República Federativa do Brasil
Ministério da Economia
Instituto Nacional da Propriedade Industrial

(11) PI 0916687-4 B1

(22) Data do Depósito: 12/08/2009

(45) Data de Concessão: 29/09/2020

(54) Título: MÉTODO PARA GERAR UM CÓDIGO DE AUTENTICAÇÃO DE MENSAGEM BASEADO EM CIFRA

(51) Int.Cl.: H04L 9/06; H04L 9/32.

(52) CPC: H04L 9/06; H04L 9/0631; H04L 9/0643; H04L 9/32; H04L 2209/12; (...).

(30) Prioridade Unionista: 19/08/2008 EP 08105068.4.

(73) Titular(es): NXP B.V..

(72) Inventor(es): BRUCE MURRAY; MATHIAS WAGNER.

(86) Pedido PCT: PCT IB2009053556 de 12/08/2009

(87) Publicação PCT: WO 2010/020910 de 25/02/2010

(85) Data do Início da Fase Nacional: 27/01/2011

(57) Resumo: MÉTODO PARA GERAR UM CÓDIGO DE AUTENTICAÇÃO DE MENSAGEM BASEADO EM CIFRA. A presente invenção se refere a um método para gerar um código de autenticação de mensagem baseado em cifra, em que uma matriz de estado (25) compreendida por linhas (31,34) de bytes (s' 0 - s' 15) e colunas (41,44) de bytes (s' 0- s'15) baseadas em uma mensagem a ser transmitida é gerada. O código de autenticação de mensagem baseado em cifra é gerado pela retenção dos bytes (29,30) de pelo menos uma linha (32, 34) da matriz de estado (25).

“MÉTODO PARA GERAR UM CÓDIGO DE AUTENTICAÇÃO DE MENSAGEM BASEADO EM CIFRA”.

CAMPO DA INVENÇÃO

[0001] A presente invenção se refere a um método para gerar um código de autenticação de mensagem baseado em cifra.

ANTECEDENTES DA INVENÇÃO

[0002] Códigos de autenticação de mensagem (MACs) são pedaços de informação usados para autenticar mensagens. As entradas para um algoritmo gerar um MAC são uma chave secreta e a mensagem a ser autenticada. MACs baseados em cifra (CMAC) são códigos de autenticação de mensagem baseados em cifra de bloco e são descritos, por exemplo, na publicação especial NIST (The National Institute of Standards and Technology) 800-38B, May 2005.

[0003] O CMAC em uma mensagem é construído pela divisão em blocos de tamanho igual ao tamanho do bloco da cifra subjacente, por exemplo, 128 *bits* no caso do Padrão de Encriptação Avançado (AES, do inglês *Advanced Encryption Standard*), a Criptografia de Blocos Encadeados (CBC) da mensagem (com *padding* no último bloco, se requerido), e retendo (todo ou parte) do resultado da encriptação do último bloco como o valor MAC calculado.

[0004] Para evitar certas classes de ataque, o último bloco é submetido, antes da cifragem a uma disjunção exclusiva (*XORing*) com um ou dois possíveis valores de “subchaves”, geralmente estipulados como K1 ou K2, derivados de uma encriptação do vetor zero sob a chave em uso. A escolha de qual subchave utilizar é determinada pelo fato de o último bloco de mensagem conter *padding* ou não. Os valores da subchave podem somente ser computados por partes que conheçam a chave de cifra em uso.

[0005] Se o MAC for menor que o tamanho do bloco da cifra, então o padrão determina que o MAC computado deve ser truncado pela retenção do número requerido de *bits* mais significativos.

[0006] Se um MAC for computado em uma mensagem de tamanho inferior ou igual ao tamanho do bloco da cifra, então o último bloco é também o primeiro bloco, de modo que a modificação pela chave *XORing* seja realizada neste único bloco. Isso significa que a entrada direta à operação do bloco da cifra durante esta computação do MAC não é conhecida por um observador externo.

[0007] A figura 1 mostra uma matriz de estado 1 e sua numeração de *bits* de acordo com o AES divulgado na Federal Information Processing Standard (FIPS) publication 197, November 26, 2001.

[0008] A cifra AES opera na matriz de estado 1 de *bytes*, que é do tamanho de 4 *bytes* por 4 *bytes* e tem entradas de *byte* $S_{r,c}$, em que o índice “r” indica a linha relevante e o índice “C” indica a coluna relevante da matriz de estado 1. Ao representar as saídas da operação de cifra AES como uma série de *bits*, os *bytes* são ordenados como se segue:

$S_{0,0}\ S_{1,0}\ S_{2,0}\ S_{3,0}\ S_{0,1}\ S_{1,1}\ S_{2,1}\ S_{3,1}\ S_{0,2}\ S_{1,2}\ S_{2,2}\ S_{3,2}\ S_{0,3}\ S_{1,3}\ S_{2,3}\ S_{3,3}$

[0009] Subsequentemente, os *bytes* dessa série de *bits* são numerados de acordo com uma convenção, na qual s_{15} é o *byte* mais a esquerda ou mais significativo e s_0 e o *byte* mais a direita ou menos significativo, de modo que

$$s_{r,c} = s_{15 - (4c + r)}$$

[0010] O método padrão para truncamento de um MAC de acordo com o padrão NIST mencionado anteriormente é reter o número requerido de *bits* mais significativos. Consequentemente, o truncamento de um MAC baseado em AES em 8 *bits* corresponde à retenção dos *bytes* de estado final s_{15} a s_8 inclusive.

[0011] A figura 2 mostra como um exemplo um estado AES de 16 *bytes* durante o *round* final de uma computação MAC, de acordo com o padrão. No início de uma assim-chamada cifra, uma chave do *round* (*Round Key*) inicial 21 é adicionada à matriz de estado 1 da Fig. 1 a fim de gerar uma matriz de estado 22 (operação *AddRoundKey*). A matriz de estado 22 é submetida a uma transformação *ShiftBytes* para gerar uma primeira matriz de estado

transformada 23, que é submetida a uma transformação *ShiftRow* para gerar uma segunda matriz de estado transformada 24. Em seguida, uma nova chave do *round* 26 é adicionada à segunda matriz de estado transformada 24 por disjunção exclusiva (*XORing*) de cada coluna do Estado da segunda matriz de estado transformada 24 com uma palavra de um cronograma de chave para gerar uma matriz de estado de saída 25 composta por linhas 31-34 e colunas 41-44. A matriz de estado 25 é utilizada para a computação do CMAC, em conformidade com o padrão, retendo os 8 *bytes* mais significativos S_{15} a S_8 após o truncamento e os *bytes* restantes são descartados. Os 8 *bytes* mais significativos 27 para S_{15} a S_8 são ilustrados em sombreado.

[0012] As matrizes 22-24 mostram os *bytes* correspondentes antes do efeito das operações *ShiftRows* e *SubBytes* do *round* final. Assim, com base nas observações dos *bytes* de saída sombreados e hipóteses sobre as posições correspondentes na matriz de chave do *round* final 26, um ataque de análise diferencial de consumo de energia (DPA) é capaz de recuperar alguns *bytes* das chaves do *round* 21, 26.

[0013] Nesse estágio, o atacante pode reunir informação adicional sobre a penúltima chave do *round*, uma vez que ele/ela pode computar a expansão da chave AES em ordem reversa.

[0014] O algoritmo de expansão da chave AES pode ser escrito na forma

$w_{n-4} = T(w_{n-1}) <+> w_n$

[0015] em que w_n é uma palavra de 32 *bits* correspondente a uma coluna da matriz da chave do *round* 21, 26, “<+>” indica a operação de “OU exclusivo”, e T() é uma transformação condicional tal que

$T(w_n) = S(w_n <<< 8) <+> Rconst$; se $n = 0 \bmod 4$, ou $T(w_n) = w_n$ de outra forma

<<< 8 indica rotação para esquerda de 8 posições de *bits*, S() indica aplicação *bit* a *bit* (*bitwise*) de *SubBytes*, e Rconst é uma constante do *round* que varia por *round,* mas é conhecida.

[0016] Com esta combinação de *bytes*, a execução do algoritmo de

expansão de chave adiante rende novamente mais *bytes* da chave do *round* final. Neste ponto, o atacante não pode ir adiante sem dificuldade considerável. A operação anterior à inserção da penúltima chave do *round* é uma *MixColumns*, e com apenas dois *bytes*/coluna disponíveis na entrada para *SubBytes*, as equações necessárias à construção de funções de seleção DPA baseadas em *bytes* anteriores são indeterminadas. No entanto, ele/ela já reduziu a complexidade do ataque a apenas 2^{40} (uma vez que apenas 5 *bytes* da última chave do *round* permanecem desconhecidos) e, a este nível os *bytes* remanescentes podem ser facilmente recuperados através de um ataque de força bruta.

OBJETO E RESUMO DA INVENÇÃO

[0017] É um objeto da presente invenção fornecer um método para gerar um código de autenticação de mensagens baseado em cifra, o que complica um ataque DPA.

[0018] O objeto é alcançado de acordo com a invenção por meio de um método para gerar um código de autenticação de mensagens baseado em cifra, compreendendo as etapas de:

[0019] gerar uma matriz de estado composta de linhas de *bytes* e colunas de *bytes* com base em uma mensagem a ser transmitida e

[0020] computar um código de autenticação de mensagens baseado em cifra para a mensagem por uma retenção dos *bytes* de pelo menos uma linha da matriz de estado.

[0021] Como discutido na introdução, códigos de autenticação de mensagens baseados em cifra (CMACs) são pedaços de informação usados para autenticar mensagens. Uma entrada para gerar o CMAC é a mensagem a ser autenticada. Durante a geração do CMAC, uma matriz de estado com base na mensagem a ser transmitida é gerada.

[0022] A geração de CMAC convencional é também baseada nesta matriz de estado, mas retém os 8 *bits* mais significativos desta matriz de estado para o CMAC. De acordo com o método da invenção, entretanto, o CMAC é

computado utilizando *bytes* de pelo menos uma linha da matriz de estado. Os *bytes* das linhas remanescentes são descartados. Isso pode ter o resultado de que um ataque diferencial de consumo de energia seja mais difícil de ser realizado no método da invenção.

[0023] De acordo com uma modalidade do método da invenção, os *bytes* de duas linhas são utilizados e os *bytes* das linhas restantes são descartados. Particularmente, uma linha cujos *bytes* são descartados está entre as duas linhas cujos *bytes* são retidos para o cálculo do CMAC da presente invenção. Se a matriz de estado é composta por quatro linhas e quatro colunas, então as duas linhas cujos *bytes* são retidos para o cálculo do CMAC da presente invenção podem ser as duas linhas pares ou as duas linhas ímpares da matriz de estado.

[0024] Alternativamente à computação do CMAC pela retenção dos *bytes* de pelo menos uma linha da matriz de estado, o código de autenticação de mensagem baseado em cifra pode ser calculado, submetendo pelo menos dois *bytes* da mensagem ou dois *bytes* da mensagem que foram pré-processados para uma operação de "OU exclusivo". Os *bytes* da mensagem podem ser pré-processados em ordem, por exemplo, para gerar a matriz de estado composta de linhas de *bytes* e colunas de *bytes* com base na mensagem. Então, pelo menos dois *bytes* da matriz de estado podem ser submetidos à operação de "OU exclusivo". A operação de "OU exclusivo" é normalmente referida como operação "XOR".

[0025] O código de autenticação de mensagens baseado em cifra pode ser gerado submetendo *bytes* correspondentes a uma metade da mensagem com *bytes* correspondentes à outra metade da mensagem para a operação de "OU exclusivo". Se a mensagem foi pré-processada para gerar a matriz de estado, então o código de autenticação de mensagens baseado em cifra pode ser calculado submetendo os *bytes* correspondentes a uma metade da matriz de estado com *bytes* correspondentes à outra metade da matriz de estado para a operação de "OU exclusivo". Então, todos os *bytes* da mensagem ou todos

os *bytes* da matriz de estado são usados, aliviando a possibilidade de um ataque DPA bem sucedido. Para atacar essa abordagem, a análise diferencial de consumo de energia (DPA) (ataque) seria necessária para construir e testar hipóteses sobre dois *bytes* de chave do *round* final de cada vez. A análise da correlação do atacante precisaria procurar pares de sinais de *bytes* internos que sustentou hipóteses sobre cada par de *bytes* da chave do *round*. Assim, a análise inicial DPA se torna mais difícil em relação à utilização da computação convencional CMAC acordo com o padrão.

[0026] A matriz de estado pode particularmente ser gerada de acordo com o Padrão de Encriptação Avançado (AES), conforme publicado, por exemplo, no Federal Information Processing Standard (FIPS) publication 197, November 26, 2001 e, resumidamente descrito na introdução. Dessa forma, a matriz de estado, que é usada para computar o código de autenticação de mensagens baseado em cifra (CMAC) de acordo com o método da presente invenção compreende exatamente quatro linhas de *bytes* e exatamente quatro colunas de *bytes*.

[0027] A matriz de estado não precisa ser calculada de acordo com o padrão AES e, portanto, não necessariamente compreende quatro linhas e quatro colunas. Em geral, qualquer número de linhas ou colunas pode ser usado. Um conceito mais geral é conhecido como Rijndael. Particularmente, a matriz de estado pode compreender exatamente quatro linhas de *bytes* e seis colunas de *bytes*, ou quatro linhas de *bytes* e oito colunas de *bytes*.

[0028] A mensagem para a qual o CMAC é computado pode ser uma mensagem de bloco único. Então, a mensagem pode ser representada por uma única matriz de estado quando baseada, por exemplo, no padrão AES. Em outras palavras, o tamanho da mensagem é igual ou inferior ao tamanho do bloco da cifra. Esta variante pode ter resistência melhorada contra um ataque de análise diferencial de força.

BREVE DESCRIÇÃO DOS DESENHOS

[0029] A invenção será descrita em maiores detalhes a seguir, a título de exemplos não limitativos, com referência às modalidades mostradas nos desenhos.

[0030] Fig. 1 é uma matriz de estado que representa uma mensagem de bloco único;

[0031] Fig. 2 ilustra o *round* final de uma computação MAC acordo com o padrão AES;

[0032] Figs. 3 a 6 ilustram o *round* final de uma computação CMAC da presente invenção incluindo um ataque DPA em potencial, e

[0033] Fig. 7 ilustra o *round* final de outra computação CMAC da presente invenção.

DESCRIÇÃO DETALHADA DAS MODALIDADES

[0034] Figs. 1 e 2 foram discutidas na introdução.

[0035] Figs. 3 a 5 ilustram um potencial ataque diferencial de consumo de energia em um CMAC que foi computado em conformidade com o método da presente invenção.

[0036] Para a modalidade exemplificativa, a matriz de estado 25 utilizada para computar o CMAC é obtida como discutido na introdução. A matriz de estado 25 é composta por quatro linhas 31-34 e quatro colunas 41-44. Além disso, a mensagem na qual a matriz de estado 25 é baseada é uma mensagem de bloco único, como representada pela matriz de estado 1 mostrada na Fig. 1.

[0037] Para a modalidade exemplificativa, o CMAC é computado através da retenção de *bytes* 29, 30 de duas das quatro linhas 31-34 da matriz de estado 25. Os *bytes* das linhas restantes são descartados. Para o exemplo mostrado, os *bytes* 29, 30 da linha 32 e linha 34, que são sombreados nas figuras 3 a 5, são retidos e os *bytes* das linhas restantes 31, 33 são descartados. Por isso, os *bytes* pares da matriz de estado 25 são retidos para a computação do CMAC para a modalidade exemplificativa.

[0038] Uma potencial análise diferencial de consumo de energia gera o seguinte:

[0039] Por mecanismos análogos à descrição anterior na introdução, as operações de *ShiftRows* e *SubBytes* podem ser vistas para expor *bytes* apenas nas linhas selecionadas, cujos *bytes* correspondentes estão sombreados, para um atacante DPA, enquanto os *bytes* da chave do *round* final que ele/ela pode recuperar se limitam a essas mesmas duas linhas. Com a mesma estratégia de ataque, ele/ela pode agora executar a expansão da chave inversamente, para obter um número de *bytes* da chave do penúltimo *round*, como mostrado na fig. 4 (ainda restrito às linhas selecionadas).

[0040] A aplicação deste como um processo puramente computacional, com o exemplo de separação mostrado, apenas permitirá que o atacante recupere os 6 *bytes* mostrados sombreados na chave do penúltimo *round* 21. Isto se dá devido ao efeito da transformação condicional, que inclui uma rotação de *bytes* quando n = 0 mod 4.

[0041] Para esclarecer isso, os *bytes* da relação de interação que afetam a coluna mais à esquerda da chave do penúltimo *round* 21 são marcados como A, B, C e D (para *bytes* conhecidos), com "? " indicando um *byte* desconhecido, como mostrado na Fig. 6. Na computação da coluna n-4, a transformação condicional T () entra em jogo e pode ser visto na Fig. 6, que nenhum dos *bytes* são computacionalmente disponíveis.

[0042] Na tabela da Fig. 6, que mostra a relação de iteração

$w_{n-4} = T(w_{n-1}) <+> w_n$

deve ser apreciado que os *bytes* conhecidos em T (w_{n-1}) se alinham com *bytes* desconhecidos em w_n e vice-versa, devido à rotação realizada na transformação condicional T (), então os *bytes* da chave do penúltimo *round* 21 não estão mais disponíveis por computação. Os valores hexadecimais mostrados na linha da tabela da Fig. 6 dando T(w_{n-1}) são os *bytes* do valor *Rconst* adequado.

[0043] Como ilustrado pela Fig. 5, o atacante pode ainda recuperar dois *bytes* adicionais 27 e 28 da chave do penultimo *round* 21 ao estender suas

análises DPA, desde que ele/ela conheça os *bytes* nas posições de estado correspondentes a cada traço que ele/ela reuniu.

[0044] No entanto, essa processamento adicional não melhora a posição do atacante, já que não mais de 8 *bytes* de qualquer chave do *round* 21, 26 foram recuperados. A complexidade remanescente do ataque é 2^{64}, que é uma melhora significativa na resistência remanescente resultante da utilização da abordagem MAC de truncamento padronizada. Embora *bytes* de chaves de *rounds* anteriores (por exemplo, antepenúltimo) possam ser computados, isso não rende ao atacante qualquer benefício adicional.

[0045] A figura 7 ilustra uma modalidade adicional de uma computação CMAC. Para a modalidade exemplificativa, a matriz de estado 25 utilizada para computar o CMAC é obtida como discutido acima. A matriz de estado 25 é composta de quatro linhas 31-34 e quatro colunas 41-44. Além disso, a mensagem na qual a matriz de estado 25 é baseada é uma mensagem de bloco único, como representada pela matriz de estado 1 mostrada na Fig. 1. A matriz de estado 25 é composta por 16 *bytes* s'_0-s'_{15}.

[0046] Para a modalidade exemplificativa, o CMAC é computado submetendo pelo menos dois *bytes* s'_0-s'_{15} da matriz de estado 25 a uma operação de "OU exclusivo". Particularmente, para a modalidade descrita na figura 7, o CMAC é computado por disjunção exclusiva (*XORing*) dos *bytes* s'_0, S'_2, s'_4, s'_6, s'_8, s'_{10}, s'_{12}, s'_{14} das linhas 34, 32, com os *bytes* s'_1, s'_3, s'_5, s'_7, s'_9, s'_{11}, s'_{13}, s'_{15} das linhas 33, 31.

[0047] Especialmente para a modalidade exemplificativa, o CMAC é computado da seguinte forma:

CMAC= {s'_0 <+> s'_1; s'_2 <+> s'_3; s'_4 <+> s'_5; s'_6 <+> s'_7; s'_8 <+> s'_9; s'_{10} <+> s'_{11}; s'_{12} <+> s'_{13}; s'_{14} <+> s'_{15}}

em que “<+>” indica a operação de “OU exclusivo”.

[0048] Finalmente, deve ser notado que as modalidades acima ilustram ao invés de limitar a invenção, e que aqueles versados na técnica serão capazes de conceber muitas modalidades alternativas, sem se afastar do

escopo da invenção, tal como definido pelas reivindicações anexas. Nas reivindicações, quaiquer sinais de referência colocados entre parênteses não devem ser interpretados como uma restrição à reivindicação. A palavra "compreendendo" e "compreende", e semelhantes, não exclui a presença de elementos ou etapas que não sejam as listadas em qualquer reivindicação ou no relatório como um todo. A referência no singular de um elemento não exclui a referência no plural de tais elementos, e vice-versa. O simples fato de que certas medidas são recitadas em reivindicações dependentes mutuamente diferentes não indica que uma combinação destas medidas não possa ser aproveitada.

REIVINDICAÇÕES

1. Método para gerar um código de autenticação de mensagem baseado em cifra para uma mensagem a ser transmitida, a mensagem sendo compreendida de uma pluralidade e *bytes*, o método caracterizado pelo fato de compreender as etapas de:

- gerar uma matriz de estado (25) composta de linhas (31-34) de *bytes* (s'_0-s'_{15}) e colunas (41-44) de *bytes* (s'_0-s'_{15}) baseadas em uma mensagem a ser transmitida e

- computar um código de autenticação de mensagem baseado em cifra para a mensagem usando o algoritmo Rijndael, e

- reter o estado final dos *bytes* (29, 30) de pelo menos uma linha (32, 34) da matriz de estado (25) e descartando os *bytes* das linhas remanescentes, ou

- submeter pelo menos um *byte* de pelo menos dois *bytes* (s'_0-s'_{15}) de estado final do estado de matriz (25) com pelo menos um segundo *byte* de pelo menos dois *bytes* de estado final para uma operação de “OU exclusivo”.

2. Método de acordo com a reivindicação 1, caracterizado pelo fato de a matriz de estado (25) compreender exatamente quatro linhas (31-34) de *bytes* e exatamente quatro colunas (41-44) de *bytes*, exatamente quatro linhas de *bytes* e exatamente seis colunas de *bytes,* ou exatamente quatro linhas de *bytes* e exatamente oito colunas de *bytes*.

3. Método de acordo com a reivindicação 1, caracterizado pelo fato de compreender a geração da matriz de estado (25) de acordo com o Padrão de Encriptação Avançado (*Advanced Encryption Standard*).

4. Método de acordo com a reivindicação 1, caracterizado pelo fato de compreender a computação do código de autenticação de mensagem baseado em cifra pela retenção dos *bytes* (29, 30) de exatamente duas linhas (32, 34) da matriz de estado (25).

5. Método de acordo com a reivindicação 1, caracterizado pelo fato de a mensagem ser uma mensagem de bloco único.

6. Método de acordo com a reivindicação 1, caracterizado pelo fato de compreender a computação do código de autenticação de mensagem baseado em cifra ao submeter *bytes* (s'_0, s'_2, s'_4, s'_6, s'_8, s'_{10}, s'_{12}, s'_{14}) correspondentes a uma metade da mensagem com *bytes* (s'_1, s'_3, s'_5, s'_7, s'_9, s'_{11}, s'_{13}, s'_{15}) correspondentes à outra metade da mensagem à operação de “OU exclusivo”.

7. Método de acordo com a reivindicação 1, caracterizado pelo fato de compreender a computação do código de autenticação de mensagem baseado em cifra ao submeter *bytes* (s'_0, s'_2, s'_4, s'_6, s'_8, s'_{10}, s'_{12}, s'_{14}) correspondentes a uma metade da matriz de estado (25) com *bytes* (s'_1, s'_3, s'_5, s'_7, s'_9, s'_{11}, s'_{13}, s'_{15}) correspondentes à outra metade da matriz de estado (25) à operação de “OU exclusivo”.

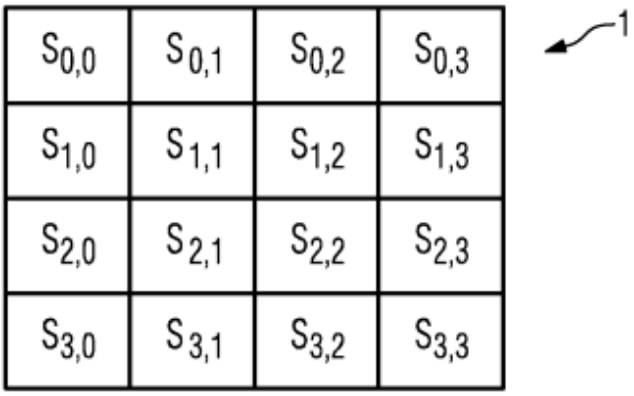


FIG. 1

W_n	?	B	?	A
W_{n-1}	?	D	?	C
$T(W_{n-1})$	S(D)<+>0x36	S(?)<+>0x00	S(C)<+>0x00	S(?)<+>0x00
W_{n-4}	?	?	?	?

FIG. 6

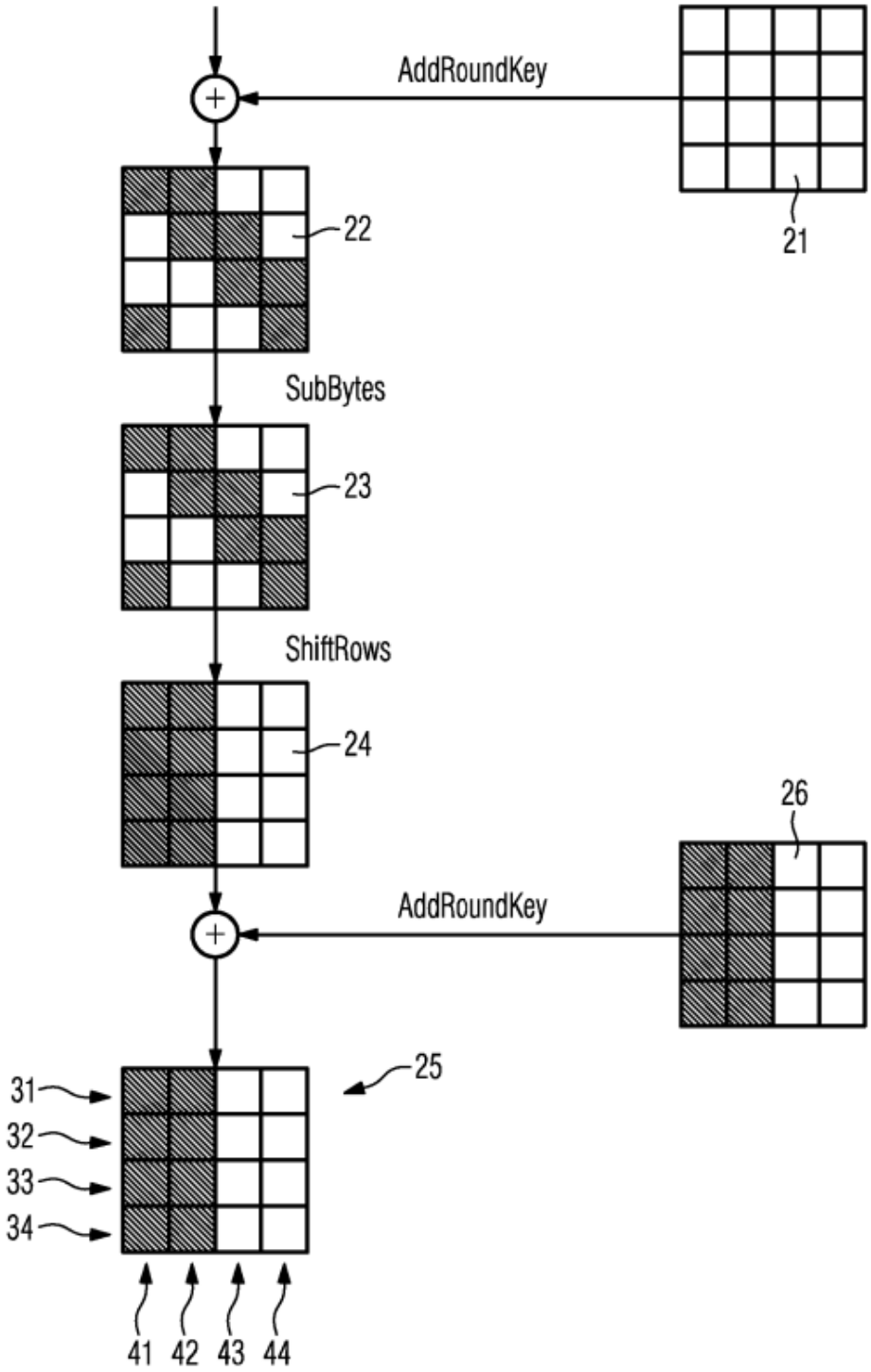

AddRoundKey
21
22
SubBytes
23
ShiftRows
24
26
AddRoundKey
25
31
32
33
34
41
42
43
44

FIG. 2

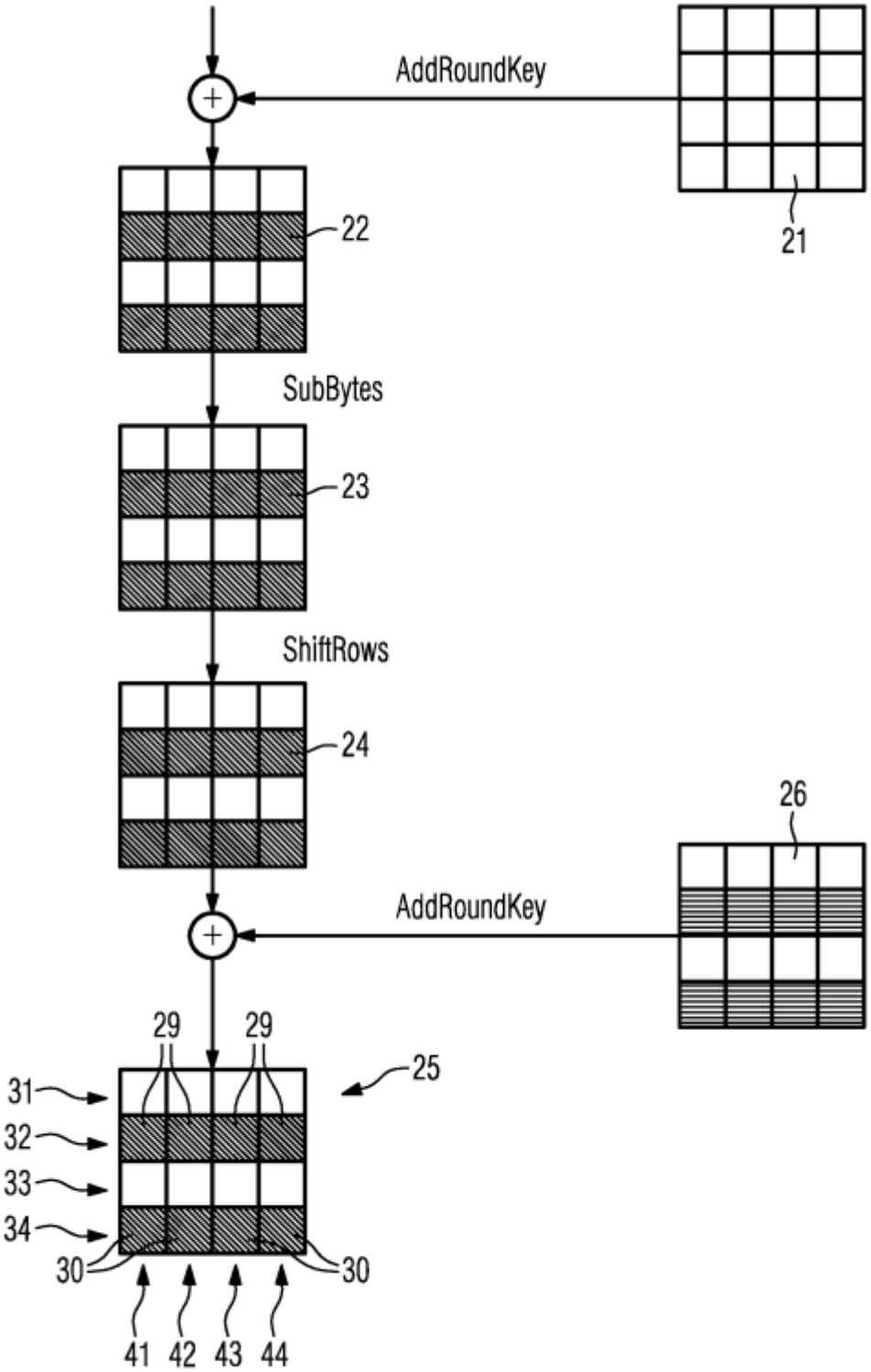

AddRoundKey
21
22
SubBytes
23
ShiftRows
24
26
AddRoundKey
29
29
25
31
32
33
34
30
30
41
42
43
44

FIG. 3

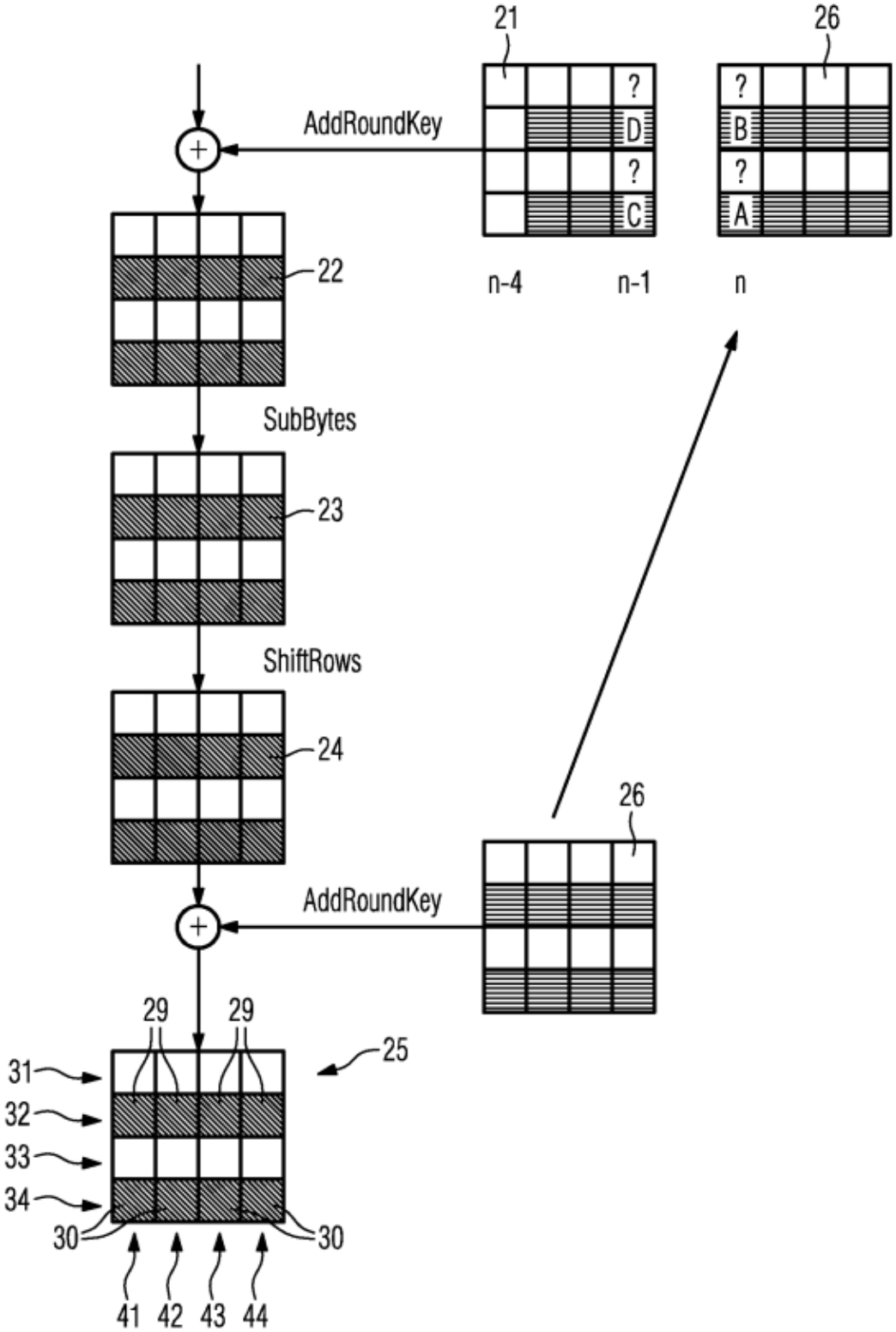

21
26
AddRoundKey
?
D
?
C
?
B
?
A
n-4
n-1
n
22
SubBytes
23
ShiftRows
24
26
AddRoundKey
29
29
25
31
32
33
34
30
30
41
42
43
44

FIG. 4

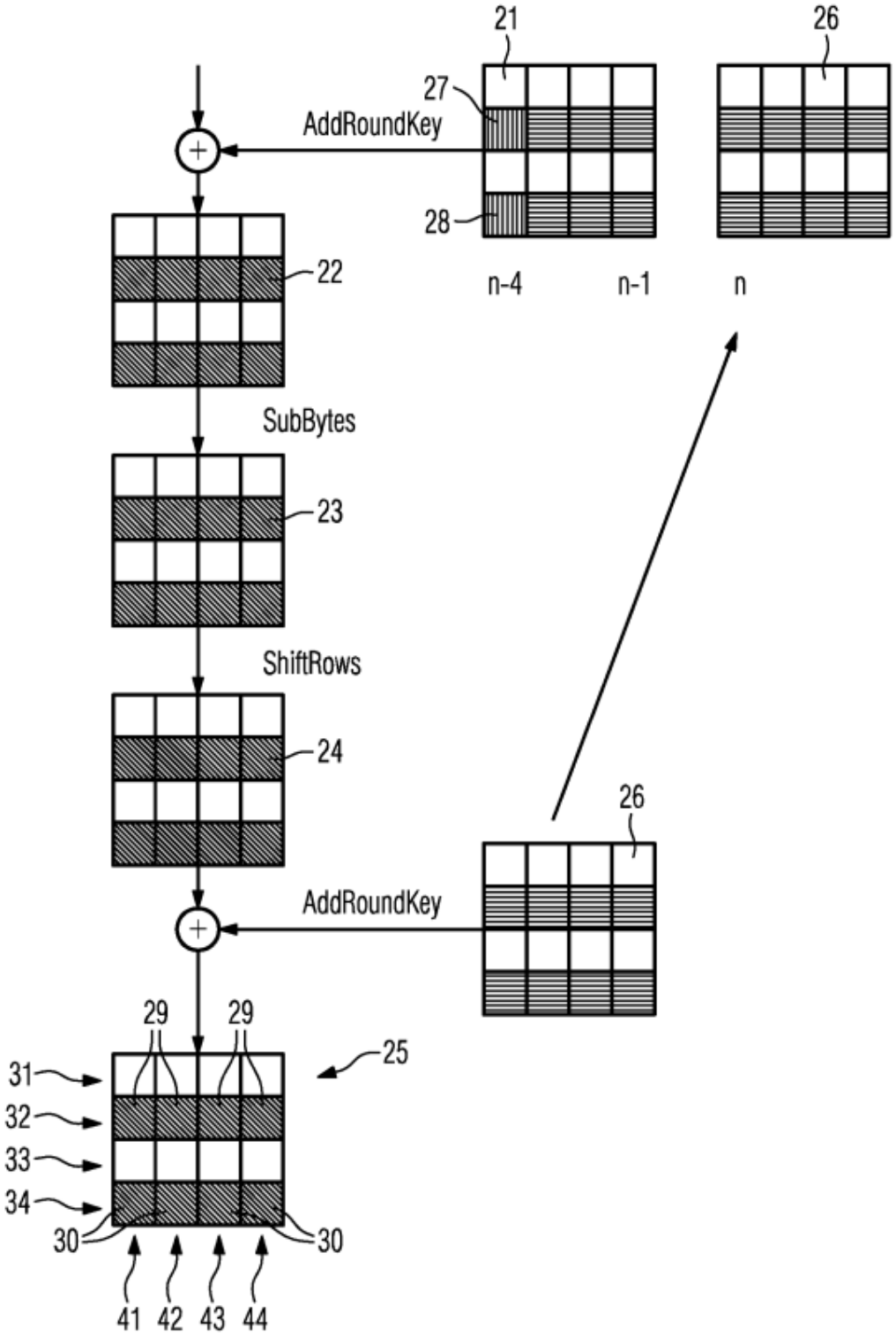

21
26
27
AddRoundKey
28
n-4
n-1
n
22
SubBytes
23
ShiftRows
24
26
AddRoundKey
29
29
25
31
32
33
34
30
30
41
42
43
44

FIG. 5

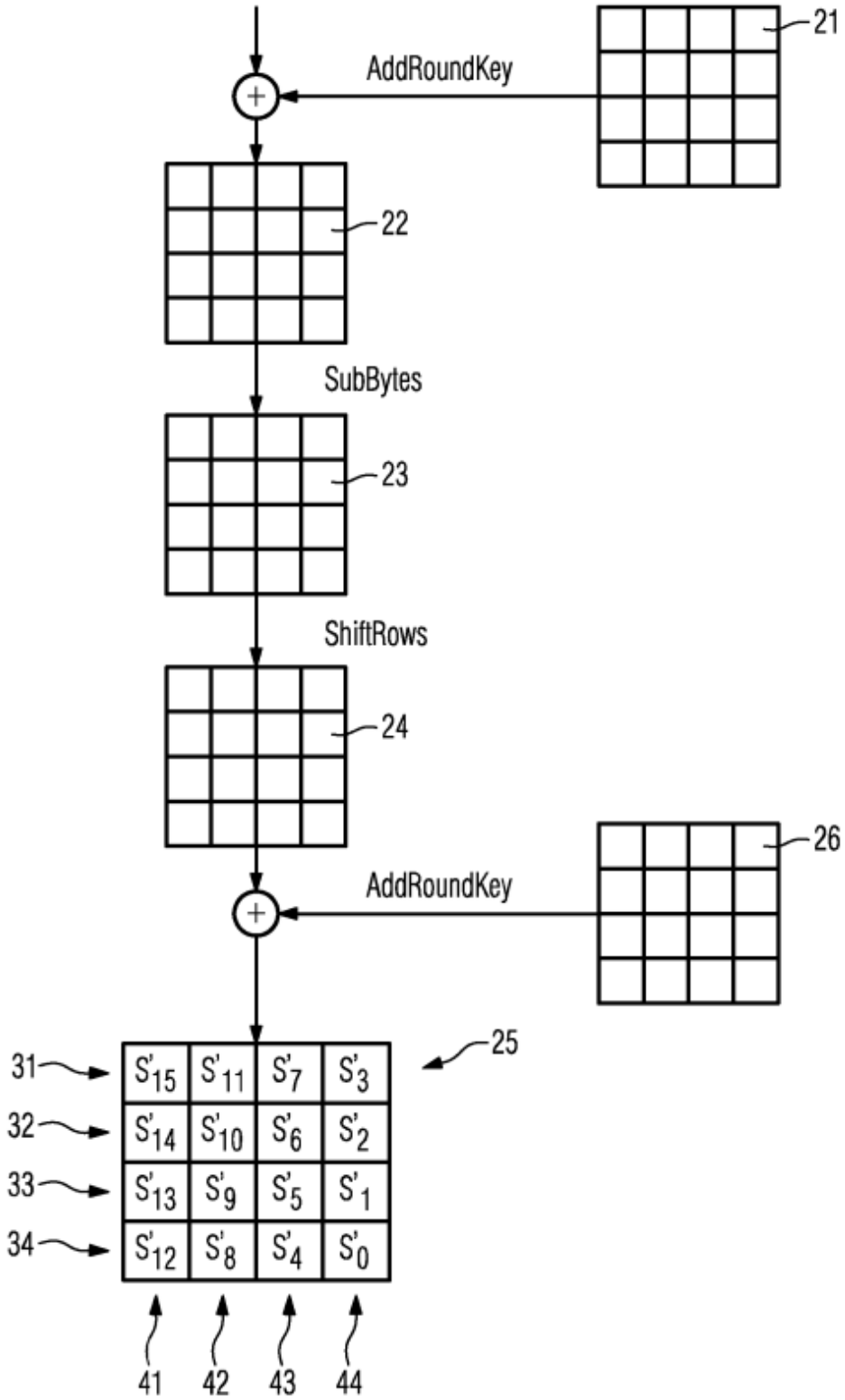

21
AddRoundKey
22
SubBytes
23
ShiftRows
24
26
AddRoundKey
25
31
32
33
34
S'15 S'11 S'7 S'3
S'14 S'10 S'6 S'2
S'13 S'9 S'5 S'1
S'12 S'8 S'4 S'0
41
42
43
44

FIG. 7