 2017 (AU). **JAY-ASINGHE, Jayasinghe Arachchige Darshana Nayanajith**; 10/4 Botany Street, Randwick, New South Wales 2031 (AU). **PARAMESWARAN, Sri**; 23 The Serpentine, Kensington, New South Wales 2033 (AU). **AMBROSE, Jude Angelo**; 72 Bungaribee Road, Blacktown, New South Wales 2148 (AU).
- (74) **Agent:** **GRIFFITH HACK**; GPO Box 4164, Sydney, New South Wales 2001 (AU).
- (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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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))

(54) **Title:** DATA ENCRYPTION PROCESS

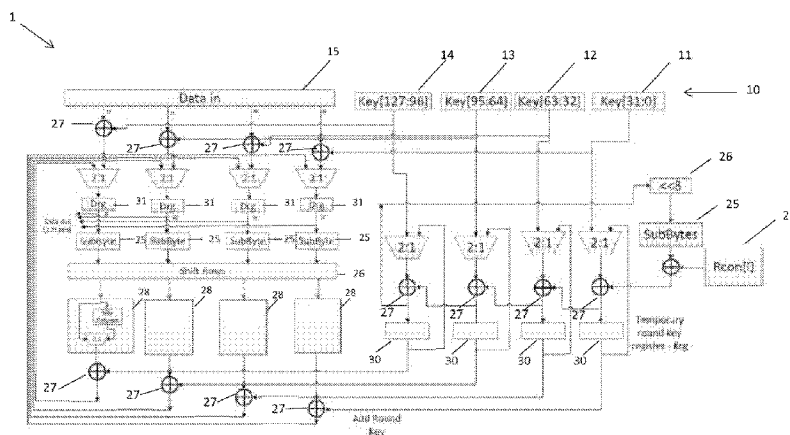


Figure 1

(57) **Abstract:** An encryption system and process are disclosed in this specification. The disclosed process reduces the feasibility of power analysis attacks by regulating the power consumed by an encryption system during an encoding process. The system synchronously encrypts complementary data using reciprocal encryption keys to balance bit transitions and decorrelate power characteristics from the data being processed (the encryption key and data package). A specific application using the AES cipher is presented.

DATA ENCRYPTION PROCESS

Field of the Invention

The present invention relates to a data encryption
5 process and system.

Background

Cryptographic algorithms transform data (often referred
to as 'plaintext') into 'ciphertext'. The object of these
10 algorithms (commonly referred to as 'ciphers') is to prevent
extraction of the plain text from the cipher text without an
independent encryption key. The encryption key is a unique
string that is used by the cipher during the encryption
process to transform the plain text to the cipher text. The
15 same encryption key may be used to decrypt the ciphertext
(symmetric key encryption), or a separate decryption key
(mathematically linked to the encryption key) may be required
(public key encryption).

The cryptographic strength of a cipher is typically
20 evaluated from the theoretical effort required to extract the
plain text from the cipher text without the encryption key.
This evaluation is usually limited to cryptanalysis
(theoretical assessment of the cryptographic algorithm).

Common cryptanalysis procedures include related key
25 attacks, differential cryptanalysis and linear cryptanalysis.
A cipher is considered cryptographically broken if there is a
cryptographic attack that is computationally faster than a
brute-force attack. Cryptographically broken ciphers may still
provide adequate security if the computational effort involved
30 in extracting the plain text makes the theoretical attack
infeasible. The Advanced Encryption Standard (AES) is an
example of a cryptographically broken cipher that is widely
adopted for secure data encryption.

The security of ciphertext may also be compromised by
35 side-channel attacks that target the implementation of a
cryptographic system. Side-channel attacks extract and analyse
data inadvertently leaked from the physical system
implementing a cipher. Common side channel attacks include

cache-timing attacks (analysing the memory usage of the encryption system) or power analysis attacks. These attacks generally require physical access to the system implementing the cipher (such as a smart card).

5

Summary of the Invention

In a first aspect, the invention provides an encryption process comprising:

- receiving a data package for encryption and a
- 10 corresponding encryption key,
- deriving complements of the received data package and the encryption key using bitwise operations,
- establishing four unique encoding pairs from the received data package, the encryption key and the respective
- 15 complements, each encoding pair having a data component and an encryption key component, and
- simultaneously encrypting the data component of each encoding pair within a single encryption system using the corresponding encryption key component.

- 20 In an embodiment, defined registers within the encryption system are cleared before writing intermediate outputs to the defined registers during encryption of the respective data components.

- In an embodiment, the respective data components of each
- 25 encoding pair are encrypted using identical encryption algorithms that are independently executed by a single processing unit within the encryption system.

In an embodiment, the execution of each encryption algorithm is synchronized to a single system clock.

- 30 In an embodiment, replacement look up tables are generated from a standardized look up table defined for the encryption algorithms, the replacement lookup table for the individual encryption algorithms reflecting the composition of the corresponding encoding pairs.

- 35 In an embodiment, a masking algorithm is executed within the single encryption system concurrently with encryption of the respective data components.

In an embodiment, an unbalanced masking cipher that uses an independent masking key to encrypt data is executed within the single encryption system concurrently with encryption of the respective data components, and the composition of the masking key is dynamically altered.

In a second aspect, the invention provides an encryption system comprising:

a parsing engine that is configured to generate four unique encoding pairs from an individual data package and corresponding encryption key, each of the unique encoding pairs comprising a data component and an encryption key component selected from:

- i. the data package,
- ii. the encryption key for the data package,
- iii. a bitwise complement of the data package, and
- iv. a bitwise complement of the encryption key, and

a processing unit that is configured to simultaneously encrypt the data component of each encoding pair using a corresponding encryption key component.

In an embodiment, the system comprises an inversion engine that is configured to generate complements of received data packages and corresponding encryption keys for incorporation in the encoding pairs generated by the parsing engine.

In an embodiment, the inversion engine is configured to generate complementary look up tables for each encryption algorithm that reflect the composition of the corresponding encoding pairs.

In an embodiment, the system comprises a memory management module that is configured to clear defined registers within the processing unit before writing intermediate outputs to the registers during encryption of the respective data components.

In an embodiment, the system comprises a system clock that is used by the processing unit to synchronize execution of each encryption algorithm within the processing unit.

In a third aspect, the invention provides an encryption process comprising:

receiving a data package for encryption and a corresponding encryption key for use encoding the data package,

generating four unique encoding pairs each having a data
5 component and an encryption key component derived from the received data package and encryption key, and

simultaneously encrypting the respective data components of each encoding pair using the corresponding encryption key component within a single encryption system.

10 In an embodiment, complements of the received data package and the encryption key are derived using bitwise operations, and

the four unique encoding pairs are established from the received data package, the encryption key and the respective
15 complements.

In an embodiment, defined registers within the encryption system are cleared before writing intermediate outputs to the defined registers during encryption of the respective data components.

20 In an embodiment, the respective data components of each encoding pair is encrypted using identical encryption algorithms that are independently implemented by a single processing unit within the encryption system.

In an embodiment, the execution of each encryption
25 algorithm is synchronized to a single system clock.

In an embodiment, complementary look up tables are generated for each encryption algorithm that reflect the composition of the corresponding encoding pairs.

In a fourth aspect, the invention provides an encryption
30 process comprising synchronously executing a plurality of identical encryption algorithms within a single encryption system to decorrelate the power consumption of the encryption system from a primary encryption process encoding a received data package and corresponding encryption key.

35 In an embodiment, four independent encryption algorithms are simultaneously executed within the encryption system to regulate static and dynamic power consumption of the encryption system.

In an embodiment, four unique encoding pairs are generated from a received data package and a corresponding encryption key, each encoding pair comprising a data component and an encryption key component.

5 In an embodiment, complements of the received data package and encryption key are derived using bitwise operations, and

the four unique encoding pairs are established from the received data package, the encryption key and the respective
10 complements.

In an embodiment, defined registers within the encryption system are cleared before writing intermediate outputs to the defined registers during encryption of the respective data components.

15 In an embodiment, each encryption algorithm is executed synchronously using a single system clock.

Brief description of the drawings

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

Figure 1 is a block representation of the Advanced Encryption Standard (AES) cipher illustrating the logical relationship between the encryption key and plaintext.

25 **Figure 2** is a schematic representation of an encryption system that regulates power consumption during an encryption process by executing a plurality of individual ciphers with bitwise balanced inputs.

Figure 3 is a block representation of an encryption
30 algorithm involving the synchronous execution of multiple block ciphers.

Detailed Description

An embodiment of an encryption process and system which
35 reduces the scope for power analysis attacks is described in this specification. A particular implementation of power regulating algorithm 50 in accordance with an embodiment of the present invention is depicted in Figure 3. The illustrated

algorithm 50 uses the Advance Encryption Standard (AES). However, the invention is equally applicable to other ciphers (such as the triple DES, Serpent, Twofish, RC6 and MARS algorithms).

5 The algorithm depicted of Figure 3 comprises a plurality of identical encryption ciphers that are independently executed using a single encryption system. The same methodology can be applied to regulate power consumption with a single cipher (described later in this specification). Both
10 single and multiple cipher embodiments regulate the power consumed by the encryption system during the encoding process. The encoding processes disclosed in this specification are typically executed within an encryption system using a single source of power. This reduces the feasibility of power
15 analysis attacks targeting individual components within the encryption system. A single encryption system may comprise multiple processing units that share a common power supply and are not readily isolated for independent power analysis (such as enclosed systems with tamper resistance and/or a tamper
20 shutdown mechanism).

 The ciphers illustrated in Figure 3 are ideally implemented by a single processing unit (such as an ASIC, FPGA, multicore processor or software within a single processing core) within the encryption system. The encryption
25 system executes the respective encryption ciphers simultaneously to regulate the static and dynamic power consumption of the processing unit.

 The inputs to the encryption system are typically limited to the data package being encrypted (often referred to as the
30 'plaintext') and a corresponding encryption key. The encryption system generates four unique encoding pairs from the inputs, using bitwise complements of the data package and the encryption key to supplement the original inputs.

 Each encoding pair comprises a data component (either the
35 received data package or the complement of the data package) and an encryption key component (either the encryption key or the complement of the encryption key). The four ciphers depicted in Figure 3 each receive a unique encoding pair. The

encoding pairs for each cipher (as depicted from left to right) are:

1. data/key,
- 5 2. data/complement(key),
3. complement(data)/key,
4. complement(data)/complement(key).

10 The encryption system regulates the dynamic and static power consumption of the processing unit by synchronously encoding the plaintext component of each encoding pair. This decorrelates the power consumption of the processing unit from the encryption system inputs (the encryption key and the plaintext), making conventional side-channel power consumption
15 attacks infeasible.

 The encryption system duplicates the logic operations of the fundamental cipher at each clock cycle (using the algorithm depicted in Figure 3) to synchronously produce complementary bit patterns in the registers of the processing
20 unit. Balancing the bit patterns produced by the cipher removes underlying correlations between the register states and the processing unit power consumption.

 The power consumption attributable to memory in typical processing units is correlated to the state of the processing
25 unit registers and transitions between different register states (static and dynamic power consumption respectively). Conventional power analysis attacks use this correlation to calculate the register values produced by a cipher and derive a corresponding encryption key.

30 Static power consumption is the power used to maintain the state (bit pattern) of a register. The power consumed maintaining a zero register (a register comprising no non-zero bits) is different to the power required to maintain a register with non-zero bits.

35 Dynamic power consumption is the power used to transition between different memory states (bit patterns). The power consumed transitioning from zero to non-zero bit values is different than the power consumption of non-zero bit values

transitioning to zero bit value.

The encryption system is capable of regulating both static and dynamic power consumption within the processing unit by synchronously executing four identical ciphers. The composition of the encoding pairs depicted in Figure 3 offset bit transitions within the processing unit by balancing bit transitions in complementary registers. Each zero to non-zero bit transition is offset by a non-zero to zero bit transition at a corresponding bit position in a complementary register.

This regulates the dynamic power consumption of the processing unit. The encryption system regulates the static power consumption of the processing unit by simultaneously balancing maintained bit states within the registers. Each zero bit state that is maintained during a clock cycle is offset by the maintenance of a non-zero bit state at a corresponding bit position in a complementary register.

This ensures that the power consumption for the entire encryption system (the four block ciphers) remains substantially constant for each clock cycle, effectively decoupling the system's power consumption from the plaintext and the encryption key. The encryption system may also generate complementary look up tables for ciphers with byte substitutions (described later in relation to the AES cipher).

A block representation of an Advanced Encryption Standard (AES) implementation is depicted in Figure 1. The Advanced Encryption Standard is a widely adopted specification for encryption of electronic data. It is based on the Rijndael block cipher.

The Advanced Encryption Standard (AES) algorithm uses a secret key and fixed lookup table to encrypt input data packages (the 'plaintext'). The data package being encrypted is decomposed into 4x4 byte matrices (termed the 'state'). The individual state matrices generated for a data package are encoded in 10, 12 or 14 rounds (depending on the length of the encryption key). Each round can be decomposed in several sub-procedures. The sub-procedures for a 128 bit encryption key include:

1. Application of an 'exclusive or' (XOR) operation 27 using the state matrix (128 bits of the data package) and the 128 encryption bit key.

2. Decomposition of the state (following the XOR operation 27) into 16 individual bytes and substitution 25 of isolated bytes for values from a lookup table 20.

3. Cyclic permutation of each row from the state 26 and subsequent permutation of the columns within the state 28.

4. Decomposition of the encryption key into bytes and substitution 25 of 4 bytes from the encryption key with values from the lookup table 20.

5. Application of an 'exclusive or' (XOR) operation 27 using the state and one byte from the modified key (the selected byte depending on the round of encryption).

The power used by a processor implementing the Advanced Encryption Standard (AES) can be divided into static and dynamic consumption. Static power consumption is the power drawn by the circuit maintaining an existing state between two consecutive clock pulses. Dynamic power consumption is the power drawn by the circuit changing state.

Every gate of the circuit draws power for each clock cycle. The amount of power drawn by a gate depends on the operation of the gate. The power regulating algorithm disclosed in this specification decorrelates the power consumption of the encryption circuit from the primary encryption process using the received data package and corresponding encryption key.

A schematic representation of an encryption system 114 is depicted in Figure 2. The system 114 comprises a single processing unit 102 that simultaneously executes four identical ciphers. The illustrated encryption system 114 comprises a generic computing system with dedicated modules

that execute the power regulating algorithm. The encryption algorithm may also be implemented in an Application Specific Integrated Circuitry (ASIC), Programmable Gate Arrays (such as an FPGA) or another compatible processor.

5 The illustrated encryption system 114 receives data for encryption (data packages) and corresponding encryption keys from a single data source 120 (although the data package and encryption keys may originate from different sources in practice). The data source 120 in Figure 2 is depicted
10 external to the encryption system 114. Typical external data sources include remote computing systems that are connected to the encryption system by a data network (such as the Internet) and portable memory device (such as USB drives). Data and/or encryption keys may also be received from applications running
15 on the host computing system (i.e. the computing system implementing the encryption algorithm). For example, the host computing system may implement an application that facilitates reception of plaintext data packages directly from a user (through keyboard 100 and user interface 112).

20 The encryption system 114 incorporates a parsing engine 110 that processes received data packages and encryption keys prior to encoding. The parsing engine 110 generates four unique encoding pairs from the received inputs. Each encoding pair comprises a data component and an encryption key
25 component.

 The components of each encoding pair are derived from the inputs that the encryption system receives. The encryption system 114 includes an inversion engine 104 that generates the individual components of the encoding pairs. The inversion
30 engine 104 generates bitwise complements of the data packages and encryption keys received by the encryption system 114. The complements are transmitted to the parsing engine 110 with the corresponding data package and encryption key.

 The parsing engine 110 combines the data package,
35 encryption key and complements in four unique encoding pairs. Each encoding pair comprises a data component (either the received data package or the complement of the data package) and an encryption key component (either the encryption key or

the complement of the encryption key). The composition of the four unique encoding pairs are presented in Table 1.

Encoding pair	Data component	Encryption key component
1	data	key
2	data	complement(key)
3	complement(data)	key
4	complement(data)	complement(key)

Table 1: Composition of the four unique encoding pairs

Each encoding pair is transferred to the processing unit 102 and allocated to a corresponding cipher. The respective ciphers simultaneously encrypt the data component of each encoding pair using the corresponding encryption key component.

The inversion engine 104 may also generate complementary look up tables that reflect the composition of the corresponding encoding pairs. Complementary lookup tables are commonly used with block ciphers that utilize static lookup tables (such as the Rijndael cipher).

The encryption system illustrated in Figure 2 incorporates a memory management module 108. The memory management module 108 'clears' defined registers within the processing unit 102 before the registers are used to store intermediate outputs during the encoding process (i.e. before the registers are written to by the processing unit 102). The registers are uniformly 'pre-charged' with either zero or non-zero bit values by the memory management 108 module to clear any residual differences that may be exploited. The memory management module 108 is typically integrated with the processing unit 102 to facilitate this functionality.

The processing unit 102 synchronously executes the individual ciphers to ensure that the static and dynamic power consumption of the encryption system 114 is regulated. The illustrated encryption system incorporates a system clock 106 that cyclically generates a reference signal for the processing unit 102. The processing unit 102 regulates execution of the respective ciphers using the reference signal

from the system clock 106.

A block representation of an encryption algorithm that regulates static and dynamic power consumption is depicted in Figure 3. The illustrated algorithm 50 comprises four

5 independent ciphers 51, 52, 53, 54 that are executed by a single encryption system. The Rijndael cipher is depicted in Figure 3(although the power regulating methodology is equally applicable to other encryption algorithms).

10 The depicted algorithm 50 regulates the static and dynamic power consumed by the encryption system by offsetting logic gate operations within a corresponding processing unit. The operations performed by the encryption algorithm 50 for a 128 bit encryption key include:

- 15 1. decomposing an input data package received by the encryption system into 128 bit data blocks,
2. replicating each data block to produce a corresponding duplicate data block,
- 20 3. generating complements of each data block and duplicate data block,
4. replicating the corresponding encryption key received
25 by the encryption system to produce a duplicate encryption key,
5. generating complements of the encryption key and the duplicate encryption key,
- 30 6. establishing four unique encoding pairs from the data packages, encryption keys and respective complements,
7. generating complementary look up tables for each
35 encryption algorithm that reflect the composition of the corresponding encoding pairs, and

8. simultaneously encrypting the data component of each encoding pair within a single encryption system using the corresponding encryption key component.

5

The encoding pairs each comprise a data component 15 and an encryption key component 10. The respective components of an encoding pair are used as the inputs to one of the block ciphers executed by the processing unit.

10

The Advanced Encryption Standard defines the use of a lookup table (termed the 'Rijndael S-Box') for byte substitutions 25 in each encryption round. The encryption algorithm 50 illustrated in Figure 3 generates four distinct lookup tables 20 that replace the standard 'Rijndael S-Box' in the depicted ciphers 51, 52, 53, 54. The replacement lookup tables 20 are derived from the 'Rijndael S-Box' and depend on the composition of the encoding pair input to the corresponding cipher 51, 52, 53, 54. The illustrated lookup tables 20 are defined by the following relationships with the 'Rijndael S-Box':

20

- Table 1: $T_{00}(x) = T(x)$
- Table 2: $T_{01}(x) = \text{complement}(T(x))$
- Table 3: $T_{10}(\text{complement}(x)) = T(x)$
- Table 4: $T_{11}(\text{complement}(x)) = \text{complement}(T(x))$

25

Where:

x = an individual byte within a lookup table.

30

$T_{ij}(x)$ = the output from one of the lookup tables generated by the encryption algorithm 50 for the defined byte (x) .

35

$T(x)$ = the output from 'Rijndael S-Box' for the defined byte (x) .

The parsing engine 110 matches the replacement lookup tables 51, 52, 53, 54 with reciprocal encoding pairs. The

encoding pairs and corresponding lookup tables are synchronously processed by the ciphers 51, 52, 53, 54 illustrated in Figure 3. The data package, encryption key and lookup table combinations depicted in Figure 3 comprise:

5

Cipher	Data component	Key component	Lookup table
51	data	key	Table 1 (T_{00})
52	data	complement(key)	Table 3 (T_{10})
53	complement(data)	key	Table 2 (T_{01})
54	complement(data)	complement(key)	Table 4 (T_{11})

Table 2: Replacement lookup table/encoding pair combinations

The encoding pairs and replacement lookup tables derived by the algorithm 50 balance the overall bit composition of the processing unit inputs (i.e. the total number of zero bit states and non-zero bit states are equal for both the data components and encryption key components). This ensures that the 'exclusive or' (XOR) operations performed by the respective ciphers produce equal quantities of each possible bit transformation within the processing unit. The bit transformation balance at each clock cycle is summarized in table 3.

Bit transformation	Percentage of total bits
Zero to non-zero transition	25%
Non-zero to zero transition	25%
Non-zero state maintained	25%
Zero state maintained	25%

Table 3: The ratio of bit transformations per clock cycle

20

The encryption algorithm 50 ideally clears the data registers 31 and key registers 30 used by the processing unit to store intermediate values prior to write operations. This ensures the substitution and permutation operations performed by the respective ciphers are balanced (as summarised in Table 3).

25

The illustrated encryption algorithm 50 uniformly pre-charges the registers 30, 31 with zero or non-zero bit values

at initiation (typically during a 'start-up' routine) and before each write operation. For example, a permutation operation involves:

- copying a row/column from the 'state' matrix to a zeroed register,
- zeroing the original register that the row/column was copied from, and
- copying back the shifted row/column.

A similar process is applicable for pre-charging registers with non-zero bit values.

The operations executed by each cipher offset operations from another cipher within the processing unit. This decorrelates the power profile for the processing unit (the power consumed during each clock cycle) from the primary encryption process (encoding the data package with the corresponding encryption key). The encryption algorithm depicted in Figure 3 consumes approximately four times the power of an individual cipher to conceal correlations between the encryption key and the power characteristics of the system. This decorrelation makes side-channel power consumption attacks on the overall processing unit infeasible.

Masking power consumption profile of the processing unit

The power regulating algorithm disclosed in this specification may be augmented with masking processes that actively disrupt power analysis attacks. Conventional masking implementations produce power consumption characteristics that are not correlated with the underlying 'secure' encryption algorithm. The extraneous 'noise' produced by masking algorithms interferes with power consumption relationships used in power analysis attacks. The encryption algorithm can be executed concurrently with a masking process to improve resistance to power analysis attacks. The masking process and encryption algorithm are typically executed within a single processing unit.

Compatible masking algorithms include pseudo-random power consumption processes and auxiliary 'unbalanced' ciphers. These masking techniques are typically executed synchronously

with the regulated encryption algorithm 50 depicted in Figure 3 to mislead power analytic correlations. For example, the encryption process depicted in Figure 3 may be executed simultaneously with an auxiliary 'unbalanced' cipher that intentionally 'leaks' a disruptive power trace (uncorrelated to the encryption key components 10). The disruptive power trace is usually correlated to a slow changing 'masking' key (an independent encryption key used by the masking algorithm) that complicates power analysis attacks.

Masking keys are usually selected at random and may be dynamically altered during the encryption process to avoid isolation of the 'leaked' power trace. This complicates elimination of the intentionally 'leaked' power trace from the overall power consumption characteristics of the encryption system. For example, individual bits within the masking key may be slowly 'flipped' (transitioned from zero-to-one or one-to-zero) while the encryption system is operating.

Ideally, a random or pseudo-random key alteration process is used to change the composition of the masking key during encryption. The 'bit flipping' process can be randomized by allocating each bit a 'flipping probability' that generally defines the 'flipping' rate of the corresponding bit (and the alteration rate of the encryption key). The state of each bit within the masking key is evaluated at distinct intervals during the encryption process using a random or pseudo-random probability function that is initiated with the 'flipping probability' of the corresponding bit. The masking key is typically evaluated using uniform 'bit flipping probabilities' (of magnitude 0.001% for slow changing keys) during each round of encryption.

Exemplary power regulating software using the AES cipher

An exemplary power regulating encryption algorithm using the Advanced Encryption Standard (AES) is described in this section of the specification. The methodology can be applied to other block ciphers (such as DES, IDEA, RC5 and Blowfish). The disclosed algorithm is implemented in software that is optimized for a single 32-bit processing unit (a processor

with 32-bit registers). The methodology may also be extended to other software optimizations (including optimizations for multi-core processors and 64-bit processors) and dedicated hardware systems.

5 Conventional software implementations of the Advanced Encryption Standard (AES) algorithm encode data in discrete blocks. Each data block (termed the 'state') is compiled from 16 bytes of data extracted from a data package. The data blocks are allocated in a 4 byte by 4 byte matrix (the state
10 matrix) and encrypted in independent cycles of the AES cipher.

Each cycle of the AES cipher encodes an individual state matrix. The encoded output is termed 'ciphertext'. Successive state matrices generated from a common data package are subjected to identical encryption rounds using a common
15 encryption key. The output of each encryption cycle is compiled in the 'ciphertext' (the encrypted output produced from a data package).

The encryption cycles for the AES cipher comprise 10, 12 or 14 encoding rounds (depending on the length of the
20 encryption key). Each round of an encoding cycle comprises several defined sub-procedures. The sub-procedures for a 128 bit encryption key include:

1. Application of an XOR operation 27 using the state
25 matrix (128 bits of the data package) and the 128 encryption bit key.

2. Decomposition of the state (following the XOR operation 27) into 16 individual bytes and substitution
30 25 of isolated bytes for values from a lookup table 20.

3. Cyclic permutation of each row from the state 26 and subsequent permutation of the columns within the state
28.

35 4. Decomposition of the encryption key into bytes and substitution 25 of 4 bytes from the encryption key with values from the lookup table 20.

5. Application of an XOR operation 27 using the state and one byte from the modified key (the selected byte depending on the round of encryption).

5

The power regulating cipher disclosed in this section of the specification uses expanded state, encryption key and reference (lookup table) matrices to balance bit transitions. The state matrix for each encoding cycle is compiled from successive entries within a corresponding data package.

10

Each iteration of the state matrix is passed to the AES cipher and encrypted in an independent encoding cycle. The same encryption key is used for each cycle. The parsing engine 110 and the inversion engine 104 compile input registers for the encoding cycles from the data package and corresponding encryption key. Each 32-bit input register is compiled from a single entry (comprising one byte of data) from one of the inputs to the encryption algorithm (either the data package or the encryption key).

15

20

The parsing engine 110 uses data slots within the respective registers to allocate input entries (generated from the data package and encryption key). Each 32-bit input register has four data slots. The respective data slots each have capacity for one byte (8 bits). The parsing engine 110 copies a single 8 bit entry from either the data package or encryption key (the source) to two data slots within a single input register (the destination). Sequential copy operations are typically used to reproduce the source entry in both destination data slots. The remaining two data slots within the register are filled with bitwise complements of the copied entry (the same entry the parsing engine 110 copied from the data package or encryption key to the first two slots within the register).

25

30

35

Each 32-bit input register contains two copies of an original input entry (one byte from either the data package or the encryption key) and two bitwise complements generated from the original input entry. The inversion engine 104 generates bitwise complements for each original entry copied to the

input registers by the parsing engine 110.

The parsing engine 110 and the inversion 104 compile reciprocal input registers for the state and encryption key. The reciprocal input registers pair input entries from the state matrix with corresponding entries from the encryption key. Each set of reciprocal input registers contains four unique encoding pairs. The encoding pairs are linked by relative position within the respective registers (using the data slots for alignment). The parsing engine 110 and the inversion engine 104 use the data slots to align the 'state' and 'encryption key' pairs compiled in reciprocal input registers.

The input pairs generated by the parsing engine 110 and the inversion engine 104 mirror the encoding pairs used by the power regulating algorithm to offset power correlations (presented in Table 1). A pair of reciprocal input registers is presented in Table 4.

slot 1	slot 2	slot 3	slot 4
data	data	complement(data)	complement(data)
key	complement(key)	Key	complement(key)

Table 4: Byte allocation within reciprocal input registers

The reciprocal input registers presented in Table 4 are compiled for a 32-bit processing unit. The composition of the registers facilitates power regulation with a single AES cipher. The AES cipher manipulates the state matrix and encryption key pairs contained within the input registers during the encryption cycle to produce four 'ciphertext' outputs.

The allocation of encoding pairs within reciprocal input registers balances bit transitions for XOR operations between the respective state and encryption key entries. The processing unit performs the XOR operations at register level. Each set of reciprocal input registers produces the bit transformation pattern presented in Table 3 when subjected to XOR operations.

A compatible set of lookup table registers can be used to

regulate the substitution operations 25 performed by the AES cipher. The parsing engine 110 and the inversion engine 104 produce compatible lookup table registers from the 'Rijndael S-Box' (the lookup table defined in the Advanced Encryption Standard). An exemplary set of registers showing the relative composition of entries within compatible state, encryption key and lookup table registers is presented in Table 5.

slot 1	slot 2	slot 3	slot 4
data	data	complement(data)	complement(data)
key	complement(key)	key	complement(key)
Sbox(a_{ij})	Sbox(b_{ij})	complement(Sbox(c_{ij}))	complement(Sbox(d_{ij}))

Table 5: Replacement lookup table/encoding pair combinations

Where:

$$a_{ij} = \text{XOR}(\text{data}(i, j), \text{key}(i, j))$$

$$b_{ij} = \text{XOR}(\text{data}(i, j), \text{complement}(\text{key}(i, j)))$$

$$c_{ij} = \text{XOR}(\text{complement}(\text{data}(i, j)), \text{key}(i, j))$$

$$d_{ij} = \text{XOR}(\text{complement}(\text{data}(i, j)), \text{complement}(\text{key}(i, j)))$$

The composition of the lookup table registers facilitates register level substitutions. The power regulating cipher is capable of identification substitution registers with a single lookup operation and substituting the selected register for a corresponding register of the expanded state matrix. The parsing engine 110 and/or the inversion engine 104 align entries within the lookup table with corresponding entries in the state and encryption registers. The position of entries within the respective registers mirrors the composition used by the ciphers illustrated in Figure 3 (summarized in Table 2).

The reciprocal composition of the input registers and lookup table registers facilitate register level substitutions. For example, the state register presented in Table 5 may be replaced by a corresponding lookup table register (selected by the AES cipher) to complete a 'state' substitution operation 27.

The lookup table registers are compiled using the same

methodology that was used to produce the input registers. The parsing engine 110 copies one entry (comprising one byte of data) from the 'Rijndael S-Box' to two data slots within a single lookup table register. The remaining two data slots within the register are filled with bitwise complements of the copied entry (the same entry from 'Rijndael S-Box').

Each 32-bit lookup table register contains two copies of an original entry from the 'Rijndael S-Box' and two bitwise complements generated from the original 'S-Box' entry. The inversion engine 104 generates the bitwise complements. The parsing engine 110 and the inversion engine 104 use data slots within the respective registers to align the state, encryption key and lookup table entries.

The general process used to compile 32-bit registers for the AES cipher comprises:

- sequentially extracting entries from a single source (the data package, encryption key or 'Rijndael S-Box')
- duplicating each of the extracted entries,
- generating bitwise complements the duplicates, and
- allocating the duplicated entries and bitwise complements within a 32-bit register.

The parsing engine 110 and the inversion engine 104 produce four unique encoding pairs within each set of reciprocal input registers. The encoding pairs are defined by position (mirrored data slots) within the respective registers. For example, the data entry in slot 2 of the state register is paired with the key entry in slot 2 of the encryption key register. S-Box entries are aligned with reciprocal entries in the 'state' register (as presented in Table 5). The registers compiled for the encryption key may be reused in multiple encryption cycles as the key entries remain unchanged for each iteration of the state matrix.

The parsing engine 110 and the inversion engine 104 match each 8 bit entry from a data package with a corresponding 8 bit entry from the encryption key. The matched inputs (8 bits from the data package and 8 bits from the encryption key) are duplicated, complimented and compiled into reciprocal input registers. The parsing engine 110 passes successive entries

from the data package to the AES cipher in order. The order used by the parsing engine 110 defines the composition of the expanded state and encryption key registers used for the row and column permutations in each encoding cycle.

5 The expanded state matrix comprises 64 bytes of data compiled in a 4-by-16 structure. Each row of the expanded state matrix comprises entries from four successive input registers. The power regulating cipher substitutes the 32-bit input registers compiled by the parsing engine 110 and the
10 inversion engine 104 for the individual entries (8 bits) defined in conventional 4-by-4 state matrices. The byte level composition of an exemplary expanded state matrix is presented in Table 6.

$d01$	$d01$	$\overline{d01}$	$\overline{d01}$	$d02$	$d02$	$\overline{d02}$	$\overline{d02}$	$d03$	$d03$	$\overline{d03}$	$\overline{d03}$	$d04$	$d04$	$\overline{d04}$	$\overline{d04}$
$d05$	$d05$	$\overline{d05}$	$\overline{d05}$	$d06$	$d06$	$\overline{d06}$	$\overline{d06}$	$d07$	$d07$	$\overline{d07}$	$\overline{d07}$	$d08$	$d08$	$\overline{d08}$	$\overline{d08}$
$d09$	$d09$	$\overline{d09}$	$\overline{d09}$	$d10$	$d10$	$\overline{d10}$	$\overline{d10}$	$d11$	$d11$	$\overline{d11}$	$\overline{d11}$	$d12$	$d12$	$\overline{d12}$	$\overline{d12}$
$d13$	$d13$	$\overline{d13}$	$\overline{d13}$	$d14$	$d14$	$\overline{d14}$	$\overline{d14}$	$d15$	$d15$	$\overline{d15}$	$\overline{d15}$	$d16$	$d16$	$\overline{d16}$	$\overline{d16}$

15 **Table 6: Composition of an expanded state matrix**

Where:

$d01, d02, \dots, d16$ = 16 individual 8-bit entries from a data package.

20

$\overline{d01}, \overline{d02}, \dots, \overline{d16}$ = bitwise complements of the 8-bit data package entries.

The power regulating cipher uses the expanded state
25 matrix to balance bit transitions for row and column permutations (shift row and mix columns operations). These operations replicate the permutations executed by conventional AES ciphers. The composition of the exemplary state matrix (presented in Table 6) following a 'shift rows' operation is
30 presented in Table 7.

$d01$	$d01$	$\overline{d01}$	$\overline{d01}$	$d02$	$d02$	$\overline{d02}$	$\overline{d02}$	$d03$	$d03$	$\overline{d03}$	$\overline{d03}$	$d04$	$d04$	$\overline{d04}$	$\overline{d04}$
$d06$	$d06$	$\overline{d06}$	$\overline{d06}$	$d07$	$d07$	$\overline{d07}$	$\overline{d07}$	$d08$	$d08$	$\overline{d08}$	$\overline{d08}$	$d05$	$d05$	$\overline{d05}$	$\overline{d05}$
$d11$	$d11$	$\overline{d11}$	$\overline{d11}$	$d12$	$d12$	$\overline{d12}$	$\overline{d12}$	$d09$	$d09$	$\overline{d09}$	$\overline{d09}$	$d10$	$d10$	$\overline{d10}$	$\overline{d10}$

d_{16}	d_{16}	$\overline{d_{16}}$	$\overline{d_{16}}$	d_{13}	d_{13}	$\overline{d_{13}}$	$\overline{d_{13}}$	d_{14}	d_{14}	$\overline{d_{14}}$	$\overline{d_{14}}$	d_{15}	d_{15}	$\overline{d_{15}}$	$\overline{d_{15}}$
----------	----------	---------------------	---------------------	----------	----------	---------------------	---------------------	----------	----------	---------------------	---------------------	----------	----------	---------------------	---------------------

Table 7: The expanded state matrix after row permutation

The regulating AES cipher balances bit transitions (producing the ratio outlined in Table 3) at each clock cycle for row and column permutation operations. Some of the operations may be combined and/or transformed into a sequence of lookup tables to improve execution speed and memory usage (similar to the use of T-tables with conventional AES ciphers). Other memory management techniques can be used to accommodate the expanded state, encryption key and lookup table matrices produced by the parsing engine 110 and the inversion engine 104. The output from each regulated encoding cycle mirrors the composition of the reciprocal input registers and lookup table registers compiled by the parsing engine 110 and inversion engine 104. For example, the encoding output produced from the input and lookup table registers presented in Table 5 contains standardized 'ciphertext' in data slot 1 (ciphertext that is compatible with conventional implementations of the Advanced Encryption Cipher).

The parsing engine 110 and the inversion engine 104 are capable of decomposing the output registers from each encoding cycle to compile standardized 'ciphertext' output for the power regulation algorithm. The decomposition process generally involves:

- locating 'standardized' output bytes within the output registers produced during each encoding cycle,
- isolating the 'standardized' output bytes contained in successive registers, and
- sequentially appending the 'standardized' output bytes to the ciphertext.

Generating reciprocal registers

An exemplary process for generating reciprocal registers is described in this section of the specification. The process is described using the Advanced Encryption Standard (AES). The methodology can be applied to other block ciphers (such as DES, IDEA, RC5 and Blowfish). The disclosed algorithm is

implemented in software that is optimized for a single 32-bit processing unit (a processor with 32-bit registers). The methodology can be extended to other software optimizations (including optimizations for multi-core processors and 64-bit processors).

The process is executed by the parsing engine 110 and the inversion engine 104. The parsing engine copies entries from a data package and corresponding encryption key to designated input registers. The inversion engine 104 generates bitwise complements of select entries within the respective registers to produce sets of reciprocal registers. The same process is used to generate lookup tables registers from the 'Rijndael S-Box'.

The parsing engine 110 generates precursor registers for the state, encryption key and lookup table from a single entry (8 bits) extracted from a corresponding source. The 32-bit precursor registers contain four copies of the extracted entry. The parsing engine copies the selected entry (8 bits from the corresponding source) to each slot within the designated 32-bit register.

The inversion engine 104 generates bitwise complements of two entries within the precursor registers using a conversion register. Each conversion register contains two unique 8 bit entries: a zero entry (00000000) and a complement entry (11111111). The inversion engine 104 executes an XOR operation between the conversion register and a corresponding precursor register to produce a corresponding register (state, encryption key or lookup table) for the AES cipher. The conversion process for a state register is presented in Table 8.

slot 1	slot 2	slot 3	slot 4
data	data	data	data
00000000	11111111	00000000	11111111
data	complement(data)	data	complement(data)

Table 8: Precursor and conversion registers for the 'state'

A similar process is used to generate the reciprocal

encryption key and lookup table registers. The inversion engine 104 uses the same conversion register to produce reciprocal lookup table and state registers. A complementary conversion register is used to produce the encryption key register. The complementary conversion registers produce four unique encoding pairs within the reciprocal input registers. A complementary conversion register and reciprocal key register for the registers contained in Table 8 are presented in the Table 9.

slot 1	slot 2	slot 3	slot 4
key	key	key	key
00000000	11111111	11111111	00000000
key	complement(key)	complement(key)	key

Table 9: Complementary conversion register and reciprocal encryption key register

There are 24 unique input register combinations that produce encoding pairs compatible with the 32-bit power regulating cipher described in this section of the specification. Each set of complementary conversion registers contains four matched entry (8 bit) combinations. The 24 unique conversion register combinations define different permutations (data slot positions) of these matched entry:

- zero entry | zero entry
- complement entry | complement entry
- zero entry | complement entry
- complement entry | zero entry

Each state/encryption key register combination produced from a set of complementary conversion registers defines four unique encoding pairs. The allocation of encoding pairs within the reciprocal input registers is defined by the corresponding conversion registers selected by the inversion engine 104.

The inversion engine 104 is capable of randomly allocating entries within the respective input registers for successive encoding processes. This is facilitated by randomizing the conversion register selection process. The inversion engine 104 uses a pseudo-random number generator to

randomize conversion register selection. Each set of complementary conversion registers selected by the inversion engine 104 is typically used for the entire encoding process for a corresponding data package.

5 The complementary conversion registers are stored in a conversion lookup table with a unique reference. Each row of the conversion lookup table comprises a pair of complementary registers and the corresponding unique reference. The inversion engine 104 matches the pseudo-random number to the
10 unique reference to select conversion registers for an encoding process. A new pseudo-random number is typically used for each encoding process.

 The conversion lookup table can have up to 24 unique complementary conversion register combinations. More
15 (including duplicate conversion register pairs) or less entries may be used in practice. Entries from an exemplary conversion lookup table are presented in Table 10 using hexadecimal representation for the conversion registers.

Reference	State conversion	Key conversion
1	00 00 FF FF	00 FF 00 FF
2	00 FF 00 FF	00 00 FF FF
3	00 FF FF 00	FF 00 FF 00
:	:	:
24	FF FF 00 00	FF 00 00 FF

20 **Table 10: Conversion register lookup table**

 The parsing engine 110 uses the conversion registers to isolate the encrypted data package (the 'ciphertext') at the conclusion of the encoding process. The conversion registers
25 define the location of the standardised ciphertext within the output registers generated by the power regulating cipher. The standardised ciphertext is compatible with conventional AES ciphers.

 The power regulating cipher produces four encoded
30 outputs. Each 32-bit output register contains one entry from each of the encoded outputs. The allocation of respective outputs within the output registers depends on the conversion

registers used to generate inputs for the encoding process. The relationship between state and encryption key registers presented in Tables 8 and 9 and the output produced by the power regulating cipher is presented in Table 11.

5

slot 1	slot 2	slot 3	slot 4
data	complement (data)	data	complement (data)
key	complement (key)	complement (key)	Key
out	out	complement (out)	complement (out)

Table 11: Relationship between input and output registers

The parsing engine 110 generates an output conversion register by performing an XOR operation with the complementary conversion registers applied to the cipher inputs. The output register facilitates isolation of the standardised ciphertext (the output from the power regulating cipher that is compatible with conventional AES ciphers). An output conversion register produced from the 'state' and 'key' conversion registers presented in Tables 8 and 9 is presented in Table 12.

10

15

Register	slot 1	slot 2	slot 3	slot 4
data	00000000	11111111	00000000	11111111
key	00000000	11111111	11111111	00000000
output	00000000	00000000	11111111	11111111

Table 12: Generation of the output conversion register

The parsing engine 110 executes an XOR operation between the output registers generated by the power regulating algorithm and the output conversion register. This produces four copies of standardized ciphertext in each of the converted output registers. The combination is presented in Table 13.

20

25

slot 1	slot 2	slot 3	slot 4
out	out	complement (out)	complement (out)
00000000	00000000	11111111	11111111
out	out	out	out

Table 13: Combining output and output conversion register

The parsing engine 110 extracts converted output entries (8 bits) from successive output registers and compiles the extracted entries in standardised ciphertext registers. The standardised ciphertext registers are compatible with conventional AES ciphers.

Embodiments of the present invention may be implemented using software (including firmware) and/or dedicated hardware (including integrated circuits and programmable logic devices). Software embodiments can be platform independent (leveraging a virtual machines to interface with underlying hardware), compiled for execution by a target operating systems (such as Windows, OSX, Android, iOS) or developed for customised hardware platforms with defined instruction sets (such as 'system on chip' hardware). Hardware systems can incorporate dedicated circuitry (including 'application specific integrated circuits' or ASIC) and/or programmable logic device (such as 'field programmable gate arrays').

Software instructions for performing embodiments of the invention may be stored in a non-transitory computer readable medium (such as a magnetic hard drive or solid state drive), data signals (typically transmitted via a communication network) or read only memory (such as PROM, EPROM and EEPROM).

It will be understood to persons skilled in the art of the invention that many modifications may be made without departing from the spirit and scope of the invention.

It is to be understood that, if any prior art publication is referred to herein, such reference does not constitute an admission that the publication forms a part of the common general knowledge in the art, in Australia or any other country.

In the claims which follow and in the preceding description of the invention, except where the context requires otherwise due to express language or necessary implication, the word "comprise" or variations such as "comprises" or "comprising" is used in an inclusive sense, i.e. to specify the presence of the stated features but not to

preclude the presence or addition of further features in various embodiments of the invention.

Claims

1. An encryption process comprising:
 - receiving a data package for encryption and a
 - 5 corresponding encryption key for use encoding the data package,
 - deriving complements of the data package and the encryption key using bitwise operations,
 - establishing four unique encoding pairs from the data
 - 10 package, the encryption key and the respective complements, each encoding pair having a data component and an encryption key component, and
 - simultaneously encrypting the data component of each encoding pair within a single encryption system using the
 - 15 corresponding encryption key component.
2. The process of claim 1 wherein the four unique encoding pairs comprise:
 - i. the data package and the encryption key,
 - 20 ii. the data package and a bitwise complement of the encryption key,
 - iii. a bitwise complement of the data package and the encryption key, and
 - iv. a bitwise complement of the data package and a
 - 25 bitwise complement of the encryption key
3. The process of claim 1 or claim 2 comprising clearing defined registers within the encryption system before writing intermediate outputs to the defined registers during
- 30 encryption of the respective data components.
4. The process of any one of claims 1 to 3 comprising encrypting the respective data components of each encoding pair using identical encryption algorithms that are
- 35 independently executed by a single processing unit within the encryption system.
5. The process of claim 4 comprising synchronizing execution

of each encryption algorithm to a single system clock.

6. The process of claim 4 or claim 5 comprising generating replacement look up tables from a standardized look up table defined for the encryption algorithms, the replacement lookup table for the individual encryption algorithms reflecting the composition of the corresponding encoding pairs.

7. The process of any one of claims 1 to 6 comprising executing a masking algorithm within the single encryption system concurrently with encryption of the respective data components.

8. The process of any one of claims 1 to 6 comprising executing an unbalanced masking cipher that uses an independent masking key to encrypt data within the single encryption system concurrently with encryption of the respective data components, and dynamically altering the composition of the masking key.

9. An encryption system comprising:

a parsing engine that is configured to generate four unique encoding pairs from an individual data package and corresponding encryption key, each of the unique encoding pairs comprising a data component and an encryption key component selected from:

- i. the data package,
 - ii. the encryption key for the data package,
 - iii. a bitwise complement of the data package, and
 - iv. a bitwise complement of the encryption key, and
- a processing unit that is configured to simultaneously encrypt the data component of each encoding pair using a corresponding encryption key component.

10 The system of claim 9 wherein the four unique encoding pairs comprise:

- i. the data package and the encryption key,
- ii. the data package and a bitwise complement of the

- encryption key,
- iii. a bitwise complement of the data package and the encryption key, and
- iv. a bitwise complement of the data package and a
5 bitwise complement of the encryption key.

11. The system of claim 9 or claim 10 comprising an inversion engine that is configured to generate complements of received data packages and corresponding encryption keys for
10 incorporation in the encoding pairs generated by the parsing engine.

12. The system of claim 11 wherein the inversion engine is configured to generate complementary look up tables for each
15 encryption algorithm that reflect the composition of the corresponding encoding pairs.

13. The system of any one of claims 9 to 12 comprising a memory management module that is configured to clear defined
20 registers within the processing unit before writing intermediate outputs to the registers during encryption of the respective data components.

14. The system of any one of claims 9 to 13 comprising a
25 system clock that is used by the processing unit to synchronize execution of each encryption algorithm within the processing unit.

15. The system of any one of claims 9 to 14 comprising a
30 masking module that is configured to execute a masking algorithm concurrently with the four identical encryption algorithms.

16. The system of any one of claims 9 to 14 comprising a
35 masking module that is configured to execute an unbalanced masking cipher concurrently with the four identical encryption algorithms and dynamically alter the composition of an independent masking key used by the masking module to encrypt

data.

17. An encryption process comprising:

receiving a data package and a corresponding encryption
5 key ,

generating four unique encoding pairs each having a data
component and an encryption key component derived from the
data package and encryption key, and

simultaneously encrypting the respective data components
10 of each encoding pair using the corresponding encryption key
component within a single encryption system.

18. The process of claim 17 comprising deriving bitwise

complements of the received data package and the encryption
15 key, and establishing the four unique encoding pairs from the
data package, the encryption key and the respective
complements.

19. The process of claim 17 or claim 18 wherein the four
20 unique encoding pairs comprise:

- i. the data package and the encryption key,
- ii. the data package and a bitwise complement of the
encryption key,
- iii. a bitwise complement of the data package and the
25 encryption key, and
- iv. a bitwise complement of the data package and a
bitwise complement of the encryption key.

20. The process of any one of claims 17 to 19 comprising

30 clearing defined registers within the encryption system before
writing intermediate outputs to the defined registers during
encryption of the respective data components.

21. The process of any one of claims 17 to 20 comprising

35 encrypting the respective data components of each encoding
pair using identical encryption algorithms that are
independently implemented by a single processing unit within
the encryption system.

22. The process of claim 21 comprising synchronizing execution of each encryption algorithm to a single system clock.

5 23. The process of claim 21 or claim 22 comprising generating complementary look up tables for each encryption algorithm that reflect the composition of the corresponding encoding pairs.

10 24. An encryption process comprising synchronously executing a plurality of identical encryption algorithms within a single encryption system to decorrelate power consumption from a primary encryption process encoding a received data package and corresponding encryption key.

15 25. The process of claim 24 comprising simultaneously executing four independent encryption algorithms within the encryption system to regulate static and dynamic power consumption of the encryption system.

20 26. The process of claim 24 or claim 25 comprising generating four unique encoding pairs from a data package and a corresponding encryption key, each encoding pair comprising a data component and an encryption key component.

25 27. The process of claim 26 comprising deriving bitwise complements of the data package and the encryption key, and establishing the four unique encoding pairs from the data package, the encryption key and the respective complements.

30 28. The process of claim 26 or claim 27 wherein the four unique encoding pairs comprise:

- i. the data package and the encryption key,
- ii. the data package and a bitwise complement of the encryption key,
- 35 iii. a bitwise complement of the data package and the encryption key, and
- iv. a bitwise complement of the data package and a

bitwise complement of the encryption key.

29. The process of any one of claims 24 to 28 comprising clearing defined registers within the encryption system before
5 writing intermediate outputs to the defined registers during encryption of the respective data components.

30. The process of any one of claim 24 to 29 comprising synchronizing execution of each encryption algorithm to a
10 single system clock.

31. The process of any one of claims 17 to 30 comprising executing a masking algorithm within the single encryption system concurrently with the encryption algorithm.
15

32. The process of any one of claims 17 to 31 comprising executing an unbalanced masking cipher that uses an independent masking key to encrypt data within the single encryption system concurrently with the encryption algorithm,
20 and dynamically altering the composition of the masking key.

33. An encryption process comprising synchronously encrypting complementary data using reciprocal encryption keys within a single encryption system to regulate the power consumed by the
25 encryption system.

34. An encryption apparatus, comprising a computer programmed to implement an encryption process in accordance with any one of claims 1 to 8, 17 to 23, 31, 32, 24 to 30 or 33.

30 35. A computer program, comprising instructions for controlling a computer to implement an encryption process in accordance with any one of claims 1 to 8, 17 to 23, 31, 32, 24 to 30 or 33.

35 36. A computer readable medium, providing a computer program in accordance with claim 35.

37. A data signal, comprising a computer program in

accordance with claim 35.

38. An encryption system comprising computing hardware that is configured to receive a data package and a corresponding
5 encryption key, generate four unique encoding pairs each having a data component and an encryption key component derived from the data package and encryption key, and simultaneously encrypt the respective data components of each
10 encoding pair using the corresponding encryption key component.

39. An encryption system comprising computing hardware that is configured to synchronously execute a plurality of identical encryption algorithms to decorrelate power consumption from a
15 primary encryption process encoding a received data package and corresponding encryption key

40. An encryption system comprising computing hardware that is configured to synchronously encrypt complementary data using
20 reciprocal encryption keys to regulate power consumption.

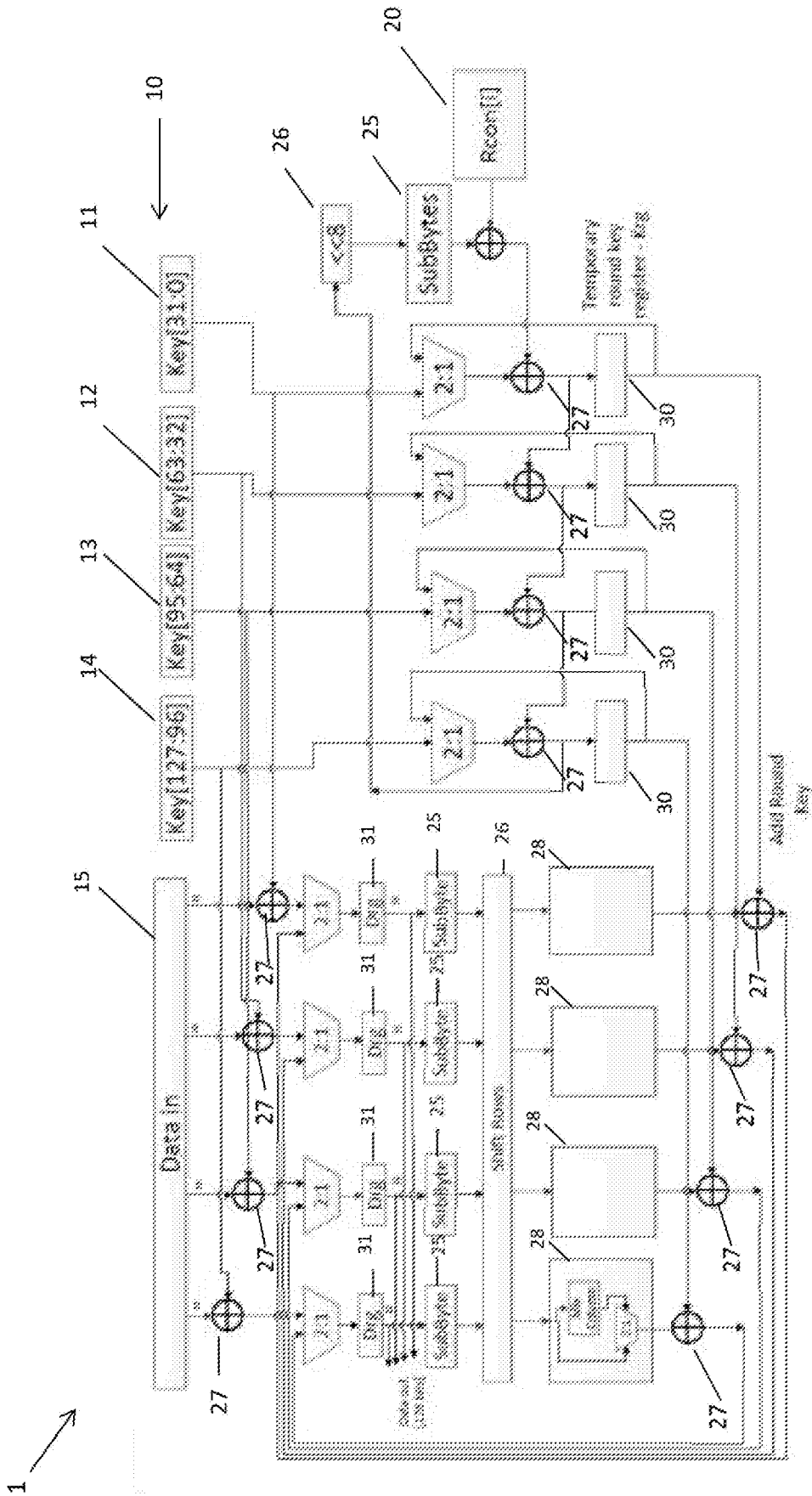


Figure 1

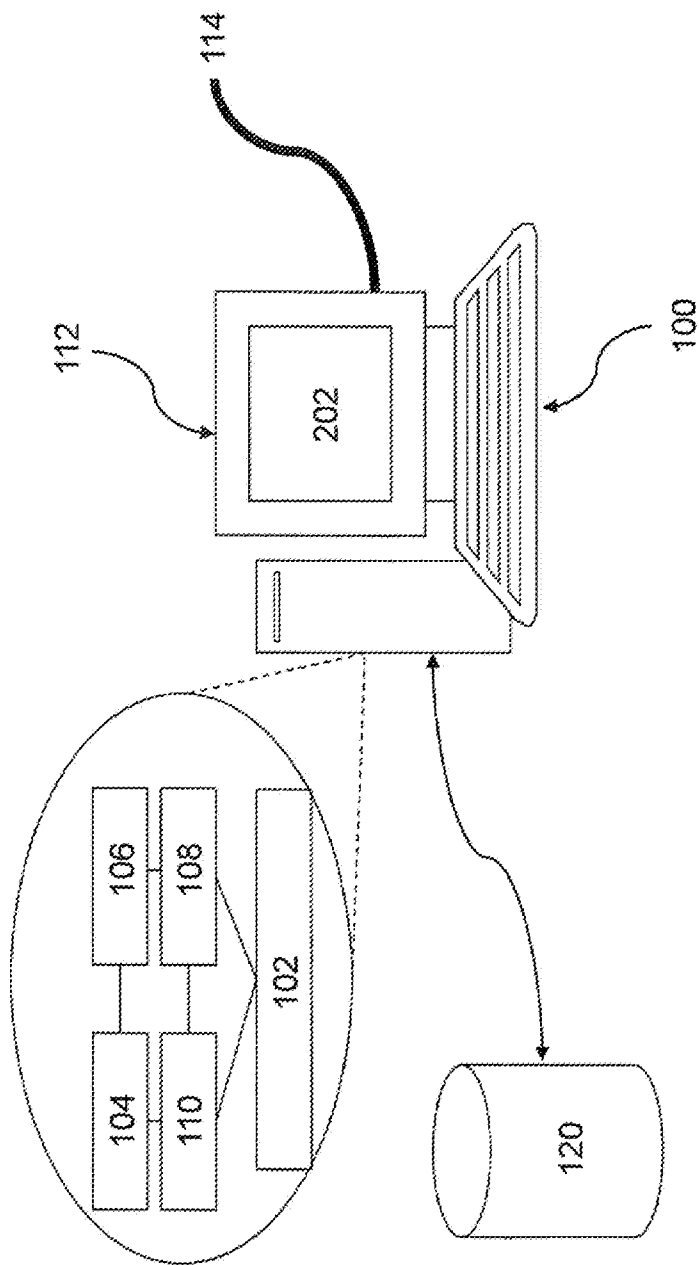


Figure 2

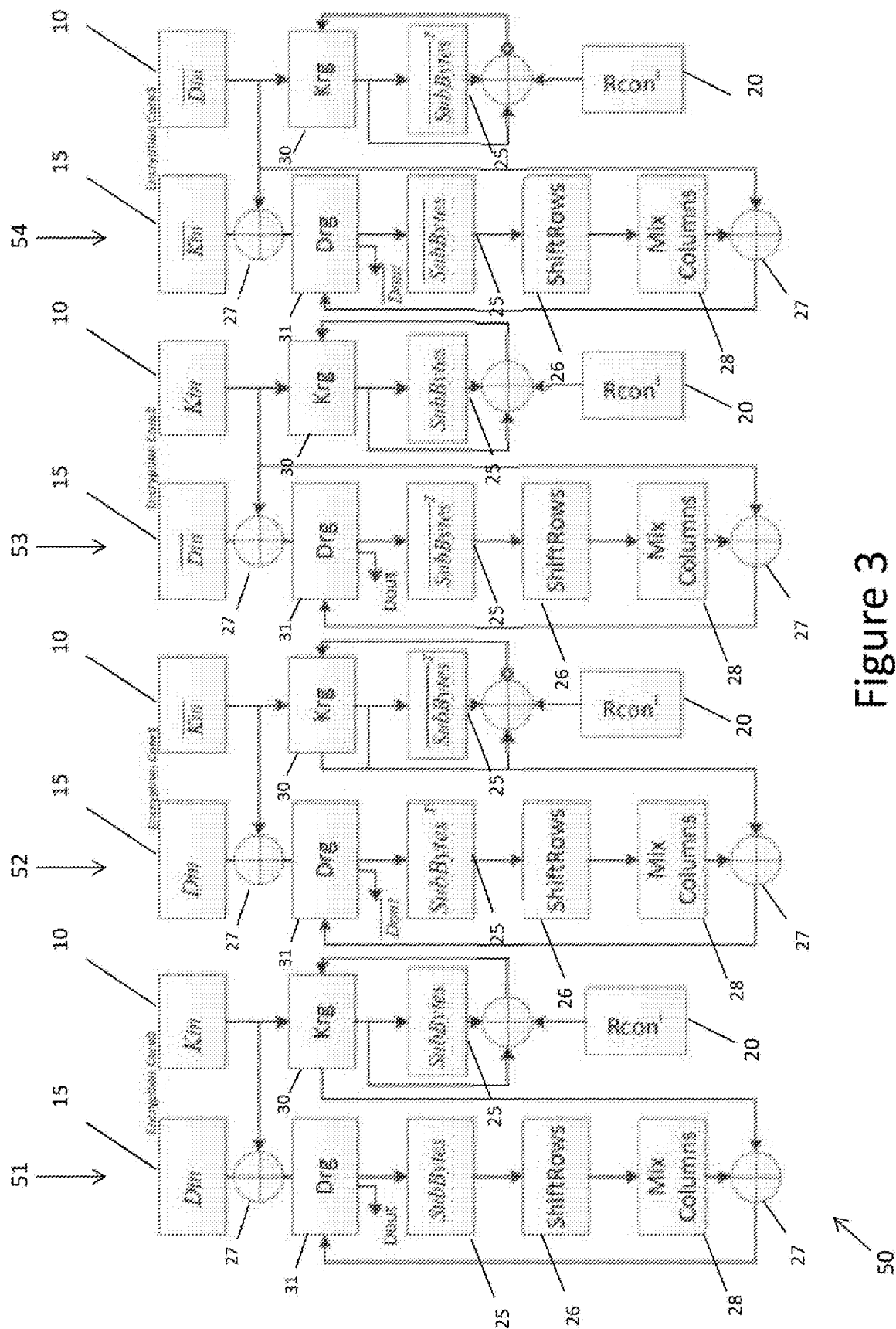


Figure 3

INTERNATIONAL SEARCH REPORT

International application No.

PCT/AU2014/000884

A. CLASSIFICATION OF SUBJECT MATTER

H04L 9/00 (2006.01) G06F 21/72 (2013.01) G06F 12/14 (2006.01)

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)

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)

Invention 1: WPI, EPODOC & keywords: encrypt, decrypt, crypt, encode, decode, cipher, secret, scramble, alter, secure, decompose, plaintext, data package, data input, key, complement, balance, reciprocal, conjugate, negate, regulate, fill, even, correlate, offset, substitute, replace, inverse, "XOR" or transform, bitwise, binary, logic gate, four, quad, "4", multiple, pair, dual, combine and similar terms; **Google Patents, Google Scholar, The Lens, USPTO Patent Collection, USPTO Patent Application & keywords:** cipher, encrypt, decrypt, encode, decode, complement, reciprocal, conjugate, "XOR", "exclusive, plaintext, key, bitwise, binary, logic gate, four, quad, multiple and similar terms;

Invention 2: WPI, EPODOC & keywords: encrypt, crypt, encode, cipher, secret, scramble, alter, secure, plaintext, data package, data input, key, power, side-channel, electricity, voltage, current, energy, analysis, attack, hack, intercept, consumption, decorrelate, regulate, constant, consistent and similar terms; **Google Patents, Google Scholar, The Lens, USPTO Patent Application & keywords:** cipher, encrypt, encode, plaintext, key, synchronous, identical, copy, duplicate, power, side-channel, electricity, voltage, current, energy, analysis, attack, consumption, decorrelate, regulate, constant, consistent and similar terms; **Applicant/Inventor Search by Google Search:** (encryption OR decryption OR cipher) (Ignjatovic OR Jayasinghe OR parameswaran OR ambrose OR "Newsouth Innovations") and similar terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	



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	"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
"E" earlier application or patent but published on or after the international filing date	"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
"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)	"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
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
18 November 2014Date of mailing of the international search report
18 November 2014

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustalia.gov.au

Authorised officer

Benjamin Lam
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. +61 2 6225 6121

INTERNATIONAL SEARCH REPORT		International application No.
C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		PCT/AU2014/000884
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2007/0064933 A1 (BANTWAL) 22 March 2007	
A	WO 2002/041566 A2 (HONEYWELL INTERNATIONAL INC.) 23 May 2002	
A	KOCHER, P. et al., Differential Power Analysis, Advances in Cryptology - CRYPTO' 99, Lecture Notes in Computer Science, Volume 1666, 1999, pp 388-397, Date: 16 Dec 1999	
A	CHENG, H. et al., PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems, 11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN, Architectures, Methods and Tools, DSD '08, pages 383-390, 3-5 September 2008	
A	Fischer, W. et al., Differential Power Analysis of Stream Ciphers, Topics in Cryptology - CT-RSA 2007, Lecture Notes in Computer Science, Volume 4377, 2006, pp 257-270	

Form PCT/ISA/210 (fifth sheet) (July 2009)

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
the subject matter listed in Rule 39 on which, under Article 17(2)(a)(i), an international search is not required to be carried out, including
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a)

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

See Supplemental Box for Details

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☒ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

Supplemental Box**Continuation of: Box III**

This International Application does not comply with the requirements of unity of invention because it does not relate to one invention or to a group of inventions so linked as to form a single general inventive concept.

This Authority has found that there are different inventions based on the following features that separate the claims into distinct groups:

- Claims 1-23, 31-32 (only when claims 31 and 32 depend on any one of claims 17-23), 34-37 (only when claims 34 and 35 depend on any one of claims 1-8, 17-23), 38 are directed to an encryption process and an encryption system. The feature of (in particular) "generating four unique encoding pairs and simultaneously encrypting a data component of each encoding pair using a corresponding encryption key component" is specific to this group of claims.
- Claims 24-30, 31 (only when claim 31 depends on any one of claims 24-30), 32 (only when claim 32 depends on any one of claims 24-31), 33, 34-37 (only when claims 34 and 35 depend on any one of claims 24-33), 39, 40 are directed to an encryption process. The feature of (in particular) "synchronously executing a plurality of identical encryption algorithms to decorrelate power consumption from a primary encryption process encoding a received data package and corresponding encryption key", or "synchronously encrypting complementary data using reciprocal encryption keys to regulate power consumption" is specific to this group of claims.

PCT Rule 13.2, first sentence, states that unity of invention is only fulfilled when there is a technical relationship among the claimed inventions involving one or more of the same or corresponding special technical features. PCT Rule 13.2, second sentence, defines a special technical feature as a feature which makes a contribution over the prior art.

When there is no special technical feature common to all the claimed inventions there is no unity of invention.

In the above groups of claims, the identified features may have the potential to make a contribution over the prior art but are not common to all the claimed inventions and therefore cannot provide the required technical relationship. Therefore there is no special technical feature common to all the claimed inventions and the requirements for unity of invention are consequently not satisfied *a priori*.

INTERNATIONAL SEARCH REPORT Information on patent family members		International application No. PCT/AU2014/000884	
This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.			
Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	Publication Date	Publication Number	Publication Date
US 2007/0064933 A1	22 March 2007	US 7801307 B2	21 Sep 2010
		CN 101461173 A	17 Jun 2009
		CN 101461173 B	20 Jun 2012
		EP 1927212 A2	04 Jun 2008
		JP 2009503580 A	29 Jan 2009
		KR 20080031906 A	11 Apr 2008
		WO 2007120169 A2	25 Oct 2007
WO 2002/041566 A2	23 May 2002	AU 1982202 A	27 May 2002
		US 7277543 B1	02 Oct 2007
End of Annex			
Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001. Form PCT/ISA/210 (Family Annex)(July 2009)			