

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局



(43) 国際公開日
2010 年 9 月 30 日(30.09.2010)

PCT

(10) 国際公開番号
WO 2010/109560 A1

- (51) 国際特許分類:
H04L 9/08 (2006.01) H04L 9/32 (2006.01)
G06F 21/24 (2006.01)
- (21) 国際出願番号: PCT/JP2009/006158
- (22) 国際出願日: 2009 年 11 月 17 日(17.11.2009)
- (25) 国際出願の言語: 日本語
- (26) 国際公開の言語: 日本語
- (30) 優先権データ:
特願 2009-069435 2009 年 3 月 23 日(23.03.2009) JP
- (71) 出願人 (米国を除く全ての指定国について): 株式会社日立製作所(HITACHI, LTD.) [JP/JP]; 〒1008280 東京都千代田区丸の内一丁目 6 番 6 号 Tokyo (JP).
- (72) 発明者; および
- (75) 発明者/出願人 (米国についてのみ): ヴィオムカミーユ(VUILLAUME, Camille) [FR/JP]; 〒2150013 神奈川県川崎市麻生区王禅寺 1 0 9 9 番地株式会社日立製作所システム開発研究所内 Kanagawa (JP). 桶屋勝幸(OKEYA, Katsuyuki) [JP/JP]; 〒2150013 神奈川県川崎市麻生区王禅寺 1 0 9 9

番地株式会社日立製作所システム開発研究所内 Kanagawa (JP). 吉野雅之(YOSHINO, Masayuki) [JP/JP]; 〒2150013 神奈川県川崎市麻生区王禅寺 1 0 9 9 番地株式会社日立製作所システム開発研究所内 Kanagawa (JP).

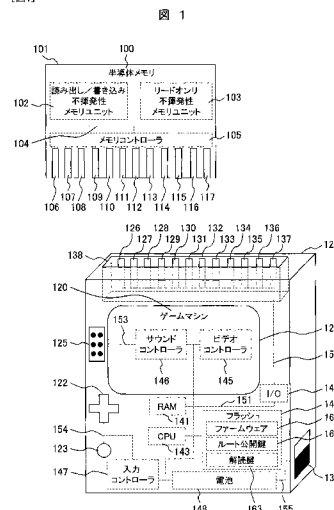
- (74) 代理人: ポレール特許業務法人(Polaire I.P.C.); 〒1040032 東京都中央区八丁堀二丁目 7 番 1 号 Tokyo (JP).
- (81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア

[続葉有]

(54) Title: SEMICONDUCTOR MEMORY, COMPUTER MACHINE CONNECTED WITH SEMICONDUCTOR MEMORY, AND DATA PROCESSING METHOD OF SAME

(54) 発明の名称: 半導体メモリ、当該半導体メモリと結合されるコンピュータマシン、及び、そのデータ処理方法

[図1]



- 100 semiconductor memory
- 102 read/write non-volatile memory unit
- 103 read-only non-volatile memory unit
- 105 memory controller
- 120 game machine
- 146 sound controller
- 145 video controller
- 142 flash
- 161 firmware
- 162 root public key
- 163 decryption key
- 148 battery
- 147 input controller

(57) Abstract: A machine wherein the processing of illegally redistributed digital data is prevented. A semiconductor storage medium is provided with digital data such as programs, images, music, and movies; a public key; a private key; and a circuit for handling communication with the computer machine; wherein the public key is derived from at least the digital data and private key. First, the semiconductor storage medium transfers the public key to the computer machine. Second, the computer machine sends a challenge (random number) to the storage medium. Third, the storage medium issues a digital signature for the challenge, then sends it to the computer machine. Fourth, the computer machine uses the public key to verify the digital signature. Fifth, the storage medium transfers the digital data to the computer machine. Sixth, the computer machine uses the public key to verify the digital data.

(57) 要約:

[続葉有]

WO 2010/109560 A1



(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ
(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, SE, SI, SK, SM, TR), OAPI (BF, BJ,
CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN,
TD, TG).

添付公開書類:

— 国際調査報告 (条約第 21 条(3))

非合法に再流通されたデジタルデータが処理されることを防止するためのマシンを提供する。半導体格納媒体は、例えば、プログラム、イメージ、音楽、映画など、デジタルデータと、公開鍵と、秘密鍵 (private key) と、そして、コンピュータマシンとの通信を操作する回路とを備えており、公開鍵が少なくともデジタルデータと秘密鍵 (private key) とから導き出される。第 1 に、半導体格納媒体は、公開鍵をコンピュータマシンへ転送する。第 2 に、コンピュータマシンがチャレンジ (乱数) を格納媒体へ送る。第 3 に、格納媒体がチャレンジのデジタル署名を発行し、そして、それをコンピュータマシンへ送り返す。第 4 に、コンピュータマシンが、公開鍵を使ってデジタル署名を検証する。第 5 に、格納媒体がデジタルデータをコンピュータマシンへ転送する。第 6 に、コンピュータマシンが、公開鍵を使ってデジタルデータを検証する。

明 細 書

発明の名称：

半導体メモリ、当該半導体メモリと結合されるコンピュータマシン、及び、そのデータ処理方法

参照による取り込み

[0001] 本出願は、平成21年（2009年）3月23日に出願された日本特許出願第69435号（特願2009-069435）の優先権を主張し、その内容を参照することにより、本出願に取り込む。

技術分野

[0002] 本発明は、デジタルコンテンツを格納するための半導体メモリと、半導体メモリに格納されたデジタルコンテンツを処理するためのコンピュータマシンに関し、より詳細には、本発明は、半導体メモリと、非合法的に再流通されたデジタルコンテンツを処理することを防止するコンピュータマシンと、そして、そのための処理方法に関する。

背景技術

[0003] 認証は、海賊行為、未認証のコピー、又は、偽造を取り締まるため、重要な役割を担っている。かかる認証は、例えば、パスポートの所有者を認証し、又は、反偽造を目的とした無線通信可能なチップを埋設した安全なパスポートにも採用されており、そこでは、認証用チップが製造物の出所を認証する。加えて、著作権で保護されたデジタルメディアは、海賊行為などの未認証行為を防止するため、DRM（digital right managements）として知られる認証用トークンを含んでいる。その結果、それらを支える機器（CD、DVD、メモリカード、その他の同様なメモリ装置）は、しばしば、認証メカニズムを備えている。最後に、認証は、競合者が製品をリバースエンジニアリングすることを防止し、かつ、製品を安価に大量生産するためにも有用である。例えば、競合者によって製造された再充填用（リフィル）キットや偽のカートリッジの使用を防止するため、認証用チップをプリンターカートリッジに装

着することもできる。

[0004] 認証は、被認証装置 (prover) がその特定のための証拠を認証装置 (verifier) に提示すプロトコルから構成されている。通常、認証装置は、被認証装置だけが解決可能な要求 (チャレンジ: challenge) を送付する。この要求が共通鍵暗号化法に依っている場合には、被認証装置及び認証装置は共通の秘密鍵を共有しており、それに対し、公開鍵暗号化法 (public-key cryptography) の場合には、被認証装置だけが秘密鍵を所有し、そして、どの認証装置も、被認証装置によって提供される要求の解を確認 (チェック) することが出来る。

[0005] 公開鍵暗号化システム、特に、デジタル署名は、認証の目的のために利用することができる。認証装置はランダムなメッセージを、その秘密鍵を利用してメッセージに署名した被認証装置に送信する。そして、認証装置は、秘密鍵ではなく、被認証装置の公開鍵だけを用いて署名を認証することによって、被認証装置の身元を確認することができる。公開鍵暗号化法に基づく認証システムでは、認証装置は被認証装置の秘密鍵を知らないことから、認証装置が被認証装置を装うことができないという利点を有している。加えて、公開鍵基盤を利用することにより、製品ラインの認証が、容易に可能となる。即ち、被認証装置は、要求に対してデジタル的な署名を行い、そして、その公開鍵と共に、認証局装置 (certificate authority) によって署名され、被認証装置の公開鍵のデジタル署名を含んだ被認証装置の公開鍵のための証明書 (certificate) を提供する。そして、認証装置は、被認証装置の身元を確認するために、両方の署名を認証することができる。かかるアプローチは、個別ユニットの公開鍵ではなく、証明のための認証手段 (例えば、全製品ラインのための単一の公開鍵であり、この公開鍵は製造会社から得られる。) を格納しなければならないという認証装置への格納要求を取り除くだけではなく、簡単な取り消し機構をも含んでいる。ある被認証装置の秘密鍵が知られ、その信用が失われた場合には、当該被認証装置の公開鍵は、全ての認証装置に対してブラックリスト内へ送ってもよい。

- [0006] 最近では、CDやDVDなどの媒体よりもよりフレキシブルであることを理由に、半導体メモリを、ゲームやコンピュータプログラムのデジタルデータのための格納手段として利用することに、再び、関心が集まっている。特に、半導体メモリユニットは、認証、コピープロテクション、データの暗号化などの機能性を向上するため、カスタム回路を含むことができる。

先行技術文献

特許文献

- [0007] 特許文献1：米国特許第6, 782, 477号：「プライベートコピープロテクションとオンラインセキュリティーを提供するための改ざん防止ハードウェアを使用するための方法とシステム」
特許文献2：米国特許第6, 606, 707号：「半導体メモリカード」

非特許文献

- [0008] 非特許文献1：ラルフ C. メルクレ：「認証されたデジタル署名」CRYPTO 1989：218-238頁
非特許文献2：カミーユ ヴィオム、カツユキ オケヤ、エリック ダーメン、そして、ヨハネス ブックマン：メモリトークンによる公開鍵認証、WISA 2008, LNCS 5379: 84-98頁

発明の概要

発明が解決しようとする課題

- [0009] 他方、例えば上記の特許文献1に開示されたデジタル署名は、プログラムの完全性(integrity)を、より具体的には、プログラムが悪意の者により変更されたことをチェックするために有用である。しかしながら、デジタル的に署名されたプログラムは、そのデジタル署名を含め、コピーされ、そして、第三者によって非合法に再流通される可能性がある。実際、かかる第三者は、プログラムとデジタル署名とを含んだ格納媒体と、プログラムを走らせるコンピュータマシンとの間のチャンネルを監視することが出来る。その結果として、そのプログラムとデジタル署名とを引き出し、そして、そのコピ

一を非合法に再流通することが可能となる。

[0010] 他方、認証は、ある特定の格納媒体が、例えば、ライセンスを介して、公認の配給者によって認証されていることを保障している。特許文献2は、認証のための回路を備えている半導体メモリカードを開示している。しかしながら、認証は、認証が成功した後に転送されたプログラムがその特定の格納媒体に確かに格納されていたことを保障するものではない。例えば、悪意の者は、認証を目的として合法的な格納媒体を利用し、その後、悪意の又は非合法に流通されたプログラムを、合法的な格納媒体に格納されたプログラムの代わりに挿入することも可能である。

[0011] 非特許文献2は、半導体メモリカードとコンピュータマシンとを開示しており、そこでは、半導体メモリカードとコンピュータマシンとが相互に認証プロトコルに従事し、そして、解読鍵を導き出す。その結果、解読鍵によって暗号化されたデータは、半導体メモリカードに源を発するものであることが保障される。しかしながら、コンピュータマシンと半導体メモリカードは、最初に、多くのコンピュータマシンが多くの半導体メモリカードと結合され得る場合には、実用的ではなく、共通のマスター共通鍵を割り当てなければならない。

[0012] 本発明は、上記で述べた従来技術を考慮して達成されたものである。従って、本発明の一の目的は、メモリユニットや最初に共通鍵を割り当てるコンピュータなしに、メモリユニット内に格納されたデジタルデータの完全性を確立し、もって、非合法に再流通されたデジタルデータが処理されるのを防止するためのマシンを提供することである。更に、上記及び他の目的と新規な特徴が、続く記載や添付の図面から明らかとなろう。

課題を解決するための手段

[0013] 上記の目的を達成するため、本発明によれば、まず、コンピュータマシンと結合されるための半導体メモリであって、第1のデータ、第1の鍵、そして、第2の鍵を格納し、前記第1の鍵は、前記第1のデータと前記第2の鍵とから導き出された所定の値であるところの不揮発性メモリユニットと、そ

して、コンピュータマシンからのチャレンジを受信し、そして、コンピュータマシンへ第2のデータを転送し、第2のデータが、前記第2の鍵と前記チャレンジとから導き出されるところのメモリコントローラとを備えている半導体メモリが提供されている。

[0014] また、本発明によれば、上述した目的を達成するため、半導体メモリと結合するためのコンピュータマシンであって、CPUと、第1の鍵を格納している不揮発性メモリユニットと、前記半導体メモリからのデータを受信するための入出力インターフェイスとを備えているものにおいて、前記半導体メモリからのデータは、第2の鍵と、第1のデータと、そして、第2のデータとを含んでおり、そして、前記コンピュータマシンは、更に、前記第2の鍵を検証するための署名検証ユニットと、チャレンジを発生するための乱数発生器と、前記第1と第2のデータを処理するためのハッシュ関数ユニットと、そして、前記ハッシュ関数ユニットによって処理されたデータと第2の鍵を比較するための比較ユニットを備えており、もって、第1と第2のデータの完全性を確立するコンピュータマシンが提供されている。

[0015] 更に、本発明によれば、上述した目的を達成するため、データ処理方法であって、半導体メモリからの第1の鍵を受信するステップと、前記半導体メモリへチャレンジを送信するステップと、前記半導体メモリから第1のデータを受信するステップと、前記チャレンジと前記第1の鍵により前記第1のデータの完全性をチェックするステップと、前記半導体メモリから第2のデータを受信するステップと、前記第1の鍵により前記第2のデータの完全性をチェックするステップと、そして、前記第2のデータを処理するステップを備えたデータ処理方法が提供されている。

[0016] 更には、本発明によれば、上述した目的を達成するため、データ処理方法であって、第1の鍵とその証明書を送信するステップと、少なくとも一のデータのハッシュ関数を送信するステップと、チャレンジを受信するステップと、複数の秘密値を送信するステップと、複数のイメージ値を送信するステップと、複数のツリー値を送信するステップと、そして、前記データを送信

するステップを備えているデータ処理方法が提供されている。

[0017] 上記に加え、ここに開示された実施例の中の好適なものを以下に略述する。

半導体格納媒体は、例えば、プログラム、イメージ、音楽、映画など、デジタルデータと、公開鍵と、秘密鍵（private key）と、そして、コンピュータマシンとの通信を操作する回路とを備えており、公開鍵が少なくともデジタルデータと秘密鍵（private key）とから導き出される。第1に、半導体格納媒体は、公開鍵をコンピュータマシンへ転送する。第2に、コンピュータマシンがチャレンジ（乱数）を格納媒体へ送る。第3に、格納媒体がチャレンジのデジタル署名を発行し、そして、それをコンピュータマシンへ送り返す。第4に、コンピュータマシンが、公開鍵を使ってデジタル署名を検証する、第5に、格納媒体がデジタルデータをコンピュータマシンへ転送する。第6に、コンピュータマシンが、公開鍵を使ってデジタルデータを検証する。

発明の効果

[0018] 秘密鍵（private key）を知らなくとも、チャレンジの公開鍵やデジタル署名を利用し、コンピュータマシンは、正当（合法）な格納媒体を偽造品から区別することができる。更に、公開鍵を利用して、デジタルデータの有効性をチェックできることから、非合法プログラムを挿入することはできない。その結果、本発明は、デジタルデータの認証されていないコピーの利用を防止する。

図面の簡単な説明

[0019] これら及びその他の目的、本発明の特徴や利点は、添付された以下の図面と関連させることにより、より容易に明らかとなろう。

[図1] ゲームマシンと、ゲームプログラムを格納するための半導体メモリとを備えたゲームシステムのハードウェアダイアグラムである。

[図2] 半導体メモリ内の論理メモリマッピングのブロックダイアグラムである。

[図3]半導体メモリ内のROMユニットの物理アドレスマップと、ROM領域内に格納されたデータとの機能的な関連を示すブロックダイアグラムである。

[図4]半導体メモリ内のメモリコントローラのハードウェアダイアグラムである。

[図5]メモリコントローラの機能詳細を示すブロックダイアグラムである。

[図6]ゲームマシンにより行われる検証プロトコルの機能詳細を示すブロックダイアグラムである。

[図7]ゲームマシンと半導体メモリとの間の通信プロトコルのブロックダイアグラムである。

[図8]ゲームマシンと半導体メモリとを備えたゲームシステムのハードウェアダイアグラムである。

[図9]半導体メモリ内での論理メモリマッピングのブロックダイアグラムである。

[図10]半導体メモリ内のメモリコントローラのハードウェアダイアグラムである。

[図11]メモリコントローラの機能詳細を示すブロックダイアグラムである。

[図12]ゲームマシン内のセキュリティーチップのハードウェアダイアグラムである。

[図13]ゲームマシンにより行われる検証プロトコルの機能詳細を示すブロックダイアグラムである。

[図14]ゲームマシンと半導体メモリとの間の通信プロトコルのブロックダイアグラムである。

[図15]ゲームマシンと半導体メモリユニットの製造プロセスのブロックダイアグラムである。

発明を実施するための形態

[0020] <<定義及び表記法>>

まず、以下に用いられる定義と表記法について述べる。

- [0021] 整数は、幾つかの表現を使って記述することができる。最も一般的な表現は十進法表現である。以下の図面における記述では、十進法表現に加えて、十六進法及び二進法表現が使用される。十六進法で書かれた場合、プレフィックス“0x”が添えられる。二進法で書かれた場合には、括弧“(…)”が書き添えられる。
- [0022] 例えば、「0xF」は整数 15 の十六進法表現である。同様に、「(1111)」は 15 の二進法表現である。
- [0023] デジタルデータは、ビット列であり、ビットストリングとも呼ばれる。同様に、ビットの連続は、バイト列としても表現することができ、1 バイトは 8 ビットから成る。例えば、ストリング(100101101)は 9 ビットを、又は、2 つのバイト、「0x01」と「0x2D」とを有している。
- [0024] 「x||y」は、ビットストリング「x」と「y」の連結(concatenation)に言及している。例えば、「0x012D=0x01||0x2D」のように記述する。
- [0025] 「x(+)y」は、ビットストリング「x」と「y」のビット毎の排他的ORを言及している（「(+)」は排他的ORの演算記号を表す）。「x >> 1」は、ビットストリング「x」の右への移動を言及する。
- [0026] ビットストリング「x」の最下位ビット(LSB)は、その二進表現の最右のビットである。
- [0027] ハッシュ関数は任意の長さのビットストリングを所定の長さの短いビットストリングにマッピング(map)するアルゴリズムである。例えば、周知のハッシュ関数である「MD5」は、任意のサイズのメッセージを 128 ビットのハッシュ値にマッピングする。この意味において、ハッシュ値は、大きな入力メッセージの小さな「指紋」とも考えられる。ハッシュ関数の他の重要な特性は、同一のハッシュ値を持った二つのメッセージを見つけ出すことは困難であるということであり、この特性は、「衝突困難性(collision-resistance)」と呼ばれる。以下の好適な実施の形態での記載においては、表記法「 $y = H(x)$ 」が使われるが、ここで「x」は入力メッセージであり、「H」はハッシュ関数であり、そして「y」はハッシュ値である。幾つかの周知の基本的なハ

ッシュ関数としては、例えば、「MD5」又は「SHA1」がある。更に、ハッシュ関数は、ブロック暗号化 (block ciphers) を含め (但し、それに限定されることない)、従来の暗号化関数 (encryption functions) によって実現することができることも周知のことである。例えば、周知のブロック暗号化「AES」は、以下のように使用することができる。メッセージ「x」が256ビットを有しているとすれば、かかるメッセージは二つの128ビットのブロック $x = x[1] || x[2]$ に分けることができ、128ビットの初期ベクトル (initial vector) が「IV」であれば、ハッシュ値「 $y = H(x)$ 」は、以下の二つのステップで計算される。

[0028] [数1]

$$1) \ y' = \text{AES}_{x[1]}(\text{IV}) \oplus \text{IV}$$

$$2) \ y = \text{AES}_{x[2]}(y') \oplus y'$$

[0029] 上記において、「 $\text{AES}_A(B)$ 」は、鍵「A」による「B」の「AES」暗号化に言及しており、「 $A(+)B$ 」は、「A」と「B」のビット毎の排他的OR (XOR)に言及している (「 $(+)$ 」は排他的ORの演算記号を表す)。

[0030] デジタル署名スキームは三つのアルゴリズムからなっている。最初の一つは、鍵対 (key-pair) の発生であり、ここでは秘密鍵 (private key) と公開鍵とが生成される。秘密鍵 (private key) は公開されず、一方、公開鍵は誰にでも配布される。第二のアルゴリズムは、署名発生アルゴリズムである。メッセージと秘密鍵とを入力として、デジタル署名を発生する。第三は、署名認証アルゴリズムであり、これはメッセージ、署名、そして、公開鍵を入力として、署名の真正 (correctness) を認証する。

[0031] 公開鍵は自由に入手可能であることから、誰でも署名を認証することはできるが、他方、有効な署名は、秘密鍵 (private key) の所有者だけがこれを発生することが出来ることに注目することが重要である。

[0032] 上記非特許文献1に開示されている「Merkle」の署名スキームは、デジタル署名を提供するための周知の技術である。他の署名スキームとは異なり、「Merkle」の署名スキームは、複雑な算術演算を利用することなく、ただ単

に、ハッシュ関数に依存するだけである。

[0033] 認証プロトコルは、被認証装置 (prover) 及び認証装置 (verifier) と呼ばれる二つの存在の間における通信プロトコルであり、そこでは、被認証装置がその身元 (identity) の証明 (proof) を与える。認証プロトコルは、少なくとも二つのステップから構成されており、認証装置がチャレンジ (challenge) を送付し、そして、被認証装置が秘密のデータを利用して応答 (response) を生成する。認証装置は、応答の有効性をチェックすることができ、そして、その結果、被認証装置の身元が正しいことを立証する。デジタル署名のスキームから認証プロトコルを構成することは単純であり、そこでは、チャレンジ (challenge) は乱数であり、そして、応答 (response) はそのデジタル署名である。例えば、上記非特許文献 2 には、「Merkle」署名を基礎とした認証技術が開示されており、そこでは、「Merkle」署名が、メモリ動作 (memory operations) だけで構成されている。

[0034] デジタル証明書 (digital certificate) は、公開鍵を確認するために有用なデジタル書面 (digital document) である。デジタル証明書は、公開鍵のデジタル署名を備えており、そのデジタル署名は、例えば、政府の機関や、その公開鍵が信頼されている確立された会社など、信頼された第三者 (trusted third party) により構成されている。加えて、証明書の階層を利用することが一般的であり、そこでは、ルート証明書 (root certificate) が、公開鍵を検証するのに利用され、そして、同様に、第二の認証が公開鍵を検証するのに利用される。この型の階層は、一般に、公開鍵基盤 (公開鍵インフラストラクチャ: public-key infrastructure)、又は、短縮して「PKI」と呼ばれる。「X.509」が、その構造やデジタル認証のフォーマットを規定するための周知の基準である。

[0035] 以下、本発明の好適な実施の形態について、添付の図面を参照しながら、詳細に述べる。

<<ゲームマシンと半導体メモリユニットの製造プロセス>>

[0036] ゲームマシンと半導体メモリの詳細を述べる前に、本発明のより良い理解

のため、ゲームマシンの製造プロセスと半導体について述べる。

[0037] 図15は、ゲームマシンと半導体メモリユニットの製造プロセスのブロック図である。このプロセスは、本発明の両方の実施例に適用可能である。

[0038] 如何なる特定の限定をも意図するものではないが、このプロセスでは、ゲームマシンの製造者1510、ゲームメーカ1530、そして、メモリ製造者1560という、三人の行為者を含んでいる。ゲームマシン製造者1510は、ゲームマシンを製造し、ゲームメーカ1530は、ゲームマシンにより実行されるソフトウェアプログラムを開発し、そして、メモリ製造者は、ゲームマシンにより実行されるソフトウェアプログラムを格納する半導体メモリユニットを製造する。

[0039] ゲームマシン製造者1510は、ルート秘密鍵 (root private key) 1511、ルート公開鍵 (root public key) 1512、そして、解読鍵 (decryption key) 1513を保持している。如何なる特定の限定をも意図するものではないが、本発明においては、ルート秘密鍵及び公開鍵は「RSA」鍵であり、そして、それらはゲームマシンの製造者によって所有されたコンピュータ上で発生され、かつ、格納されたものであるとする。幾つかの同一のゲームマシン1514が製造され、そして、商品化されており、全てのゲームマシンは、ルート公開鍵1512と解読鍵1513とを格納している。

[0040] ゲームメーカ1530がゲームマシン1514のため、対応する未暗号化ゲームヘッダ1533とゲームプログラムとを含めたビデオゲームを開発し、そして、そのゲームの千個のコピーを商品化することを希望したとする。しかしながら、ゲームメーカは、ゲームマシン製造者から、ルート証明書 (root certificate) 1515を含んだライセンスを得なければならない。

[0041] ゲームマシン製造者と同様に、ゲームメーカは、ゲームメーカが所有するコンピュータ上で発生され、そして、格納された、ゲーム秘密鍵 (game private key) 1531とゲーム公開鍵 (game public key) 1532とを保有している。最初に、ゲームメーカ1530は、ゲーム公開鍵1532を送信して、ゲームマシン製造者1510からライセンスを入手し；ゲームマシン製

造者は、解読鍵 1 5 1 3 とゲーム公開鍵 1 5 3 2 のデジタル署名を含むルート証明書 1 5 1 5 を、応答する。デジタル署名は、ルート秘密鍵 1 5 1 1 と共に、ゲームマシン製造者に帰属するコンピュータ上で発生される。また、アップデートデータ (data update) 1 5 1 6 が、ゲームプログラム内に含まれ、そして商品化されたゲームマシン 1 5 1 4 のファームウェアをアップデートするために送信される。

[0042] 次に、ゲームメカは、ゲームデータを暗号化するため、ゲーム鍵 1 5 3 5 を発生し、そして、解読鍵 1 5 1 3 を用いてゲーム鍵を暗号化する。また、ゲームメカは、ゲームヘッダ 1 5 3 3 とゲームプログラム 1 5 3 4 とを、ゲーム鍵 1 5 3 5 を用いて暗号化する。

[0043] その後、ゲームメカにより所有されるコンピュータ上では、例えば、1 5 4 1、1 5 4 3、1 5 4 5 などの暗号化データが、そして、例えば、1 5 4 2、1 5 4 4、1 5 4 6 などのゲーム証明書が発生される、各暗号化データは秘密鍵を含んでいる。より正確には、千の異なる暗号化データがランダムに発生される。また、千の異なる公開鍵が演算される。それに加え、千のゲーム証明書が発生され、各ゲーム証明書は、千の公開鍵の中の一つのデジタル署名を含み、そして、当該署名はゲーム秘密鍵 1 5 3 1 を用いて発生される。

[0044] 最後に、ゲームメカは、千の異なる暗号化データと千の異なるゲーム証明書とを、一意の暗号化されたゲーム鍵 1 5 4 7 (unique version)、一意の暗号化されたゲームヘッダ、そして、一意の暗号化されたゲームデータとを用いて結合する。この演算の結果は、例えば、1 5 5 1、1 5 5 2、1 5 5 3 のような、メモリ製造者 1 5 6 0 へ出荷される千のメモリイメージである。メモリ製造者の役割は、単純に、受信したゲームのメモリイメージを、例えば、1 5 6 1、1 5 6 2、1 5 6 3 のような千の半導体メモリユニット内にコピーすることであり、そして、それらを商品化することである。

[0045] ユーザが原品であるゲームマシンを、そして、原品である半導体メモリを購入した場合、二つの電子装置は、図 7 又は図 1 4 に記載された認証プロト

コルを開始し、そして、ユーザはゲームメーカ 1 5 3 0 が開発したゲームで遊ぶことが可能となる。

- [0046] 上述したプロセスに替え、暗号化されたデータは、メモリ製造者によって、直接、発生することも可能であろう。この場合、メモリ製造者は、ゲーム秘密鍵と公開鍵とを保持し、ゲームマシン製造者からルート証明書を発生する。

<<実施例 1>>

- [0047] 最初に、メモリコントローラを備えた半導体メモリに焦点を当てた説明を行うが、ここでは、メモリがゲームマシンによって実行されるプログラムを格納している。

<ゲームシステム>

- [0048] 図 1 はゲームシステムのハードウェアダイアグラムであり、ゲームマシン 1 2 0 と、ゲームプログラムを格納するための半導体メモリ 1 0 0 とを備えている。如何なる特定の限定をも意図するものではないが、この実施例では、ゲームマシン 1 2 0 は、手で保持できるポータブルなゲームマシンであるものとする。

- [0049] 半導体メモリ 1 0 0 は、プラスチックの覆い 1 0 1 の内部に取り囲まれた電子部品を含んでいる。図 1 では、破線により示された部品はプラスチックの覆い 1 0 1 の内部に配置されている。

- [0050] 半導体メモリ 1 0 0 は二つのタイプのメモリユニット、即ち、読み書きが可能な不揮発性のメモリユニット (read-write non-volatile memory unit) 1 0 2 と、読み出し専用の不揮発性のメモリユニット (read-only non-volatile memory unit) 1 0 3 とを備えている。如何なる特定の限定をも意図するものではないが、この実施例では、読み書き可能な不揮発性のメモリユニットは「EEPROM」チップであり、他方、読み出し専用のメモリユニットは、1 回だけプログラマブルな「ROM」チップであるものとする。「ROM」チップ 1 0 3 は、ゲームプログラムを格納することができ、「EEPROM」チップ 1 0 2 は、セーブされたゲーム状態を格納することができる。メモリユニット 1 0

2と103の詳細な内容は、後に、図2を参照しながら述べる。

[0051] 半導体メモリ100は、メモリユニット102と103をメモリコントローラ105に結合するためのバス104を備えている。メモリコントローラ105は、ゲームマシン120との通信を操作し、そして、メモリユニット102と103内に格納されたデータにアクセスする回路である。通信プロトコルは、後に図7を参照しながら説明する。最後に、半導体メモリ100は、物理的には、ピン106を介して、ゲームマシン120に結合することができる。本実施例においては、12本のピンは、特に限定されるものではないが、以下のようにになっている。

- ー ピン106は、入力接地信号 (input ground signal) 、
- ー ピン107は、入力電源供給ピン (input power supply pin) 、
- ー ピン108は、入力クロック信号 (input clock signal) 、
- ー ピン109は、入力リセット信号 (input reset signal) 、
- ー ピン110～117は、ゲームマシン120と半導体メモリ100との間の双方向のデータ交換のために利用されるものである。

[0052] ゲームマシン120は、プラスチックの覆い121、例えば方向性パッド (directional pad) 122やボタン123のような入力コントローラ、液晶 (LCD) スクリーン124、スピーカ125、そして、オン／オフスイッチ139を有している。半導体メモリ100が空洞部138内に挿入可能となっており、それによって、ゲームマシン120と半導体メモリ100とを物理的に結合するための連結が得られる。本実施例では、12個のコンタクトは、特に限定されるものではないが、以下の通りである。

- ー コンタクト126は、出力接地信号 (output ground signal) 、
- ー コンタクト127は、出力電源供給ピン (output power supply pin) 、
- ー コンタクト128は、出力クロック信号 (output clock signal) 、
- ー コンタクト129は、出力リセット信号 (output reset signal) 、
- ー コンタクト130～137は、ゲームマシン120と半導体メモリ100との間の双方向のデータ交換のために利用されるものである。

- [0053] 即ち、コンタクト１２６～１２９は、ゲームマシン１２０から半導体メモリ１００への一方向の通信のために使用され、これに対して、コンタクト１３０～１３７は、双方向の通信に使用することができる。
- [0054] プラスチックの覆い１２１内には幾つかの電子部品がある。ＲＡＭ(Random Access Memory)１４１は、アクセスの早い、読み書き可能な揮発性メモリユニットであり、プログラムデータを共に、ＣＰＵ(Central Processing Unit)１４３によって処理された中間データをも格納する。フラッシュメモリユニット１４２は、読み書き可能な不揮発性メモリユニットである。フラッシュメモリ内に格納されるデータには、以下のものを含んでいる：
- ー ゲームマシンのハードウェアを制御するプログラムである、ファームウェア１６１、
 - ー ルート公開鍵１６２、
 - ー 解読鍵１６３。
- [0055] 入出力（Ｉ／Ｏ）インターフェイスユニット１４４は、ゲームマシン１２０と半導体メモリ１００との間の通信を操作する回路である。
- [0056] ビデオコントローラ１４５は、スクリーン１２４に取り付けられた回路であり、映像をスクリーン１２４に対して発生し、そして、出力する。サウンドコントローラ１４６は、音、又は、音楽を表現するデジタル情報をアナログ電気信号に変換し、そして、その信号はスピーカ１２５によって音響信号に変換される。入力制御ユニット１４７は、ＣＰＵ１４３によって処理することが出来るように、例えば、方向性パッド１２２又はボタン１２３に加えられた圧力である入力信号を、デジタル情報に変換する。
- [0057] 最後に、電池１４８は、ゲームマシン内に封入された電子部品に対して電力を供給する。
- [0058] ＲＡＭ、フラッシュメモリ、ＣＰＵ、Ｉ／Ｏインターフェイス、ビデオコントローラ、サウンドコントローラは、異なる電子部品への制御信号やデータ信号、そして、そこからの制御信号やデータ信号を伝達するためのバス１５１に結合されている。加えて、電力が電池１４８により、バス１５１を介

して電子部品へ供給されている。同様に、コンタクト１２６～１３７、そして、Ｉ／Ｏインターフェイス１４４が、Ｉ／Ｏバス１５２に結合されている。スピーカ１２５とサウンドコントローラ１４６は、ケーブル１５３と結合されている。方向性パッド１２２、ボタン１２３、そして、入力インターフェイス１４７が、ケーブル１５４と結合されている。最後に、電池１４８とオン／オフスイッチ１３９が、ケーブル１５５に結合されている。

[0059] ユーザがスイッチを利用してゲームマシン１２０の電源をオンすると、フラッシュメモリ１４２に格納されたファームウェア１６１がＣＰＵ１４３により実行される。半導体メモリ１００が空洞１３８内に挿入され、それにより半導体メモリ１００のピン１０６～１１７をゲームマシン１２０のコンタクト１２６～１３７に結合すると、ゲームマシンとメモリとが連動し、図７を参照して後に述べる通信プロトコルを行う。通信プロトコルの普通に終了した後、半導体メモリ内に格納されていたプログラムがＲＡＭ１４１に転送され、そして、ＣＰＵ１４３により実行される。プログラムは、例えば、映像をスクリーン１２４上に表示し、スピーカ１２５を介して音楽や音を鳴らし、そして、方向性パッド１２２又はボタン１２３を押しているユーザの動きに反応することができる。しかしながら、通信プロトコルが失敗した場合には、プログラムは実行されない。

[0060] 図２は、半導体メモリ１００の内における論理メモリマッピングのブロックダイアグラムである。

[0061] 如何なる特定の限定をも意図するものではないが、本実施例では、半導体メモリ１００の論理アドレスが４８ビットの整数として表現されており、図２の左側に１６進法により表記されている。これに替えて、異なるアドレススペースを使用することもでき、その場合、データは、物理的なチップ上の異なるサイズと異なるロケーションで、異なる順序で格納することができる。

[0062] 半導体メモリ１００上に格納されたデータをアクセスするために、ゲームマシンは、論理アドレスとデータ長さとを半導体メモリへ送信する。次に、

メモリコントローラ 105 が論理アドレスを物理アドレスに翻訳（変換）し、例えば、ROM ユニット 102 又は EEPROM 103 など、関連するメモリユニットを作動し、そして、関連するメモリユニット上に格納されているデータをゲームマシン 100 へ送出する。

[0063] 48 ビットのアドレス値は 256, 000 ギガバイトのデータにアドレスすることができる。しかしながら、半導体メモリは、例えば、1 ギガバイト程度の、より小さな容量を有している。この場合、論理メモリのある部分だけが、実際に有効なアドレスである。図 2 において、許されていないアドレスは、斜線によって表示されている。

[0064] 論理アドレスに加え、図 2 は、ROM ユニット 102、EEPROM ユニット 103、又は、メモリコントローラ 105 のうち、どのメモリユニット又はチップ上に、対応するデータが物理的に格納されているかを示している。メモリの特性（読出し専用か、又は、読み書き両方か）に対応して、ある論理アドレスに格納されたデータは読み出し可能であり、そして、時には書き込みも可能である。最後に、データへのアクセスは、アクセス制御ポリシー（access control policy）によって規定されており、図 2 において大文字「U」によって示されるユーザアクセスと、大文字「S」によって示される管理者アクセスとに分けられる。ユーザアクセスとは、ゲームマシン 120 によって直接アクセスされるデータを言い、他方、管理者アクセスとは、直接にはアクセスすることが出来なデータを意味する。より詳細には、メモリコントローラだけが管理者データにアクセスすることができる。

[0065] 論理アドレスは、以下の五つの領域に分類される。

- ー 暗号化データ領域 200 は、半導体メモリ 100 の認証の期間に利用される。
- ー 暗号化ヘッダ領域 210 は、認証の期間に転送されたデータと組み合わせられ、半導体メモリの正真性（authenticity）を保証するパブリックパラメータを含んでいる。
- ー ゲームヘッダ領域 220 は、半導体メモリ内に格納されたゲームプログ

ラムに関連する一般的な情報を含んでいる。

ー セーブゲーム状態領域 230 は、ユーザによってセーブされたゲームの状態を含んでいる。

ー ゲームプログラム領域 240 は、最初のセクタ 241 にファームウェアアップデートを、そして、他のセクタにゲームプログラムを格納している。

[0066] 上述した領域が、以下に詳細に述べられる。

[0067] 暗号化データ 200 は、秘密テーブル 201、イメージテーブル 202、そして、ツリーテーブル 203 を含んでいる。秘密テーブル 201 は、「0x0」から「0x3FFFFFF」までのアドレス値を、即ち、4 メガバイトを占有している。物理的には、秘密テーブル 201 からのデータは、ROM ユニット 102 内に配置されている。加えて、秘密テーブル 201 へのアクセスは、管理者アクセスポリシー (supervisor access policy) によって管理することから、制限される。

[0068] イメージテーブル 202 は、「0x400000」から「0x7FFFFFF」までのアドレス値を占有し (4 メガバイト)、そして、ツリーテーブル 203 は、「0x800000」から「87FFEF」までを占有している。物理的には、それらは ROM ユニット 102 内に配置され、自由にアクセス可能である。

[0069] それらのテーブル内に格納されたデータについては、後に、図 3 を参照しながら述べる。

[0070] 以下には、暗号化ヘッダ 210 が述べられる。アドレス「0x1000000」には、システムパラメータ 211 が格納されている。システムパラメータは、どのタイプのアルゴリズムを認証のために使用するのか、認証に使用するチャレンジのサイズ、認証を行うことができる総回数、そして、どのタイプの証明書が使用されるのかを、一義的に規定している。本実施例では、特に制限はないが、認証アルゴリズムは、ハッシュ関数「MD5」と共に、8 ビットのチャレンジを使用する「Merkle」署名に基づくことができ、16,384 回まで認証を行なうことができ、そして、証明書は「X.509」のデジタル証明書でも可能であるとしている。その場合、システムパラメータは、以下のような

「ASCII」キャラクターで記述されたストリングから構成される：

- MD5_WITH_MERKLE
- CHALLENGE_8
- TIMES_16384
- CERTIFICATE_X509

[0071] それに替えて、例えば、「SHA-1」又は「AES」のようなブロック暗号化に基づくものなど、「MD5」とは異なるハッシュ関数を使用することもでき、チャレンジのサイズや行われる認証の回数が異なっても、そして、証明書のタイプも異なってもよい。

[0072] 公開鍵 2 1 4 は、アドレス「0x100900」に格納される。公開鍵は、後に図 7 を参照しながら詳細に記載する認証プロトコルにおいて、役割を演ずる。認証アルゴリズムがハッシュ関数「MD5」と共に「Merkle」署名を使用する場合には、公開鍵は 1 6 バイトの二進データから構成される。

[0073] ゲーム証明書は、アドレス「0x100600」に格納される。ゲーム証明書は、公開鍵を有効にするために使用される。ルートゲーム証明書は、アドレス「0x100300」に格納され、そして、ゲーム証明書を有効にするために使用される。手順の詳細は、後に、図 7 を参照しながら述べる。システムパラメータ 2 1 1、ルート証明書 2 1 2、ゲーム証明書 2 1 3、そして、公開鍵 2 1 4 は、物理的に ROM ユニット 1 0 2 内に格納されており、そして、自由にアクセス可能である。

[0074] 2 バイトの署名インデックス 2 1 5 は、アドレス「0x109400」に格納される。物理的には、署名インデックス 2 1 5 は、後に図 4 を参照しながら述べるメモリコントローラ 1 0 5 内における 2 バイトの不揮発性の読み書き可能なレジスタ内に配置される。署名インデックスは、秘密テーブル 2 0 1、イメージテーブル 2 0 2、そして、ツリーテーブル 2 0 3 で使用される暗号化データの配置を示している。

[0075] ゲームヘッダ領域の内容を以下に述べる。かかる内容は、物理的には、ROM ユニット上に格納されており、そして、自由にアクセス可能である。

- [0076] ゲームID221は、アドレス「0xFF0000000000」に格納されており、ゲームIDは、どのゲームが半導体メモリ上に格納されているかを一義的に規定する。ゲーム鍵222は、アドレス「0xFF0000000010」に格納される。ゲーム鍵は、ゲームヘッダデータの残りの部分と共に、ゲームプログラム240をも解読するために使用される。ゲームプログラムの権限のない解読を防止するため、ゲーム鍵221は、ゲームマシン120のフラッシュメモリユニット142内に格納された解読鍵163を使用して暗号化されている。
- [0077] その後のアドレスにおけるデータが、ゲーム鍵222により暗号化される。プログラムのエントリポイント223 (program entry point) は、アドレス「0xFF0000000020」に格納されている。このプログラムのエントリポイントは、ゲームプログラムのメインルーチンのアドレスを示している。それ故、認証が成功した後は、ゲームマシン120は、プログラムのエントリポイントに特定されたアドレスに配置されているプログラムデータを読み込む。
- [0078] 16バイトを占有するアドレス「0xFF0000000030」には、第1のハッシュデータ224が格納される。このデータは、ゲームプログラム領域240の第1セクタ内に格納されたデータに適用された「MD5」ハッシュ関数の出力である。より詳細には、第1のハッシュデータ224は、アドレス「0xFFFF00000000」における1メガバイトのデータを連結した結果であり、それを「MD5」により処理し、もって、16バイトのハッシュ値を得る。ゲームプログラム領域240には、合計1024セクタが存在することから、アドレス「0xFF0000000030」から「0xFF000000402F」まで、1024個のハッシュ値が存在する。
- [0079] セーブゲーム状態230は、アドレス「0xFFFF00000000」から「0xFFFF00002FFFF」までに格納される。物理的には、それらのアドレスはEEPROMに対応しており、それ故、セーブゲーム状態領域230内のデータは、読み出し、又は、書き込みができる。更に、データは、セーブゲーム状態領域230内において、自由にアクセス可能である。第1のセーブゲーム232から

16番目233まで、16のセーブ用スロットがある。状態テーブル231は、各セーブゲームスロットの状態、即ち、アクティブか又は非アクティブかを示している。

[0080] 最後に、ゲームプログラム領域が、アドレス「0xFFFF00000000」から「0xFFFF3FFFFFFF」まで広がっており、それらは物理的にROMユニット102に対応している。ゲームプログラムデータは自由にアクセス可能であるが、しかしながら、ゲーム鍵222によって暗号化されている。ゲームデータは、1024のセクタに編成されており、各セクタは1024バイトを含んでいる。その結果、半導体メモリは、1ギガバイトまでゲームプログラムを格納することができる。ファームウェアのアップデートは、ゲームプログラム領域の最初のセクタ241内に格納されている。

[0081] データサイズを略述すると、ROMユニット102は、1ギガバイトよりも僅かに大きい、1,082,709,024バイトを収容しており、そして、EEPROMは、196,608バイトを収容することが分かる。

[0082] 図3は、ROMユニット103の物理的アドレスマップと、そして、ROM領域内に格納されたデータとの機能的な関係を表すブロックダイアグラムである。

[0083] 以下に記載される物理的アドレスのマップに代えて、異なる順番で、又は、異なる物理的配置に格納されたデータにより、異なるアドレスの仕方を利用することもできる。

[0084] 以前にも説明したように、ROMユニット103は、暗号化データ200、暗号化ヘッダ210の一部、ゲームヘッダ220、そして、ゲームプログラム240を格納している。他方、ゲームマシン120は、48ビットのアドレスを使用して、半導体メモリの利用可能なメモリユニットに対し、データを読み出し又は書き込むことができる。他方、内部的には、各メモリユニットはそれ自身の物理的アドレスマップを有している。如何なる特定の限定をも意図するものではないが、本実施例では、ROMユニット103は、32ビットのアドレス空間を使用している。ゲームマシン120によって送ら

れた論理アドレスを物理アドレスへ変換すること、そして、半導体メモリ 100の対応するメモリユニットを作動することが、メモリコントローラ 105の役割である。

[0085] 暗号化データ 200は、秘密テーブル 201、イメージテーブル 202、そして、ツリーテーブルの、三つのテーブルを含んでいる。

[0086] 秘密テーブル 201は、262, 144個の16バイトの乱数、即ち、「x[1]」301～「x[262144]」306によって満たされている。秘密「x[1]」は、物理アドレス「0x0」に配置され、秘密「x[2]」は「0x10」に、そして、秘密「262144」がアドレス「0x3FFFF0」に配置されるとまで、以下同様になっている。

[0087] イメージテーブル 202は、262, 144個の各16バイトの値、即ち、「y[1]」311～「y[262144]」316によって満たされている。如何なるインデックス「 $1 \leq i \leq 262,144$ 」に対しても、イメージ「y[i]」が「y[i]=H(x[i])」として規定されており、ここで「H」はハッシュ関数「MD5」である。換言すれば、「y[i]」は、ハッシュ関数「MD5」を介した秘密「x[i]」のイメージであり、例えば、「y[1]=H(x[1])」が成り立つ。イメージ「y[1]」は、物理アドレス「0x400000」に配置され、イメージ「x[2]」はアドレス「0x400010」に、そして、アドレス「0x7FFFF0」に配置されたイメージ「y[262144]」に至るまで、以下同様となっている。

[0088] ツリーテーブル 203は、各々が16バイトの、32767個の値、即ち、「t[1]」321～「t[32767]」328を含んでおり、ここでは、「t[1]」はアドレス「0x800000」に配置され、「t[2]」は「0x800010」に、そして、アドレス「0x87FFF0」に配置された「t[32767]」に至るまで、以下同様となっている。ツリーデータは、その機能的な関係に応じて、二つのカテゴリに分類することができる。

[0089] ツリーテーブル 203の上部、より正確には、「t[16384]」から「t[32767]」までにおいては、ツリーデータはイメージテーブル 202からのイメージデータに依存しており、如何なるインデックス「 $0 \leq i \leq 16383$ 」に対しても、

ツリーデータ「 $t[i+16384]$ 」は、以下のように規定される。

[0090] [数2]

$$t[i+16384] = H(y[16*i+1] || y[16*i+2] || \dots || y[16*i+16])$$

[0091] 例えば、「 $t[16384]=H(y[1] || y[2] || \dots || y[16])$ 」、又は「 $t[16385]=H(y[17] || y[18] || \dots || y[32])$ 」である。

[0092] ツリーテーブル203の下部においては、如何なるインデックス「 $1 \leq i \leq 16383$ 」に対しても、ツリーデータ「 $t[i]$ 」は、以下のように規定される。

[0093] [数3]

$$t[i] = H(t[2*i] || t[2*i+1])$$

[0094] 例えば、「 $t[1]=H(t[2] || t[3])$ 」、又は、「 $t[8192]=H(t[16384] || t[16385])$ 」である。

[0095] 以上の機能的な関係から、ツリーデータ203の配列は、二分のハッシュツリーとして見ることができ、そこでは、ツリーの根（ルート：root）は、テーブル「 $t[1]$ 」の先頭値であり、その葉（leaves）は、「 $t[16384]$ 」、「 $t[16385]$ 」…「 $t[32676]$ 」である。加えて、葉を除いて、全てのノード「 $t[i]$ 」は、「 $t[i] = H(t[2*i] || t[2*i+1])$ 」なる機能的関係に従って、ノード「 $t[2*i]$ 」と「 $t[2*i+1]$ 」に接続されている。

[0096] 暗号化ヘッダ210は、アドレス「0x87FFF0」から出発し、そして、アドレス「0x8893F0」で終わる。暗号化ヘッダ210は、ROM103内には、部分的にしか格納されていないことに留意のこと。より詳細には、システムパラメータ211、ルート証明書212、ゲーム証明書213、そして、公開鍵214がROMユニット103内に格納されているが、しかし、署名インデックス215はメモリコントローラ105内に格納されている。

[0097] 公開鍵214は、ゲームヘッダ220のハッシュ値と連結されたツリーテーブル「 $t[1]$ 」321の先頭のハッシュ値であると規定される。

[0098] [数4]

$$\text{Public key} = H(t[1] || H(\text{game header}))$$

[0099] ゲームヘッダ220は、ROMユニット103内に完全に格納され、アド

レス「0x8893F0」から出発し、アドレス「0x88D420」までである。最後に、ゲームプログラムは、アドレス「0x88D420」から始まってアドレス「0x4088D420」までである。

<メモリコントローラ>

[0100] 図4は、メモリコントローラ105のハードウェアダイアグラムである。

[0101] メモリコントローラは、「READ」、「WRITE」、そして、「CHALLENGE」の、三つのタイプのメッセージを取り扱う。

ー 「READ」メッセージはアドレスとデータ長さとを含んでおり、「READ||Address||Length」のようにフォーマットされ、そこでは、アドレスは48ビットのアドレスデータであり、長さは16ビットである。「READ」メッセージを受信すると、メモリコントローラは、アドレスと長さが読み出し可能なデータに対応することを確認し、対応するメモリユニットからデータを取り出してそれをゲームマシン120へ送り返す。

ー 「WRITE」メッセージは48ビットのアドレスと8ビットのデータペイロード（data payload）を含んでおり、「WRITE||Address||Data」のようにフォーマットされる。「WRITE」メッセージを受信すると、メモリコントローラ105は、アドレスがEEPROMの書き込み可能なメモリに対応することを確認し、そして、そのデータペイロードをEEPROM領域に書き込む。

ー 「CHALLENGE」メッセージは8ビットのデータペイロード、即ち、チャレンジを含んでおり、「CHALLENGE||Data」のようにフォーマットされる。かかるデータを受信すると、メモリコントローラ105は秘密テーブル201からデータを選択し、そして、それをゲームマシン120へ送り返す。

[0102] 上述したメッセージを如何に処理するかについて、図5を参照しながら詳細に述べる。

[0103] メモリコントローラ105は、ピン106～117を介して、ゲームマシン120と交信する。特に、ピン110～117は双方向のデータ交信のために使用されており、ゲームマシン120から受信され又はゲームマシンへ送出されるデータは、8ビットのI/Oバッファ401内に格納される。接

地ピン１０６、電力供給ピン１０７、クロックピン１０８、リセットピン１０９、そして、Ｉ／Ｏバッファは、全て、制御ユニット４１０へ結合されている。

[0104] チャレンジレジスタ４２１と共に、シフトユニット４２２、秘密アドレスレジスタ４３０、三つの加算器４４１、４４２、４４３、そして、制御ユニット４１０が、図５を参照して詳細に述べるメッセージの処理機能を実現している。この制御ユニット４１０は、以下のものを備えている。

- ー メモリコントローラ１０５内の他のユニットを作動、制御し、そして、受信したメッセージに応じて信号を送信する、有限状態マシン４１１。
- ー 読み出し、又は、書き込まなければならないデータの物理アドレスを格納するアドレスレジスタ４１２。
- ー 読み出されるデータの長さを格納する長さレジスタ４１３。
- ー 入力データに整数「１」を加算するインクリメントユニット４１５。
- ー 入力データから整数「１」を減算するデクリメントユニット４１６。
- ー 入力された論理アドレスを物理アドレスに翻訳（変換）するアドレス変換マップユニット４１７。

[0105] ROMユニット１０３が直接的に読み出された場合で、ゲームマシン１２０からの入力メッセージが「READ」メッセージであり、そして、論理アドレスが、直接アクセスが可能なROM領域に対応する場合、又は、ROMユニット１０３が間接的に読み出された場合で、入力メッセージが「CHALLENGE」メッセージである場合、コントロールユニットは、以下のステップを行う。

[0106] アドレスレジスタ４１２には適当な値が設定される。直接アクセスの場合、論理アドレスは、アドレス翻訳マップ４１７によって物理アドレスへ変換され、そして、物理アドレスがアドレスレジスタ４１２内でコピーされる。長さの値は、長さレジスタ４１３内でコピーされる。

[0107] その後、アドレスレジスタの値が、バス４５２を介してROMユニット１０３へ送られ、そして、ピン１０７からの電力供給信号とピン１０８からのクロック信号も、バス４５２を介して送られる。最後に、ROMチップセレ

クト（ROM CS）が、ROMユニット103を作動するため、「1」に設定される。

[0108] その結果、ROMユニット103は、バス451を介して、特定のアドレスに配置された1バイトを戻す。このバイトは、I/Oバッファ401とデータピン110から117を介して、ゲームマシンへ送られる。次に、アドレスレジスタ412はインクリメントユニット415によってインクリメントされ、そして、長さレジスタはデクリメントユニット416によってデクリメントされる。有限状態マシン411は上記のステップを、長さレジスタが零（0）になるまで繰り返す。

[0109] データがEEPROMユニット102から読み出され、又は、書き込まれる場合、同様のステップが、バス454、453を用いて行われる。唯一の相違は、バス454がROMチップセレクト（ROM CS）を「1」に設定し、付加的なR/W信号が、読み出しのための「0」、又は、書き込みのための「1」に設定され、そして、バスが双方向であることである。

[0110] 次に、間接的なROMへのアクセス、即ち、「CHALLENGE」メッセージがメモリコントローラにより受信され、そして、秘密テーブル201からのデータが選択されてゲームマシンに送り返される場合に関連するユニットについて述べる。秘密アドレスレジスタ430は、物理アドレスをROMユニット103内に表示する32ビットのレジスタであり、以下の部品から構成される。

- 4ビットのバイトカウンタ431。
- シフトユニット422に接続されたチャレンジビット「m」432。
- 3ビットの秘密カウンタ433。
- 暗号化ヘッダ210の一部である、16ビットの署名インデックス215。かかる署名インデックス215は制御ユニット410に接続されており、ゲームマシン120によって読み出すことができる。署名インデックス215は14ビットの署名カウンタ435と二つの零ビット436とから成る。

- [0111] チャレンジが半導体メモリ 100へ送出されると、チャレンジレジスタ 421内に格納される。次に、チャレンジ値が、シフトユニット 422により、右へ1ビットだけ移動される。この処理において、チャレンジの最右側のビットが削除され、しかしながら、この最右側のビットは、秘密アドレスレジスタ 430のチャレンジビット 432へ転送される。
- [0112] 第1の加算器 441の1ビット入力が、制御ユニット 410によって「1」に設定されると、加算器 441は、バイトカウンタ 431の値を1だけインクリメントする。この時点で、秘密アドレスレジスタ 430は、ROMユニット 103内の秘密テーブル 201の一つのエレメントに対応する物理アドレスを持っている。秘密アドレスレジスタの32ビットの値は、アドレスレジスタ 412へコピーされ、そして、バス接続 452を介してROMユニット 103へ送られる。従って、1バイトが、バス接続 452を介して、ROMユニット 103から送出され、そして、I/Oバッファ 401とデータピン 110～117を介して、ゲームマシン 120へ送られる。
- [0113] しかしながら、加算器 441は、4ビットだけを取り扱うことができ、インクリメント動作の結果が4ビットよりも大きくなると、より具体的には、「1」が「0xF」に加算されると、オーバフローが起き、そして、加算器 441の1ビットの出力が「1」に設定される。
- [0114] 加算器 441の1ビット出力は、第2の加算器 442の1ビット入力と制御ユニット 410に連結されている。制御ユニット 410が第1の加算器 441でのオーバフローを検出すると、チャレンジ値がシフトユニット 422によって1ビットだけ右に移動し、そして、最右側のチャレンジビットが、再び、秘密アドレスレジスタ 430のチャレンジビット 432へ転送される。加えて、加算器 441のオーバフロービットも第2の加算器 442へ送られ、従って、それが3ビットの秘密カウンタ 433をインクリメントする。しかしながら、秘密カウンタ 433の値が「0x7」である場合は、インクリメント動作の結果が3ビットよりも大きくなるため、第2のオーバフローが生じる。この第2のオーバフローは、制御ユニット 410と第3の加算器 44

3の両方に送られる。

- [0115] 従って、第3の加算器が14ビットの署名カウンタ435の値をインクリメントし、これによって署名インデックスを1だけ上昇する。また、制御ユニットが第2の加算器442におけるオーバフローを検知すると、受信されたチャレンジに対応する秘密データが完全に送信され、そして、メモリコントローラはアイドル状態となり、ゲームマシン120からの次のメッセージを待つ。しかしながら、署名カウンタ435の値が「0x3FFF」であり、そして、第2のオーバフロービットが「1」に設定された場合には、第3のオーバフローが生じる。これは、利用可能な秘密データが使い尽くされ、そして、半導体メモリ100がゲームマシン120とはもはや結合できないことを意味する。それ故、第3のオーバフローが制御ユニット410によって検出されると、ゲームマシン120からの如何なる新たなメッセージをも拒否する。

<メモリコントローラの機能の明細>

- [0116] 図5は、メモリコントローラの機能の明細を示すブロックダイアグラムである。如何なる特定の限定をも意図するものではないが、本実施例では、図5に記載された機能は、メモリコントローラ105における有限状態マシン411内の回路として実現されている。
- [0117] 最初に、メモリコントローラ105は聴取状態にあり、ゲームマシン120からのメッセージを待つ。メッセージがデータピン110～117を介して到着すると、以下のステップが行われる。最初に、メッセージのタイプが特定される。既に説明したように、「READ」、「WRITE」、そして、「CHALLENGE」の、メッセージには三つのタイプがある。
- [0118] ステップ511では、「READ」メッセージを検出する。ステップ512では、論理アドレスと長さペイロードとが、メモリコントローラ105によりチェックされる。より詳細には、論理アドレスは、読み出し可能(readable)で、かつ、直接にアクセス可能なデータに対応するものでなければならない。例えば、アドレス「0x880000」は読み出し可能ではなく、拒否される。

同様に、アドレス「0x1009400」は読み出し可能ではあるが、データ長さが「4」であれば、アドレス「0x1009402」は読み出し可能ではないことから、そのメッセージは拒絶される。加えて、アドレス「0x0」は秘密テーブル201に対応しており、そして、直接にアクセスすることはできない。アドレスが読み出し可能でかつ直接アクセスが可能であれば、メッセージ内に設けた長さとアドレスとが、アドレスレジスタ412と長さレジスタ413にコピーされる。

[0119] ステップ521と531では、読み出されるアドレスの物理的位置、即ち、ROM103、EEPROM102、又は、メモリコントローラ105によって、三つの場合を識別する。

[0120] ROMアドレスの場合には、ステップ523において、ROMチップセレクト（ROM CS）信号が「1」に設定され、そして、アドレスがバス452を介して送られる。加えて、アドレスレジスタ412がインクリメントされ、そして、長さレジスタ413がデクリメントされる。従って、ROMユニット103は、バス接続を介して、ゲームマシン120へ送られた特定アドレスの1バイトデータを戻す。長さレジスタが「0」よりも大きい限り、ステップ523が繰り返され、長さレジスタが「0」になると、データは完全に転送され、そして、メモリコントローラは最初の聴取状態に戻る。

[0121] EEPROMアドレスの場合には、ステップ533が実行される。ステップ533は、EEPROMユニット102を読み出しモードで起動することを除いて、基本的に、ステップ523と同じである。

[0122] 論理アドレスがROMでもEEPROMでもない場合には、ステップ541において、署名インデックス215が戻される。

[0123] ステップ551では、「WRITE」メッセージが検出される。ステップ552では、メッセージのアドレスペイロードが検証され、もしも有効であれば、アドレスがアドレスレジスタ412にコピーされる。次に、ステップ553では、EEPROMチップセレクトとR/Wを「1」に設定し、そして、アドレスをバス接続454へ送出することにより、対応するEEPROMの物

理アドレスが書込みモードで起動される。同様にして、「WRITE」メッセージのデータペイロードが、バス接続453を介して、EEPROMチップ102へ転送される。

[0124] メッセージが「READ」でも「WRITE」タイプでもない場合には、「CHALLENGE」である筈である。ステップ561では、ペイロードされたデータがチャレンジレジスタ421へコピーされる。

[0125] ステップ562では、シフトユニット422により、チャレンジ値421が1ビットだけ右に移動され、そして、最右側のビットがチャレンジビット432へコピーされる。

[0126] この時点では、秘密アドレスレジスタ430は、ゲームマシン120へ転送しなければならない第1のバイトの物理アドレスを含んでいる。このデータ転送は、ステップ563において行われ、即ち、バス接続452において、そのアドレスが秘密アドレスに設定され、そして、ROMチップセレクトが「1」に設定される。従って、1バイトがROMユニット103から受信され、そして、ゲームマシンへ送出される。加えて、バイトカウンタ431が第1の加算器441によってインクリメントされる。

[0127] インクリメント期間にオーバフローが起こらなければ、ステップ563が繰り返される。本実施例では、如何なる特定の限定をも意図するものではないが、第1の加算器が4つのビットを操作し、それ故、オーバフローが生じる前に、バイトカウンタ431が16回インクリメントされ、そして、16バイトが、ROMユニット103からゲームマシン120へ転送される。例えば、秘密アドレスが初めにアドレス「0x10」を格納している場合、アドレス領域「0x10」～「0x20」内の全てのバイトが転送され、即ち、これらの16バイトは、第2の秘密値「x[2]」302に対応することとなる。

[0128] 最初のオーバフローが第1の加算器441で起こった場合、ステップ565において、秘密カウンタ433が第2の加算器442によってインクリメントされる。このインクリメント動作期間にオーバフローが起こらなければ、チャレンジ421が、シフトユニット422によって右に1ビットだけ移

動され、そして、ステップ562において、最右側ビットがチャレンジビット453へコピーされ、そして、ステップ453～566が、再び実行される。もしも第2のオーバフローが起これば、秘密データの転送は完了する。その場合、署名カウンタ435は、ステップ567において、第3の加算器443によりインクリメントされる。このインクリメント動作期間にオーバフローが起これなければ、メモリコントローラは、その初期状態へ戻る。もしも第3のオーバフローが生じれば、秘密テーブル201からのデータが利用し尽くされたことを意味し、そして、ステップ569において、メモリコントローラは永続的休止状態(perpetual idle state)に入り、ゲームマシン120からの如何なる新たなメッセージをも拒否する。

＜「CHALLENGE」メッセージ処理の例＞

[0129] 以下に、「CHALLENGE」メッセージ処理の一例を述べる。

[0130] メモリコントローラ105は、「CHALLENGE」メッセージを、2進法により表現されたデータペイロード「(11010001)」と共に受信するものとする。更には、秘密アドレスレジスタ430は、初めは32ビット値「0x00000100」を有しているものとする。より詳細には、署名インデックスは「1」であり、他の全てのレジスタは「0」であるとする。これは、秘密値「x[17]」305の物理アドレスであることに留意されたい。

[0131] 第1のチャレンジビットは「1」であり、従って、秘密アドレスレジスタは、秘密「x[18]」の物理アドレスである「0x110」に設定される。次に、秘密「x[18]」の16バイトが、クロックピン108からのクロック信号の度に、バイトカウンタ431をインクリメントすることにより、一バイト毎、ゲームマシン120へ転送される。その結果、秘密アドレスレジスタが、順次、以下のアドレス、即ち、「0x110」、「0x111」、「0x113」…「0x11F」に格納する。

[0132] この時点において、バイトカウンタがインクリメントされると、第1の加算器441においてオーバフローが生じ、そして、秘密アドレス430が、再び、「0x110」となる。従って、秘密カウンタ432がインクリメントされ

、そして、チャレンジビットは、チャレンジ421を右に移動した後に「0」に設定され、即ち、秘密アドレスは、秘密「x[19]」のアドレスである「0x120」となる。次に、秘密「x[19]」の16バイトがゲームマシンへ転送される。

[0133] 第3と第4のチャレンジビットは「0」であり、それ故、次に転送されるべき秘密値は「x[21]」と「x[23]」である。最後に、終わりの4つのチャレンジビットは、「1」、「0」、「1」、「1」であり、それ故、転送される秘密値は、「x[26]」、「x[27]」、「x[30]」、「x[32]」である。8つの秘密値が完全に転送された後に、署名レジスタがインクリメントされ、そして、秘密アドレスレジスタに格納された値は、秘密「x[33]」のアドレスである「0x00000200」となる。メモリコントローラは、最初の聴取状態へ戻り、そして、次のメッセージを待つ。

[0134] 従って、「CHALLENGE」メッセージを処理した後、八つの秘密値「x[18]」、「x[19]」、「x[21]」、「x[23]」、「x[26]」、「x[27]」、「x[30]」、「x[32]」が転送されるが、しかし、八つの秘密値「x[17]」、「x[20]」、「x[22]」、「x[24]」、「x[25]」、「x[28]」、「x[29]」、「x[31]」は転送されない。その結果、秘密値の半分だけが明かされることとなる。

[0135] 図6は、ゲームマシン120によって行われる検証プロトコル (verification protocol) の機能の明細を示すブロックダイアグラムである。如何なる特定の限定をも意図するものではないが、この実施例では、この検証プロトコルは、ゲームマシンのフラッシュメモリ142内に格納され、そして、CPU143によって実行されるソフトウェアプログラムとして実現されるものとする。

[0136] 検証プロトコルの開始では、ゲームマシン120は、そのRAMユニット141内に、図7を参照しながら詳細に述べる認証プロトコル (authentication protocol) の期間に入手された公開鍵214と署名インデックス215とを格納する。

[0137] ステップ602では、ゲームマシンは、アドレス「0xFF0000000000」と長

さ「0x4030」と共に「READ」メッセージを送付することにより、ゲームヘッダ220を要求する。ステップ603では、ゲームマシンはゲームヘッダを入手し、そして、アドレス「0x800000」と長さ「0x10」と共に「READ」メッセージを送付することにより、ツリー「t[1]」321の先頭を要求する。ステップ604では、ゲームマシンは「t[1]」を得て、そして、以下の演算を行う。

$$H1 = H(t[1] || H(\text{game header}))$$

[0138] ステップ605において、「H1」が公開鍵と比較される。ゲームマシンによって引き出されたデータが本物(authentic)であれば、演算された値「H1」は公開鍵214と等しい筈である。もしそうでなければ、ステップ606において、検証(verification)は失敗となる。

[0139] もしも「H1」が公開鍵と等しい場合、ゲームマシンは、ステップ611において、8ビットのペイロード(乱数)と共に「CHALLENGE」メッセージを送信する。半導体メモリ100の応答は8個の秘密値「x'[1]」…「x'[j]」…「x'[8]」からなり、そして、ステップ612において、ゲームマシンにより受信される。実際、これらの8つの秘密値は、16個の秘密値「x[16*index+1]」、「x[16*index+2]」…「x[16*index+16]」の中から、メモリコントローラ105により選択され、そして、それら16個の秘密の半分の選択が、図5に示すように、チャレンジ値に従って行われる。ここで、「index」は署名インデックス215である。

[0140] 加えて、変数「A」が以下のように設定される。即ち、「A = 0x00400000 + 0x100*index」であり、ここで、「index」は署名インデックス215である。即ち、「A」は、イメージテーブル202からのエレメント「y[16*index+1]」の論理アドレスである。例えば、もしも署名インデックス215が「0」であれば、「A」は「y[1]」311の論理アドレスである。署名インデックスが「1」であれば、「A」は「y[17]」315の論理アドレスであり、以下も同様である。

[0141] ステップ621～627では、16個のイメージ「y[16*index+1]」、「y[

「 $y[16 \cdot \text{index} + 2j]$ 」…「 $y[16 \cdot \text{index} + 16]$ 」から、丁度、8個のエレメント「 y' [1]」…「 y' [j]」…「 y' [8]」が、即ち、それらの半分が引き出され、従って、それらのステップは8回、繰り返される。より詳細には、

- 秘密「 x' [j]」が実際に「 $x[16 \cdot \text{index} + 2j - 1]$ 」であれば、「 y' [j]」は「 $y[16 \cdot \text{index} + 2j]$ 」として選択される。
- しかし、秘密「 x' [j]」が「 $x[16 \cdot \text{index} + 2j]$ 」であれば、「 y' [j]」は「 $y[16 \cdot \text{index} + 2j - 1]$ 」として選択される。

[0142] 例えば、署名インデックスが「1」であるとして、「 y' [1]」の選択を考える。「 x' [1]= $x[17]$ 」が受信されれば、「 $y[18]$ 」が選択される。しかしながら、「 x' [1]= $x[18]$ 」であれば、「 $y[17]$ 」が代わりに選択される。

[0143] 以下には、正しいイメージを選択するための手順が述べられる。ステップ 6 2 2 において、チャレンジの最右のビット、即ち、最小桁ビット (LSB) が、右移動によって抽出される。一方、チャレンジビットが「0」であれば、「 $x[16 \cdot \text{index} + 2j - 1]$ 」が受信されたことを意味する。その結果として、「 $y[16 \cdot \text{index} + 2j]$ 」が必ず選択される。これは、ステップ 6 2 6 において、「0x10」を変数「A」に加算し、そして、「READ」メッセージをアドレス「A」と長さ「0x10」と共に送信することにより行われる。その後、「0x10」が変数「A」に加算される。ステップ 6 2 7 では、対応するイメージ値「 $y[16 \cdot \text{index} + 2j]$ 」が半導体メモリ 1 0 0 から受信される。更に、受信された秘密「 $x[16 \cdot \text{index} + 2j - 1]$ 」のハッシュ値と、受信されたイメージ「 $y[16 \cdot \text{index} + 2j]$ 」とが、以下のようにして、変数「H1」に連結される。

$$H1 = H1 \parallel H(x' [j]) \parallel y' [j]$$

[0144] もしも秘密値とイメージの値とが本物であれば、この演算の結果は、以下と等しくなる筈である。

[0145] [数5]

$$\begin{aligned} H1 &= H1 \parallel H(x[16 \cdot \text{index} + 2j - 1]) \parallel y[16 \cdot \text{index} + 2j] = \\ &H1 \parallel y[16 \cdot \text{index} + 2j - 1] \parallel y[16 \cdot \text{index} + 2j] \end{aligned}$$

[0146] これは、「 $H(x[16 \cdot \text{index} + 2j - 1]) = y[16 \cdot \text{index} + 2j - 1]$ 」が成り立つことに

よる。

[0147] 他方、チャレンジビットが「1」であれば、「 $x[16 \cdot \text{index} + 2j]$ 」が受信され、そして、「 $[16 \cdot \text{index} + 2j - 1]$ 」が選択されなければならない。その結果として、ステップ624では、アドレス「A」と長さ「0x10」と共に、「READ」メッセージが半導体メモリ100へ送信される。そして、「0x20」が、変数「A」に加算される。以前の場合と同様に、以下の演算がステップ625において行われる。

[0148] [数6]

$$H1 = H1 \parallel y'[j] \parallel H(x'[j])$$

[0149] 再び、もしも受信した値が本物であれば、この演算の結果は、以下と等しくなる筈である。

[0150] [数7]

$$H1 = H1 \parallel y[16 \cdot \text{index} + 2j - 1] \parallel y[16 \cdot \text{index} + 2j]$$

[0151] 以上の手順を8回だけ繰り返した後、もしも全ての秘密値及びイメージ値が本物であれば、「H1」は以下と等しくなる筈である。

[0152] [数8]

$$H1 = y[16 \cdot \text{index} + 1] \parallel \dots \parallel y[16 \cdot \text{index} + 2j - 1] \parallel y[16 \cdot \text{index} + 2j] \parallel \dots \parallel y[16 \cdot \text{index} + 16]$$

[0153] 換言すれば、「H1」は、16個のイメージ値「 $y[16 \cdot \text{index} + 1]$ 」～「 $y[16 \cdot \text{index} + 16]$ 」の連結である筈である。

[0154] ステップ631において、「H1」のハッシュ値が計算される。以下の同等性が維持されることを思い出すこと。

[0155] [数9]

$$t[16384 + \text{index}] = H(y[16 \cdot \text{index} + 1] \parallel \dots \parallel y[16 \cdot \text{index} + 16])$$

結果として、もしも全ての受信された値が本物であれば、「H1」は「 $t[16384 + \text{index}]$ 」と等しくなる筈である。

[0156] ステップ632では、変数「j」が「 $0x4000 + \text{index}$ 」に、又は、十進法で「 $16384 + \text{index}$ 」に設定される。

[0157] ステップ641～646では、ツリーテーブル203の元素が選択

され、そして、半導体メモリ 100 から抽出される。これらのエレメントは、ステップ 631 で計算された値「H1」を、ツリーテーブル「t[1]」321 の先頭に関連付けることができる。この関連が有効性であれば、検証が成功したこととなるが、しかしながら、この関連が無効であることが明らかとなれば、検証は失敗となる。

[0158] この関連は、ツリーテーブル 203 におけるエレメント間の関係、即ち、「 $t[i] = H(t[2i] || t[2i+1])$ 」を利用している。最初に、「16384+index」は偶数の整数であるとする。それから、「 $t[16384+index+1]$ 」が与えられれば、ノード「 $t[8192+index/2]$ 」の値は、以下のように計算することが可能である。

[0159] [数10]

$$t[8192+index/2] = H(H1 || t[16384+index+1])$$

[0160] 同様に、もしも「16384+index」が奇数の整数であれば、もし「 $t[16384+index-1]$ 」が与えられれば、「 $t[8192+(index-1)/2]$ 」を以下のように計算することが可能である。

[0161] [数11]

$$t[8192+(index-1)/2] = H(t[16384+index-1] || H1)$$

[0162] 両方の場合において、ツリーにおいて 1 レベルだけ上に行くことが出来る。上記の手順を繰り返すことにより、終には、全ての受信した値が本物である場合には、「H1」は、ツリーテーブル 203 の先頭である「t[1]」と等しい筈であり、これにより、ステップ 631 で計算したハッシュ値とツリーの先頭との間の関連を確立することとなる。

[0163] 以下に、ツリーデータを選択し、そして、抽出するための手順について述べる。初めに、「j」は「16384+index」に設定される。ステップ 641 では、「j」のパリティがチェックされる。

[0164] もし「j」が奇数の整数である場合、ステップ 643 では、変数「A」が、「 $t[j-1]$ 」のアドレスである「 $0x800000+0x10*(j-2)$ 」に設定される。例えば、「index=1」であれば、初め、「j」は、奇数の整数「0x4001」、又は、十

進法での「16385」と等しく、そして、「A」は以下と等しい。

[0165] [数12]

$$A = 0x800000 + 0x40000 + 0x10 - 0x20 = 0x83FFF0$$

[0166] これは、確かに、「t[16384]」3 2 5のアドレスである。

[0167] それから、アドレス「A」及び長さ「0x10」と共に、「READ」メッセージが半導体メモリに送信され、半導体メモリは、ステップ6 4 4において、ツリーテーブル2 0 3の関連するエレメント「t[j-1]」を返送する。このエレメントは、変数「H2」内に格納される。

[0168] その後、以下の演算、即ち「H1 = H(H2||H1)」が行われる。「j」は奇数の整数であることから、「H2=t[j-1]」、そして、「H1=t[j]」であり、計算の結果は、「t[(j-1)/2] = H(t[j-1]||t[j])」と等しくなる。最後に、「j」は右移動により「(j-1)/2」へアップデートされる。

[0169] 同様に、もしも「j」が偶数の整数であれば、ステップ6 4 5では、「A」が、「t[j+1]」のアドレスである「0x800000+0x10*j」に設定される。例えば、もしも「index=0」であれば、初め、「j」は偶数の整数「0x4000」、又は、十進法の「16384」と等しく、そして「A」は「0x840000」と等しい。これは、確かに、「t[16385]」3 2 6のアドレスである。

[0170] それから、アドレス「A」と長さ「0x10」と共に、「READ」メッセージが送信され、そして、関連するエレメント「t[j+1]」が「H2」内に格納される。ステップ6 4 6では、演算「H1=H(H1||H2)」が行われる。「j」は偶数の整数であることから、「H2=t[j+1]」そして「H1=t[j]」であり、この計算の結果は、「t[j/2]=H(t[j]||t[j+1])」と等しくなる。最後に、「j」は右移動により「j/2」にアップデートされる。

[0171] 変数「j」は、上述した手順の各々の繰り返しにおいて、右に移動される。即ち、最終的には、「j」は、値「1」に到達する。この時点において、全ての受信した値が本物であれば、「H1」は、ツリーテーブル2 0 3の先頭である、「t[1]」3 2 1と等しい筈である。もしもそうであれば、全ての受信した値は、確かに、本物であり、そして、検証は、ステップ6 5 2において、

成功裡に終了する。そうでない場合は、検証は、ステップ653において、失敗となる。

[0172] 本実施例においては、変数「j」は、当初は、厳密には「32768」よりも小さな整数である「16384+index」に設定される。その結果、丁度14回の繰り返し、それにより、14回の右移動の後に、変数「j」は「1」になる。換言すれば、本実施例では、ステップ642～646において、丁度14個のエレメントが、ステップ631において演算された値、即ち、「t[16384+index]」と、ツリー「t[1]」の根（ルート）との間の関連を確立するため、ツリーテーブル203から抽出される。

[0173] 上述した検証手順においては、ゲームマシンは8個の秘密値と、8個のイメージと、14個のツリー値を入手し、そして、それらの秘密、イメージ、そしてツリー値は、非特許文献2に開示されたように、チャレンジの「Merkle」署名である。

[0174] 以下には、検証手順の一例が示される。

[0175] 署名インデックス215は「1」であり、そして、ゲームヘッダ220とテーブルツリー「t[1]」321の先頭は、ゲームマシン120にとって入手可能であるものとする。

[0176] 最初に、ゲームマシンが秘密値とイメージ値とを如何に入手し、そして、それらが如何に組み合わせられるについて述べる。

[0177] ゲームマシンは、8ビットのチャレンジ（乱数）を発生し、このチャレンジの値の二進法での表現は「(11010001)」とする。従って、ゲームマシンは、メモリコントローラにより、チャレンジ値に従って選択される8個の秘密値「x'[1]」…「x'[8]」を受信する。より詳細には、もしも秘密値が本物であれば、以下の関係、「x'[1]=x[18]」、「x'[2]=x[19]」、「x'[3]=x[21]」、「x'[4]=x[23]」、「x'[5]=x[26]」、「x'[6]=x[27]」、「x'[7]=x[30]」、そして「x'[8]=x[32]」が保持される。それ故、ゲームマシンは、以下の秘密イメージ、「y[17]」、「y[20]」、「y[22]」、「y[24]」、「y[25]」、「y[28]」、「y[29]」そして「y[31]」を選択する。また、受信した

秘密値のハッシュ値を計算し、そして、全てのものを以下のように連結する
:

[0178] [数13]

$$H1 = y[17] || H(x[18]) || H(x[19]) || y[20] || H(x[21]) || y[22] || H(x[23]) || y[24] || y[25] || H(x[26]) || H(x[27]) || y[28] || y[29] || H(x[30]) || y[31] || H(x[32])$$

[0179] 「 $y[i] = H(x[i])$ 」であることから、上記の表現は、以下と等しい:

[0180] [数14]

$$H1 = y[17] || y[18] || y[19] || y[20] || y[21] || y[22] || y[23] || y[24] || y[25] || y[26] || y[27] || y[28] || y[29] || y[30] || y[31] || y[32]$$

[0181] 「H1」のハッシュ値を連結し、そして、「 $t[16385] = H(y[17] || \dots || y[32])$ 」であることから、ゲームマシンは「 $H(H1) = t[16385]$ 」を得る。

[0182] 次に、「 $t[16385]$ 」と「 $t[1]$ 」との間の関連の計算が説明される。初め、変数「j」は「16385」に等しい。

ステップ1: $j = 16385$

[0183] 整数「16385」は奇数であり、それ故、「 $t[16384]$ 」が半導体メモリから読み出される。さて、ゲームマシンは「 $H(t[16384] || t[16385]) = t[8192]$ 」を計算することが出来る。

ステップ2: $j = 8192$

[0184] 「8192」は偶数であり、「 $t[8193]$ 」が半導体メモリから読み出され、そして、ゲームマシンは「 $H(t[8192] || t[8193]) = t[4096]$ 」を計算することが出来る。

ステップ3: $j = 4096$

[0185] 「4096」は偶数であり、「 $t[4097]$ 」が半導体メモリから読み出され、そして、ゲームマシンは「 $H(t[4096] || t[4097]) = t[2048]$ 」を計算することが出来る。

[0186] この手順は、「 $j = 2048$ 」、「1024」、「512」、「256」、「128」、「64」、「32」、「16」、「8」、「4」、そして、「2」に対して、繰り返される。

ステップ14: $j = 2$

[0187] 「2」は偶数であり、「t[3]」が半導体メモリから読み出され、そして、ゲームマシンが「 $H(t[2] || t[3]) = t[1]$ 」を計算する。

[0188] ゲームマシンは、ツリーテーブルt[1]の先頭を再計算することができ、そして、検証が首尾よく終了する。

<通信プロトコル>

[0189] 図7は、ゲームマシン120と半導体メモリ100との間の通信プロトコルのブロックダイアグラムである。

[0190] 図7の左側のブロックは、半導体メモリ100により実行される作動であり、他方、図の右側のブロックは、ゲームマシン120によって実行される作動である。中央のブロックは、ゲームマシンと半導体メモリとの間で交されるメッセージである。

[0191] ゲームマシンは、暗号化ヘッダ210への要求701を送信することによりプロトコルを開始する。この要求は、暗号化ヘッダのアドレス、即ち、「0x1000000」と、そのサイズ、即ち、「0x9402」又は二進表現で「37,890」バイトとを含んだ「READ」メッセージである。半導体メモリは、完全に暗号化ヘッダ210を送ることによって応答する。ステップ302では、ゲームマシンが、以下のサブステップを行うことによって、暗号化ヘッダ210を検証する。

- ー システムパラメータ211がサポートされていることを検証する。
- ー ルート証明書212を検証する。ルート証明書が、少なくとも、ゲーム公開鍵とデジタル署名とを含んでいる。即ち、ゲームマシンは、フラッシュメモリユニット142内に格納されているルート公開鍵によって、ルート証明書内に封入されたデジタル署名を検証し、もって、ゲーム公開鍵の有効性を確定する。
- ー ゲーム証明書213を検証する。ゲーム証明書は、少なくとも、公開鍵214とそのためのデジタル署名を含んでいる。即ち、ゲームマシンは、ルート証明書212からの公開鍵により、ゲーム証明書213内に封入されたデジタル署名を検証し、もって、公開鍵214の有効性を確定する。

ー 公開鍵 2 1 4 の使用がファームウェアによって禁止されていないかをチェックする。もしも、本当に、公開鍵が危殆化されていれば、それはもう使用されるべきではなく、これが、ファームウェアのアップデートが危殆化されたものとして知られた公開鍵のリストを含んでいる理由である。

[0192] 上記のステップの一又はそれ以上が失敗すると、プロトコルは中断される。全てのステップが成功裡に実行されると、公開鍵 2 1 4 は有効となる。そこから、ゲームマシン 1 2 0 は、公開鍵 2 1 4 と署名インデックス 2 1 5 とを、後の使用のため、RAM ユニット 1 4 1 内に保持する。

[0193] ステップ 7 1 1 は認証プロトコルであり、その中では、ゲームマシンと半導体メモリとが幾つかのメッセージを交換する。ゲームマシンにより行われる作動は、既に、図 6 を参照して述べられており、他方、半導体メモリにより行われる作動は、図 4 に記載したハードウェアユニットを利用し、既に、図 5 を参照して述べられている。

[0194] 一旦、認証が首尾よく終了すると、ステップ 7 1 3 では、フラッシュメモリ 1 4 2 内に格納された解読鍵 1 6 3 を使用して、ゲーム鍵 2 2 2 がゲームマシン 1 2 0 によって解読される。ゲーム鍵 2 2 2 を使用して、プログラムのエントリポイント 2 2 3 もまた、ゲームマシンによって解読される。

[0195] 次に、ステップ 7 2 1 では、第 1 のセクタ 2 4 1 とそのハッシュ値 2 2 4 とが、ゲーム鍵 2 2 2 により解読される。第 1 のセクタがファームウェアのアップデートを格納していることを思い出すこと。ステップ 7 2 2 では、ファームウェアのアップデートの有効性が、以下のようにしてチェックされる。

ー ゲームマシン 1 2 0 が、第 1 のセクタ 2 4 1 のハッシュ値を計算し、そして、それをゲームヘッダ内に格納されているハッシュ値 2 2 4 と比較する。二つのハッシュ値は等しい筈である。

ー ファームウェアのアップデートはデジタル署名を含んでいる。それ故、ゲームマシン 1 2 0 は、ルート公開鍵 1 6 2 によりデジタル署名を検証する。

- [0196] 上述したチェックの一つが失敗すると、プロトコルは中断される。ステップ 7 2 3 では、ゲームマシンは、ファームウェアのアップデートがフラッシュメモリユニット 1 4 2 内に格納されたファームウェア 1 6 1 よりも新しいものであることを検証する。もしもそうであれば、ステップ 7 2 4 において、古いファームウェアが新しいものと交換される。そうでなければ、ゲームマシンはファームウェアのアップデートを無視する。
- [0197] そこから、プログラムデータは、半導体メモリからゲームマシンへ転送が可能となる。解読プログラムのエントリポイント 2 2 3 は、プログラム開始のアドレスを格納しており、このアドレスは、ゲームプログラム領域 2 4 0 における半導体メモリの 1, 0 2 4 個のセクタの一つに対応している。即ち、ゲームマシンは、関連するセクタに対する要求 7 3 1 を、セクタデータ 7 3 2 を転送する半導体メモリへ送信する。セクタデータが解読され、それ故、ゲームマシンは、ステップ 7 3 3 において、ゲーム鍵 2 2 2 を使用して、セクタを解読する。加えて、ゲームヘッダ 2 2 0 は、解読されたセクタのハッシュ値を格納しており、この値はゲーム鍵 2 2 2 により解読される。
- [0198] 解読されたセクタデータの有効性をチェックするため、ゲームマシンはそのハッシュ値を計算し、そして、それを、ゲームヘッダ 2 2 0 からの解読されたハッシュ値と比較する。もしも二つのハッシュ値が等しければ、セクタデータは有効であり、ステップ 7 3 5 において、セクタ内に格納されたプログラムが実行される。二つのハッシュ値が異なっていれば、これはセクタデータが変更されている可能性があることを意味しており、プログラムは実行されない。
- [0199] プログラムは、幾つかのセクタに広がっていてもよい。従って、新たなデータが要求されるそれぞれの時点で、新たなセクタが転送され、そして、ステップ 7 3 1 ~ 7 3 6 がゲームマシンによって、再度、実行される。
- [0200] 従って、以下の効果が得られる。
- [0201] 半導体メモリは、その秘密値の半分しか明かさなことから、かかる秘密値を多数集めても、悪意の者は、異なるチャレンジに対し、正しい回答を発

行することは出来ない。実際、異なるチャレンジは、半導体メモリによって明かされていない少なくとも一つの秘密を知っていることを要求する。

[0202] 更に、悪意の者は、認証を成功した後は、プログラムを挿入することは出来ない。実際、そのように試みても、挿入されたプログラムのハッシュ値は、ゲームヘッダ内のセクタのハッシュ値の何れとも一致することはない。

[0203] 最後に、ゲームマシンは、認証プロトコルを実行するためのルート公開鍵を知ることだけを必要とする。結果として、如何なるゲームマシンも、共通鍵を共有することなく、如何なる半導体メモリでも認証することができる。

<<実施例 2>>

[0204] 第 2 に、暗号化エンジンを更に備えた半導体メモリと、そして、セキュリティーチップを更に備えたゲームマシンに焦点を当てた記述を示す。

[0205] 本発明の第 1 の実施例により得られるそれらの効果に加え、下記の記載による以下の効果が得られる。半導体メモリは、秘密値とイメージ値とを再生成できる暗号化エンジンを備えている。結果として、それらの値を不揮発性メモリ内に格納する必要はなく、たとえ悪意の者がメモリの内容に直接アクセスしても、その秘密値とイメージ値とは明かされることはない。更に、ゲームマシンはセキュリティーチップを備えている。それ故、認証を非作動状態にし、又は、調整するため、悪意の者がゲームマシンのファームウェアを改造しようとしても、かかる改造は、セキュリティーチップにより検出できる。

[0206] 図 8 は、ゲームシステムのハードウェアダイアグラムであり、ゲームマシン 820 と、そして、ゲームプログラムを格納するための半導体メモリ 800 を備えている。以下では、図 1 を参照して述べたゲームシステムとの違いが強調される。

[0207] 半導体メモリ 800 は、ROM ユニット 103 と、EEPROM ユニット 102 と、そして、バスに結合されたメモリコントローラ 801 を有している。図 1 を参照して述べたメモリコントローラとは異なり、このメモリコン

トローラは、プログラム可能な (programmable) ROM (PROM) と、128ビットの鍵「A」と128ビットの平文「B」とにより、「 $AES_A(B)$ 」のようなAES暗号化を計算することが出来る、AESエンジンとを備えている。

[0208] 図1を参照して上述したゲームマシンと比較して、このゲームマシン820は、更に、セキュリティーチップ821を備えている。後に、このセキュリティーチップ821の詳細については、図12を参照しながら述べる。フラッシュメモリユニット142はファームウェア161を格納しているが、しかしながら、ルート公開鍵162又は解読鍵163を格納することは要求されない。

[0209] 図9は、半導体メモリ800内における論理メモリマッピングのブロックダイアグラムである。

[0210] 図2を参照して述べた論理メモリマッピングとは異なり、本実施例では、如何なる特定の限定をも意図するものではないが、半導体メモリ800は、秘密テーブル201やイメージテーブル202を格納していない。更に、ツリーテーブル203、システムパラメータ211、ルート証明書212、ゲーム証明書213、そして、公開鍵214は、メモリコントローラ801のPROMユニット802内において物理的に配置されている。その結果、ROMユニット103は、単に、ゲームヘッダ220とゲームプログラム240とを格納している。ゲームヘッダ220とゲームプログラム240とを格納するのは、同様のゲームを格納している全て半導体メモリにとっては同様であることから、ROMユニット103は、マスクROMの技術、又は、オプションとして1回だけプログラム可能なROMの技術によって、実現することができる。

[0211] ゲームプログラム240は幾つかのセクタに亘って広がっていてもよく、しかしながら、第1のセクタ941は、データアップデートのために取って置かれる。このデータアップデートは、ファームウェア161の最新のバージョンと、セキュリティーチップ821により要求されるデータと、そして、そのデジタル署名とを含んでいる。

<メモリコントローラ>

[0212] 図10は、半導体メモリ800のメモリコントローラ801のハードウェアダイアグラムである。

[0213] 以下、本メモリコントローラと、図4を参照して既に述べたメモリコントローラとの違いを強調して説明する。

[0214] 制御ユニット1010は、更に、シフトユニット1012、チャレンジレジスタ1011、そして、インデックスレジスタ「j」1013を備えている。

[0215] 加えて、メモリコントローラ108は、更に、ツリーテーブル203と暗号化ヘッダ210とを格納したPROMユニット802を備えている。PROMユニットは、メモリコントローラ801、8ビットデータバス、及び、VCC接続と、クロック接続と、PROMチップセレクト(PROM CS)接続と、そして、32ビットアドレスバスとを含んでいる35ビットの制御バスと、結合されている。

[0216] メモリコントローラは、更に、18ビットカウンタであるカウンタレジスタ1021を備えている。カウンタは、その長さを128ビットまで伸ばすため、百十の零ビット1022によってパディングされ(padded)ている。署名カウンタ1024は、カウンタ1021の14個の最左側ビットから構成されており、他方、秘密カウンタ1023は、カウンタ1021の4個の最右側ビットから構成されている。署名インデックス215は、その長さを16ビットまで伸ばすため、零ビット1025によりパディングされた署名カウンタ1024から構成されている。16ビットの署名インデックス215は制御ユニットと結合されており、18ビットカウンタ1021は加算器1026と結合され、そして、零ビットでパディングされたカウンタはマルチプレクサ1033に結合されている。

[0217] 加算器1026は二つの入力を有しており、一つの8ビット入力カウンタ1021と、そして、その第2の1ビット入力を介して、制御ユニット410と結合されている。第2の1ビット入力制御ユニット1010により

「1」に設定されると、カウンタ1021がインクリメントされる。このインクリメント動作中にオーバフローが生じた場合、1ビットのオーバフロー信号がカウンタユニット1010へ転送される。オーバフローは、カウンタ1021がその最大値、即ち、「0x3FFFF」、又は、十進法で表すと「262143」に達した時に生じることから、制御ユニット1010は、永続的休止状態に入り、そして、その後に来るメッセージを拒否する。

[0218] 如何なる特定の限定をも意図するものではないが、本実施例では、ハッシュ関数「H」は、AESと排他的OR動作により達成されるものとする。これに替えて、異なるブロック暗号器とすること、又は、「MD5」や「SHA-1」など、異なるハッシュ関数とすることもできる。

[0219] 即ち、メモリコントローラは、更に、AESエンジン803と排他的ORユニット1032を備えている。AESエンジンは、右側の平文入力と、頂上の鍵入力との、二つの128ビット入力と、そして、左側の暗号文出力である、128ビットの出力とを有している。AESエンジンは、マルチプレクサ1033、1034、1035になる構成によって、モード「0」とモード「1」と呼ばれる二つのモードの動作をサポートしている。

[0220] モード「1」では、AESエンジンは、128ビットの秘密のシード (seed) 「s」を鍵入力とし、そして、平文入力をカウンタ1021の入力とする動作「 $x = AES_s(\text{counter})$ 」を行う。モード「1」を起動するため、それらの「1」とラベルが貼られた入力を選択するよう、制御ユニットがマルチプレクサ1033、1034、1035を設定する。より詳細には、マルチプレクサ1033がモード「1」に設定されると、カウンタ1021が、AESエンジンの平文入力に結合される。マルチプレクサ1034がモード「1」に設定されると、シード1041が、AESエンジンの鍵入力に結合される。そして、マルチプレクサ1035がモード「1」に設定されると、排他的ORユニット1032がオール零レジスタ (all-zero register) 1043に結合される。その結果、「 $x(+0) = x$ 」であることから、出力レジスタ1044が、動作「 $AES_s(\text{counter})$ 」の結果を格納することとなる。

[0221] モード「0」では、AESエンジンは、平文入力初期ベクトル「IV」1042であり、そして、鍵入力が、出力レジスタ1044に格納された以前のAES動作の結果、「x」である、動作「 $y = AES_x(IV) (+) IV$ 」を行う。モード「0」を起動するため、マルチプレクサ1033、1034、1035は、制御ユニット1010によって、それらの「0」とラベルが貼られた入力を選択するよう、適切に構成される。マルチプレクサ1033がモード「0」に設定されると、「IV」1042がAESエンジンの平文入力に結合される。マルチプレクサ1034がモード「0」に設定されると、出力レジスタ1042がAESエンジンの鍵入力に結合される。そして、マルチプレクサ1035がモード「0」に設定されると、排他的ORユニット1032が「IV」1042と結合される。その結果、出力レジスタ1044は、動作「 $AES_x(IV) (+) IV$ 」の結果を格納することとなる。

[0222] AESエンジン803の二つのモードにより、メモリコントローラは乱数「x」と共に、それらのイメージ「 $y=H(x)$ 」を計算することが出来る。「x」値は、秘密値の役割を果たし、他方、「y」値は、認証プロトコルにおけるイメージ値の役割を果たすことから、不揮発性メモリ内に秘密テーブルとイメージテーブルを格納するという要求を排除することができる。加えて、本実施例の秘密鍵（private key）は、それからAESエンジンのモード「1」を利用して秘密値が取り出され、シード1041となっている。

<メモリコントローラの機能の詳細>

[0223] 図11は、メモリコントローラ801の機能の詳細を示すブロックダイアグラムである。

[0224] メモリコントローラ801は、既に図5を参照して述べたメモリコントローラ105と同様の方法により、ゲームマシン820により送り出された「READ」メッセージと「WRITE」メッセージとを処理する。しかしながら、「READ」メッセージの場合、半導体メモリ801がPROMユニット802を更に備えていることから、以下のステップが行われる。ステップ1143では、PROMユニット102がその「PROM CS」入力を介して起動され、そして物

理PROMアドレスが制御ユニット1010によって送出される。従って、PROMユニットは、1バイトのデータを、当該バイトを通信ピン110～117を介してゲームマシン820へ送信する制御ユニットに対し、送出する。加えて、アドレスレジスタ412がインクリメントされ、そして、長さレジスタがデクリメントされる。長さレジスタ内に格納された値が零でない限りは、ステップ1143が繰り返される。

[0225] 「CHALLENGE」メッセージが処理される方法も、また、異なっている。メモリコントローラは、以下の手順を8回繰り返し、各繰り返しでは、チャレンジ1011に応じて、一つの秘密値「 $x' [j]$ 」及び一つのイメージ値「 $y' [j+1]$ 」を、又は、一つのイメージ値「 $y' [j]$ 」及び一つの秘密値「 $x' [j+1]$ 」を発生する。

[0226] ステップ1164では、チャレンジ1011は、シフトユニット1012により、1ビットだけ右に移動される。この処理では、チャレンジの最右側ビット(LSB)がチェックされる。

[0227] チャレンジビットが「0」であれば、メモリコントローラは、ステップ1166において、「 $x' [j]$ 」と「 $y' [j+1]$ 」を計算する。より詳細には、「 $x' [j]=AES_s(counter)$ 」、即ち、制御ユニット1010が、AESエンジン803のモード「1」を起動するように、マルチプレクサ1033、1034、1035を設定する。AESエンジンが設定された後、出力レジスタ1044は、ゲームマシン820へ送信された「 $x' [j]$ 」を格納する。次に、カウンタ1021が、加算器1026を使用することによりインクリメントされ、そして、モード「1」が再び起動される。その結果、出力レジスタが「 $x' [j+1]$ 」を格納する。「 $y' [j+1]$ 」を計算するため、モード「0」が起動される。AESエンジンが設定された後、出力レジスタ1044は、ゲームマシンへ送信された「 $y' [j+1]=AES_{x[j+1]}(IV)(+)IV$ 」を格納する。秘密「 $x' [j+1]$ 」は破棄されていることから、明らかになっていないことに留意のこと。

[0228] 同様に、チャレンジビットが「1」の場合、メモリコントローラは、モード「1」を起動することにより「 $y' [j]$ 」と「 $x' [j+1]$ 」を計算し、モード「0

」で「y' [j]」を送信し、それから、モード「0」を起動して「x' [j+1]」を送信する。同様に、秘密「x' [j]」は破棄される。

<「CHALLENGE」メッセージの処理の一例>

[0229] 以下、「CHALLENGE」メッセージの処理の一例を述べる。

[0230] メモリコントローラ 801 は、二進法表現で「(11010001)」と表現されるデータペイロードと共に「CHALLENGE」メッセージを受信するものとする。更に、初期において、カウンタ 1021 は 18 ビット値「0x00010」を含んでいるものとする。より詳細には、署名インデックスは「1」であり、そして、秘密カウンタは「0」である。

[0231] 第 1 のチャレンジビットは「1」であることから、AES エンジン はモード「1」で起動される。従って、出力レジスタ 1044 は「x' [1]=AES_s(0x10)」を格納する。次に、モード「0」が起動される、そして、出力レジスタが、ゲームマシン 820 へ送出された「y' [1]=AES_{x'}[1](IV)(+)IV」を格納する。その後、カウンタ 1021 が「0x00011」にインクリメントされ、モード「1」が起動され、よって、出力レジスタ 1044 は、ゲームマシン 820 へ送出された「x' [2]=AES_s(0x11)」を格納する。

[0232] 第 2 のチャレンジビットは「0」であり、それ故、AES エンジン は「x' [3]=AES_s(0x12)」、「x' [4]=AES_s(0x13)」、そして「y' [4]=AES_{x'}[4](IV)(+)IV」を計算する。「x' [3]」と「y' [4]」が、ゲームマシン 820 へ送出される。同様に、第 3 及び第 4 のチャレンジビットは「0」であり、それ故、「x' [5]」、「y' [6]」、「x' [7]」、そして、「y' [8]」が発送される。この時点で、カウンタ値 1021 は「0x18」である。

[0233] 次のチャレンジビットは「1」、「0」、「1」、「1」であることから、メモリコントローラ 801 は、「y' [9]」、「x' [10]」、「x' [11]」、「y' [12]」、「y' [13]」、「x' [14]」、「y' [15]」、「x' [16]」を計算して発送する。全ての動作が完了した後は、カウンタ値 1021 は「0x20」である。

<セキュリティーチップ>

[0234] 図12は、ゲームマシン820のセキュリティーチップのハードウェアダイアグラムである。

[0235] セキュリティーチップ821は、スマートカードマイクロコントローラ (smartcard microcontroller) として、又は、TPMチップ (trusted computing module) である、一般的には、耐タンパー性を持つチップ (tamper-resistant chip) として実現することができる。本実施例では、セキュリティーチップは、三つの異なるメモリユニット、即ち、ROMユニット1210、EEPROMユニット1220、そして、RAMユニット1230を備えている。

[0236] ROMユニット1210は、「OS」1211でもよいプログラムと、例えば、AES暗号化又はRSA動作のような低レベルの安全機能性をインターフェイス接続する高レベル機能1212と、図13を参照しながら詳細に述べられる認証プロトコル1213のプログラムと、そして、パターンマッチングプログラム1214とを格納している。これらのプログラムは、全ての商品化されたゲームマシンにとって同じであってもよいことから、ROMユニット1210はマスクROM技術によって実現することができる。

[0237] 他方、EEPROMユニット1220は、もし必要であれば修正され、カスタマイズされ、又は、アップデートが可能なデータを格納する。より詳細には、ファームハッシュ1221、ルート公開鍵1222、解読鍵1223、パターンファイル1224、ブラックリスト1225、そして、パッチ1226を格納する。ファームハッシュ1221は、ゲームマシン820のフラッシュメモリユニット142内に格納されたファームウェア161のハッシュ値である。ルート公開鍵1222は、半導体メモリ800がゲームマシン820と結合した場合に、ルート証明書212を有効にするために使用された公開鍵である。解読鍵1223は、半導体メモリ800がゲームマシン820と結合した場合に、ゲーム鍵222を解読するために使用可能な鍵である。パターンファイル1224は、部分的に悪意のあるコードやプログラムバグのリストを格納する。ブラックリスト1225は、危殆化されたこと

が知られた公開鍵のリストである。最後に、パッチ 1 2 2 6 は、ゲームにおける既知のバグをパッチするための部分的なコードのリストからなる。

[0238] RAM ユニット 1 2 3 0 は、プログラムの実行中に発生したテンポラリデータや変数 1 2 3 1 を格納する。

[0239] セキュリティーチップ 8 2 1 は、また、三つの演算ユニット、即ち、CPU 1 2 4 1、AES エンジン 1 2 4 2、RSA エンジン 1 2 4 3 と、そして、乱数発生器 (RNG) 1 2 4 5 を含んでいる。

[0240] 中央演算ユニット (CPU) 1 2 4 1 はマイクロプロセッサであり、例えば、プログラムのよう、ROM ユニット 1 2 1 0 内に格納されたものを、中間結果のための RAM ユニット 1 2 3 0 と、そして、カスタマイズされたパラメータを処理し又は長時間結果を格納するための EEPROM ユニットとを利用して実行する。

[0241] AES エンジン 1 2 4 2 は、暗号化に特化されたハードウェアコプロセッサである。コプロセッサは、ビットストリング「A」を鍵としてビットストリング「B」の暗号化を参照する AES 演算「 $AES_A(B)$ 」を、高速かつ低電力消費で、実行することができる。CPU 1 2 4 1 と接続して使用した場合、AES エンジン 1 2 4 3 は、データのハッシュを計算する (hashing data) などのより高度な演算は実現可能となる。例えば、1 2 8 ビットの初期ベクトル「IV」と 2 5 6 ビットのメッセージ「 $x=x[1]||x[2]$ 」とが与えられると、1 2 8 ビットのハッシュ値「 $y=H(x)$ 」が、以下の二つのステップにおいて計算され、即ち、「 $y'=AES_{x[1]}(IV)(+)IV$ 」と「 $y=AES_{x[2]}(y')(+)y'$ 」、ここでは、AES 暗号化は、AES エンジン 1 2 4 2 により行われ、排他的 OR「(+)」は、CPU 1 2 4 1 により行われ、そして、中間結果「 y' 」は RAM 1 2 3 0 内に格納される。

[0242] RSA エンジン 1 2 4 3 は、RSA 演算に特化したハードウェアコプロセッサである。特に、RSA エンジン 1 2 4 3 は、EEPROM 1 2 2 0 内に格納されたルート公開鍵 1 2 2 2 を使って、ルート証明書 2 1 2、ゲーム証明書 2 1 3、又は、アップデートデータ 9 4 1 内に封入されている RSA 署

名の検証が可能である。

[0243] 乱数発生器 1 2 4 5 は、チャレンジとして使用できる乱数を発生する。

[0244] 最後に、セキュリティーチップ 8 2 1 内のユニットは内部バス 1 2 5 1 に結合されており、セキュリティーチップ 8 2 1 は、バス 1 5 1 により、ゲームマシン 8 2 0 の他の電子コンポーネントに結合されている。他のコンポーネントとの通信は、入／出力(I/O)インターフェイス 1 2 4 4 により管理されている。

<検証プロトコル>

[0245] 図 1 3 は、ゲームマシン 8 2 0 によって行われる検証プロトコル (verification protocol) の機能詳細を表すブロックダイアグラムである。如何なる特定の限定をも意図するものではないが、この実施例では、検証プロトコルは、セキュリティーチップ 8 2 1 の ROM ユニット 1 2 1 0 内に格納されたソフトウェアプログラムとして実現され、AES エンジン 1 2 4 2 と RSA エンジン 1 2 4 3 を使って、CPU 1 2 4 1 により実行されるものとする。

[0246] 以下に述べる検証プロトコルは、既に図 1 を参照して述べたものとは、幾つかの点で異なっている。相違点の一つは、どのようにして、「CHALLENGE」メッセージを半導体メモリ 8 0 0 に送信した後に受信されたデータが処理されるかである。

[0247] ステップ 1 3 1 2 では、ゲームマシンが、半導体メモリからの 1 6 個の値、「s[1]」…「s[16]」を受信する。最初、インデックス「j」は「1」に、そして、ビットストリング「H1」は、空のビットストリングに設定されている。次に、ステップ 1 3 2 1 ~ 1 3 2 6 が 8 回繰り返される。

[0248] ステップ 1 3 2 2 では、セキュリティーチップ 8 2 1 の CPU 1 2 4 1 によって行われる右移動動作を利用して、チャレンジの最右側のビットが抽出される。このチャレンジビット (LSB) が「0」であれば、受信された「s[j]」は「x' [j]」である筈であり、そして、「s[j+1]」は「y' [j+1]」である筈である。それ故、ステップ 1 3 2 6 では、AES エンジン 1 2 4 2 と CPU 1 2 4 1 とを使って、ハッシュ値「y' [j]」が「y' [j]=AES_{s[j]}(IV)(+)IV」と

して計算される。本実施例で使われるハッシュ関数「H」の定義によれば、以下の等式が満足されることに留意のこと。

[0249] [数15]

$$y'[j] = AES_{s[j]}(IV) \oplus IV = H(x'[j])$$

[0250] その後、ビットストリング「y'[j]」と「s[j+1]」とが「H1」に添えられ、そして、カウンタ「j」が二つインクリメントされる。

[0251] 同様にして、チャレンジビットが「1」であれば、受信された「s[j]」は「y'[j]」であり、他方、「s[j+1]」であれば「x'[j+1]」である。即ち、ステップ1236では、「y'[j+1]」が「 $AES_{s[j+1]}(IV) (+) IV = H(x'[j+1])$ 」として計算される。そして、ビットストリング「s[j]」と「y'[j+1]」が「H1」に添えられ、そして、カウンタ「j」が二つインクリメントされる。

[0252] 上述のステップを8回繰り返した後、そして、受信されたデータ「s[1], ..., s[16]」が本物であるとする、ビットストリング「H1」は「y'[1]||y'[2]||...||y'[16]」と等しい筈である。ステップ1341では、AESエンジン1242を使用して、「H1」のハッシュ値が計算される。ステップ641、642、1343~1346は、PROM802内に配置されたツリーテーブル203からツリーデータを抽出することを狙ったものである。それらのステップは、ツリーテーブル203の論理アドレスが「0x0」から開始することから、アドレス「A」がステップ1343においては「(j-2)*0x10」として、そして、ステップ1343においては「j*0x10」として、簡単に、CPU1241により計算されるということを除き、基本的には、既に図6を参照して述べたステップと同様である。ステップ1344では、ハッシュ値「H(H2||H1)」が、二つのステップにより、CPU1241とAESエンジン1242とを使って、以下のように計算される、最初に、「H1' = $AES_{H2}(IV) (+) IV$ 」が計算され、そして、「H1 = $AES_{H1}(H1') (+) H1'$ 」が計算される。

[0253] ステップ641、642、1343、及び1346を14回繰り返した後、そして、全ての受信されたデータが本物であるとする、セキュリティチップは、ツリーテーブル「t[1]」321の先頭を再計算することができ、

そして検証は成功となる。もしも再計算されたデータがツリーテーブルの先頭と異なっていれば、検証は失敗となる。

<通信プロトコル>

[0254] 図14は、ゲームマシン120と半導体メモリ100との間の通信プロトコルのブロックダイアグラムである。

[0255] 図14の左側のブロックは、半導体メモリ800により実行される作動であり、他方、右側のブロックは、ゲームマシン820により実行される作動である。中央のブロックは、ゲームマシンと半導体メモリとの間で交わされるメッセージである。

[0256] 先に図7を参照して述べた通信プロトコルは、本実施例の通信プロトコルとは、幾つかの点で異なっている。これらの相異は、半導体メモリ800がAESエンジン1242を備えていること、そして、ゲームマシン820がセキュリティーチップ821を備えているという事実に起因するものである。これらのユニットのおかげで、通信プロトコルは、以下のような方法により強化できる。

[0257] ゲームマシン820がスイッチ139により給電されると、ファームウェア161がRAMユニット141にロードされ、そして、CPU143によって実行される前に、セキュリティーチップ821が、ステップ1401と1402においてその完全性を検証する。より詳細には、ステップ1401において、ファームウェアのハッシュ値が、AESエンジン1242とセキュリティーチップ1241のCPUを使って計算される。このハッシュ値は、セキュリティーチップ821のEEPROMユニット1220内に格納されたファームウェア1221のハッシュ値と比較される。もしも、それらが一致しなければ、ファームウェアは非合法に修正されていることを意味し、この場合、ゲームマシンはスタートとしない。もしもファームウェアがその完全性のチェックを通過すれば、RAM141内にロードされ、そして、CPU143によって実行される。

[0258] 次に、暗号化ヘッダ210が、半導体メモリ800から入手される。暗号

化ヘッダの有効性は、セキュリティーチップにより、以下のようにしてチェックされる。

- ー システムパラメータ 2 1 1 がサポートされていることを、セキュリティーチップが検証する。
- ー ルート公開鍵 1 2 2 2 により、ルート証明書 2 1 2 を検証する。証明書内に封入された署名が R S A エンジン 1 2 4 3 を使用して検証され、それにより、ゲーム公開鍵 2 1 4 の有効性を断定する。
- ー ルート証明書 2 1 2 内に封入されたゲーム公開鍵によってゲーム証明書 2 1 3 を検証する。ゲーム証明書内に封入されている署名は、R S A エンジン 1 2 4 3 を使用して検証され、これによって、ゲーム公開鍵 2 1 4 の有効性を断定する。
- ー 公開鍵 2 1 4 の使用が、危殆化された公開鍵のブラックリスト 1 2 2 5 内に存在しないことをチェックする。

[0259] 暗号化ヘッダ 2 1 0 が有効であれば、先に図 1 0、1 1、及び、1 2 を参照して述べた認証プロトコルが、ステップ 1 4 1 1 で実行される。成功裡の結果の後、ステップ 1 4 1 3 では、A E S エンジン 1 2 4 2 により、解読鍵 1 2 2 3 を使ってゲーム鍵 2 2 2 が解読され、そして、A E S エンジン 1 2 4 2 によって、解読されたゲーム鍵 2 2 2 を使って、プログラムのエントリポイント 2 2 3 が解読される。

[0260] ゲームプログラムの最初のセクタ 9 4 1 が、データのアップデートのために保存されていることを思い出すこと。ステップ 1 4 2 1 では、第 1 のセクタ 9 4 1 とそのハッシュ値 2 2 4 が、A E S エンジン 1 2 4 2 により、ゲーム鍵 2 2 2 を使って解読される。ステップ 1 4 2 2 では、第 1 のセクタ 9 4 1 の完全性が、セキュリティーチップ 8 2 1 によって、以下のように検証される。

- ー 第 1 のセクタのハッシュ値が A E S エンジン 1 2 4 2 を使って計算され、そして、解読されたハッシュ値と比較される。等しくない場合には、データのアップデートが非合法的に修正されている。

ー 第1のセクタは、RSAエンジン1243と、ルート公開鍵1222とを使って検証したデジタル署名とを含んでいる。デジタル署名の有効性が、データアップデート914の完全性を保障する。

[0261] しかしながら、ファームウェア161及び／又はセキュリティーチップ821内に格納されたデータが、既にアップデートされていることもある。この場合、データアップデート941は廃棄される。アップデートが要求されると、ステップ1424において実行され、即ち、ファームウェアがより新たなバージョンに取り替えられ、そのハッシュ値1221がEEPROM1220内に格納され、そして、アップデートされたパターンファイル1224又はアップデートされたブラックリスト1225など、追加データがEEPROM1220内にコピーされる。

[0262] 次に、メインプログラムが半導体メモリから転送される。ゲームプログラムは暗号化されていることから、ステップ1433では、AESエンジン1242により、ゲーム鍵222を使ってゲームプログラムが解読される。同様に、ゲームヘッダ220内に格納したセクタのハッシュ値が解読される。転送されたセクタからのデータを有効にするために、ステップ1434においては、以下のタスクが、セキュリティーチップ821によって行われる。

ー セクタのハッシュ値が、AESエンジンを使用することによって計算される。計算されたハッシュ値が、ゲームヘッダ220内に格納されたハッシュ値と異なっていれば、セクタデータは非合法的に修正されている。

ー セクタデータは、デジタル署名を含んでおり、このデジタル署名が、ルート公開鍵1222とRSAエンジン1243を用いて検証される。検証が成功の場合には、セクタの完全性が保障される。

ー 解読されたセクタはパターンファイル1224の要素と比較される。もしも一致しているとされれば、セクタデータは、バグ又は悪意のプログラムを含んでいる。この場合、セキュリティーチップ821は、パッチ1126内の訂正された部分的なプログラムを探す。パッチ (replacement) も見つか

らなければ、プログラムの実行が停止される。

[0263] もしもセクタの完全性が断定されたとすれば、ステップ735において、セクタ内に格納されたプログラムがCPU143により実行される。半導体メモリ800からもっと多くのデータが必要であれば、上述したステップが繰り返される。

[0264] さて、好適な実施例によって得られる効果について、以下に、簡単に述べる。

[0265] 秘密鍵（private key）を知らなくとも、チャレンジの公開鍵やデジタル署名を利用し、コンピュータマシンは、原品である格納媒体を偽造品から区別することができる。更に、公開鍵を利用して、デジタルデータの有効性をチェックできることから、非合法プログラムを挿入することはできない。その結果、本発明は、デジタルデータの認証されていないコピーの利用を防止する。

[0266] 本発明になる幾つかの実施例について示しかつ述べたが、開示された実施例は、本発明の範囲から逸脱することなく変更や修正がされうるものであることが理解されよう。それ故、ここに示され、かつ、述べられた詳細によって縛られることはなく、しかしながら、添えられた請求項の範囲内に入ることとなる、かかる変更や修正の全てがカバーされる。

符号の説明

[0267] 100…半導体メモリ、不揮発性のメモリユニット…102、読み出し専用不揮発性のメモリユニット…103、104…バス、メモリコントローラ…105、ピン…106～117、120…ゲームマシン、122…方向性パッド、123…ボタン、124…液晶スクリーン、125…スピーカ、138…空洞部、139…オン／オフスイッチ、126～137…コンタクト、141…RAM、142…フラッシュメモリユニット、143…CPU、入出力インターフェイスユニット…144、145…ビデオコントローラ、146…サウンドコントローラ、147…入力制御ユニット148…電池、161…ファームウェア、162…ルート公開鍵、163…解読鍵。

請求の範囲

- [請求項1] コンピュータマシンと結合されるための半導体メモリであって、
第1のデータ、第1の鍵、そして、第2の鍵を格納し、前記第1の鍵は、前記第1のデータと前記第2の鍵とから導き出された所定の値であるところの不揮発性メモリユニットと、そして、
コンピュータマシンからのチャレンジを受信し、そして、コンピュータマシンへ第2のデータを転送し、第2のデータが、前記第2の鍵と前記チャレンジとから導き出されるところのメモリコントローラとを備えていることを特徴とする半導体メモリ。
- [請求項2] 請求項1に記載した半導体メモリにおいて、前記第2の鍵は、複数の秘密値を有しており、そして、前記第2のデータは、前記秘密値の選択を含んでいることを特徴とする半導体メモリ。
- [請求項3] 請求項2に記載した半導体メモリにおいて、前記メモリコントローラは、更に、
秘密値の範囲を選択するための第1の選択ユニットと、
前記範囲内の前記秘密値の半分を選択するための第2の選択ユニットとを備えており、
前記第2のデータは、前記第2の選択ユニットにより選択された前記秘密値を含んでいることを特徴とする半導体メモリ。
- [請求項4] 請求項3に記載した半導体メモリにおいて、前記不揮発性メモリユニットは、更に、
前記秘密値の所定のハッシュ値である複数のイメージ値と、
第1の部分と第2の部分とを備えた複数のツリー値とを備えており、
そこでは、
前記第1の部分でのツリー値は、複数のイメージ値の連結の所定のハッシュ値であり、そして、
前記第2の部分でのツリー値は、複数のツリー値の連結の所定のハッシュ値であることを特徴とする半導体メモリ。

- [請求項5] 請求項4に記載した半導体メモリにおいて、前記第2のデータは、更に、イメージ値の選択と、そして、ツリー値の選択とを含んでいることを特徴とする半導体メモリ。
- [請求項6] 請求項5に記載した半導体メモリにおいて、前記第1の鍵はツリー値の連結のハッシュ値であり、そして、前記第2のハッシュ値は、前記第1のデータから導き出された所定のハッシュ値であることを特徴とする半導体メモリ。
- [請求項7] 請求項1に記載した半導体メモリであって、更に、ハッシュ機能ユニットを備えていることを特徴とする半導体メモリ。
- [請求項8] 請求項7に記載した半導体メモリであって、更に、
前記ハッシュ機能ユニットにより前記第2の鍵を処理し、もって、
第1の秘密値を発生する秘密値発生ユニットと、そして、
前記ハッシュ機能ユニットにより前記第2の鍵を処理して、第2の秘密値を発生し、前記第2の秘密値を前記ハッシュ機能ユニットにより処理してイメージ値を発生するイメージ発生ユニットとを備えたことを特徴とする半導体メモリ。
- [請求項9] 請求項8に記載した半導体メモリであって、更に、前記チャレンジの少なくとも一のビットに対応して、前記秘密値発生器か、前記イメージ発生器かの、何れか一方を作動する作動ユニットを備えていることを特徴とする半導体メモリ。
- [請求項10] 請求項9に記載した半導体メモリであって、更に、前記作動ユニットを繰り返して、一連の秘密値とイメージ値を発生するループ制御ユニットを備えており、前記第2のデータが、前記一連の秘密値とイメージ値を備えていることを特徴とする半導体メモリ。
- [請求項11] 半導体メモリと結合するためのコンピュータマシンであって、
CPUと、
第1の鍵を格納している不揮発性メモリユニットと、
前記半導体メモリからのデータを受信するための入出力インターフ

ェイスとを備えているものにおいて、前記半導体メモリからのデータは、

第2の鍵と、

第1のデータと、そして、

第2のデータとを含んでおり、そして、前記コンピュータマシンは、更に、

前記第2の鍵を検証するあめの署名検証ユニットと、

チャレンジを発生するための乱数発生器と、

前記第1と第2のデータを処理するためのハッシュ関数ユニットと、そして、

前記ハッシュ関数ユニットによって処理されたデータと第2の鍵を比較するための比較ユニットを備えており、もって、第1と第2のデータの完全性を確立することを特徴とするコンピュータマシン。

[請求項12] 請求項11に記載したコンピュータマシンにおいて、前記半導体メモリからのデータは、前記署名検証ユニットによって検証されるべき少なくとも一の証明書を含んでおり、そして、

少なくとも一つ以上の証明書は、前記第2の鍵のデジタル署名を含んでおり、そして、

少なくとも一つ以上の証明書は、前記第1の鍵に検証されるデジタル署名を含んでいることを特徴とするコンピュータマシン。

[請求項13] 請求項11に記載したコンピュータマシンであって、更に、

前記不揮発性メモリ内に格納された複数の危殆化された鍵と、そして、

前記危殆化された複数の鍵を前記第2の鍵と比較し、もって、前記第2の鍵が危殆化されていないことを確認する比較ユニットとを備えていることを特徴とするコンピュータマシン。

[請求項14] 請求項13に記載したコンピュータマシンにおいて、前記不揮発性メモリは、更に、

不安全なデータの複数のパターンと、
不安全なデータのための複数のパッチと、
半導体メモリからのデータを前記不安全なデータのパターンと比較し、もって、当該データが安全に処理され得たことを確かめる第3の比較ユニットと、そして、

不安全なデータをパッチで置き換える置き換えユニットとを備えていることを特徴とするコンピュータマシン。

[請求項15] 請求項11に記載したコンピュータマシンであって、更に、セキュリティチップを備えており、前記不揮発性メモリ、署名検証ユニット、乱数発生器、そして、ハッシュ関数ユニットが、前記セキュリティチップのハードウェアユニットであることを特徴とするコンピュータマシン。

[請求項16] データ処理方法であって、
半導体メモリからの第1の鍵を受信するステップと、
前記半導体メモリへチャレンジを送信するステップと、
前記半導体メモリから第1のデータを受信するステップと、
前記チャレンジと前記第1の鍵により前記第1のデータの完全性をチェックするステップと、
前記半導体メモリから第2のデータを受信するステップと、
前記第1の鍵により前記第2のデータの完全性をチェックするステップと、そして、
前記第2のデータを処理するステップとを備えたことを特徴とするデータ処理方法。

[請求項17] 請求項16に記載したデータ処理方法において、前記第1のデータを受信するステップは、
複数の秘密値を受信するステップと、
複数のイメージ値を受信するステップと、そして、
複数のツリー値を受信するステップとを備えていることを特徴とす

るデータ処理方法。

[請求項18] 請求項17に記載したデータ処理方法において、前記第1のデータの完全性をチェックするステップは、

前記受信した秘密値をハッシュ関数により処理し、もって、複数のイメージ値を算出するステップと、

前記受信したイメージ値と前記計算したイメージ値とをハッシュ関数により処理し、もって、第1のツリー値を算出するステップと、そして、

前記第1の計算されたツリー値と前記受信されたツリー値とをハッシュ関数により処理し、もって、第2のツリー値を算出するステップとを備えていることを特徴とするデータ処理方法。

[請求項19] 請求項18に記載したデータ処理方法において、前記第2のデータの完全性をチェックするステップは、

前記第2のデータをハッシュ関数により処理し、もって、第1のハッシュ値を算出するステップと、

前記第1のデータと第2の算出されたツリー値とをハッシュ関数により処理し、もって、第2のハッシュ値を算出するステップと、そして、

前記第2のハッシュ値を前記第1の鍵と比較するステップとを備えていることを特徴とするデータ処理方法。

[請求項20] 請求項16に記載したデータ処理方法であって、更に、

少なくとも一の証明書を受信するステップと、そして、

少なくとも一の証明書を第2の鍵により検証し、もって、前記第1の鍵を検証するステップとを備えていることを特徴とするデータ処理方法。

[請求項21] 請求項20に記載したデータ処理方法であって、更に、前記第1の鍵を、危殆化された鍵のリストと比較するステップを備えていることを特徴とするデータ処理方法。

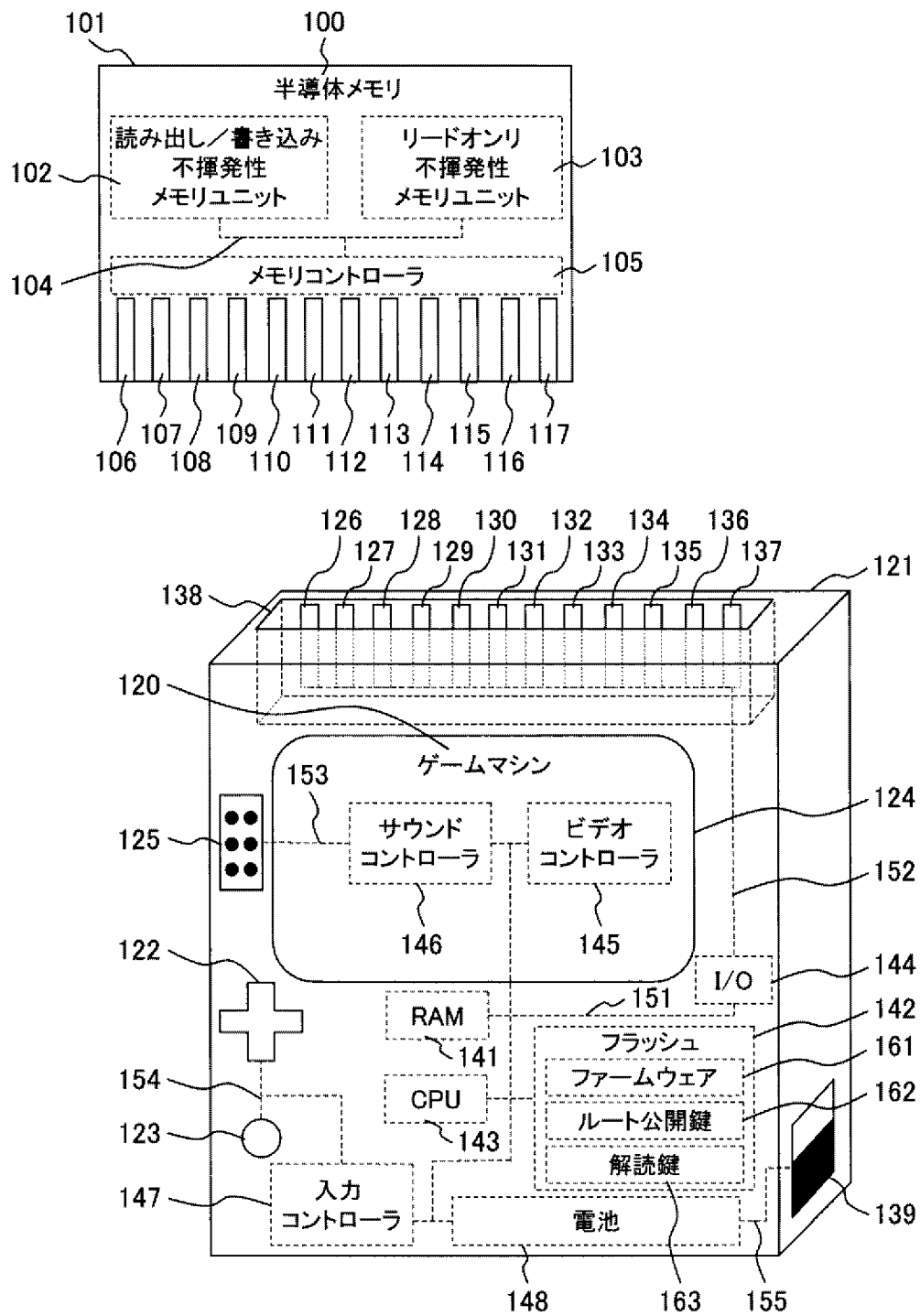
- [請求項22] 請求項20に記載したデータ処理方法において、前記第2のデータの完全性をチェックするステップは、更に、
- 前記第2のデータを不安全なデジタルデータのパターンと比較するステップと、そして、
- 不安全なデータをパッチで置き換えるステップを備えていることを特徴とするデータ処理方法。
- [請求項23] 請求項20に記載したデータ処理方法であって、更に、
- ファームウェアプログラムをハッシュ関数で処理し、もって、ハッシュ値を得るステップと、
- 前記ハッシュ値を所定の値と比較するステップと、そして、
- 前記ファームウェアプログラムを実行するステップを備えていることを特徴とするデータ処理方法。
- [請求項24] 請求項23に記載したデータ処理方法において、前記第2のデータを処理するステップは、
- ファームウェアのアップデートとそのデジタル署名とを抽出するステップと、
- 前記デジタル署名を第2の鍵により検証するステップと、そして、
- 前記ファームウェアプログラムを、前記ファームウェアのアップデートで取り換えるステップを備えていることを特徴とするデータ処理方法。
- [請求項25] データ処理方法であって、
- 第1の鍵とその証明書を送信するステップと、
- 少なくとも一のデータのハッシュ関数を送信するステップと、
- チャレンジを受信するステップと、
- 複数の秘密値を送信するステップと、
- 複数のイメージ値を送信するステップと、
- 複数のツリー値を送信するステップと、そして、
- 前記データを送信するステップを備えていることを特徴とするデータ

タ処理方法。

- [請求項26] 請求項 25 に記載したデータ処理方法であって、更に、
秘密値の範囲を選択するステップと、そして、
前記範囲内における前記秘密値の半分を選択するステップを備えていることを特徴とするデータ処理方法。
- [請求項27] 請求項 25 に記載したデータ処理方法であって、更に、
シード値をハッシュ関数により処理し、もって、第 1 の秘密値を得るステップと、
前記シード値をハッシュ関数により処理し、もって、第 2 の秘密値を得るステップと、
前記第 1 と第 2 の秘密値の一方を、前記チャレンジの少なくとも一のビットに従って選択するステップと、
前記選択された秘密値をハッシュ関数によって処理し、もって、イメージ値を得るステップと、そして、
前記選択した秘密値を廃棄するステップを備えていることを特徴とするデータ処理方法。
- [請求項28] 請求項 27 に記載したデータ処理方法において、前記シード値を処理する前記ステップ、秘密を選択する前記ステップ、前記選択した秘密を処理するステップ、そして、前記選択した秘密を廃棄する前記ステップが繰り返され、もって、一連の秘密値と一連のイメージ値とを得ることを特徴とするデータ処理方法。

[図1]

図 1



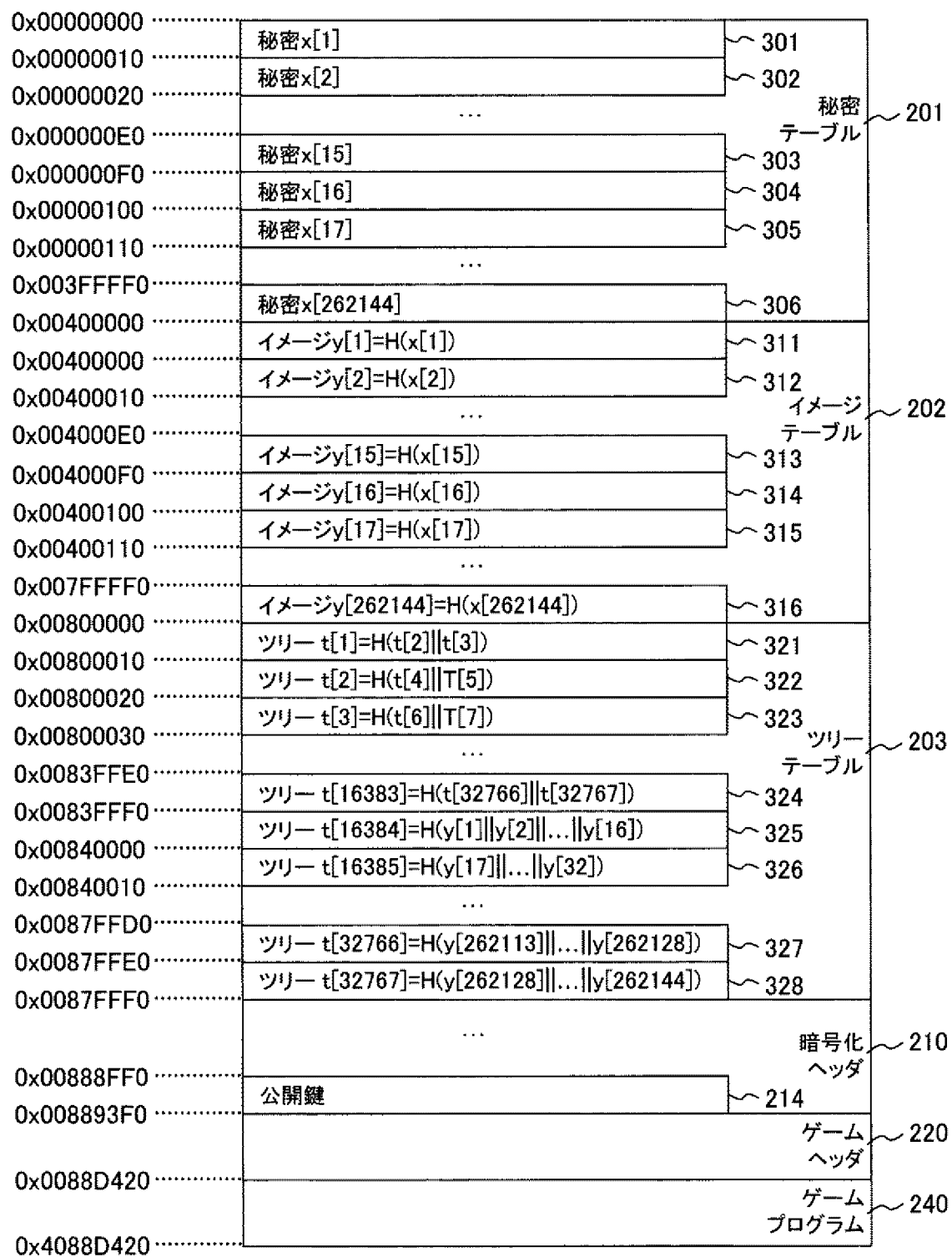
[図2]

図 2

0x0000000000000000	秘密テーブル	ROM, S	201	暗号化 データ
0x0000000400000000	イメージテーブル	ROM, U	202	
0x0000000800000000	ツリーテーブル	ROM, U	203	
0x000000087FFF0000				
...				
0x0000001000000000	システムパラメータ	ROM, U	211	暗号化 ヘッダ
0x0000001003000000	ルート証明書	ROM, U	212	
0x0000001006000000	ゲーム証明書	ROM, U	213	
0x0000001009000000	公開鍵	ROM, U	214	
0x0000001009400000	署名インデックス	コントローラ, S	215	
0x0000001009402000				
...				
0xFF00000000000000	ゲームID	ROM, U	221	ゲーム ヘッダ
0xFF00000000000010	ゲーム鍵	ROM, U	222	
0xFF00000000000020	プログラムのエントリポイント	ROM, U	223	
0xFF00000000000030	セクタ 1 のハッシュ	ROM, U	224	
0xFF00000000000040	...			
0xFF0000000000004020	セクタ1024のハッシュ	ROM, U	225	
0xFF0000000000004030				
...				
0xFFF0000000000000	状態テーブル	EEPROM, U	231	セーブした ゲーム状態
0xFFF0000003000000	セーブしたゲーム 1	EEPROM, U	232	
0xFFF0000006000000	...			
0xFFF000002D000000	セーブしたゲーム 16	EEPROM, U	233	
0xFFF0000030000000				
...				
0xFFFF000000000000	セクタ 1: ファームウェアアップデート	ROM, U	241	ゲーム プログラム
0xFFFF001000000000	...			
0xFFFF3FE000000000	セクタ 1024	ROM, U	242	
0xFFFF400000000000				

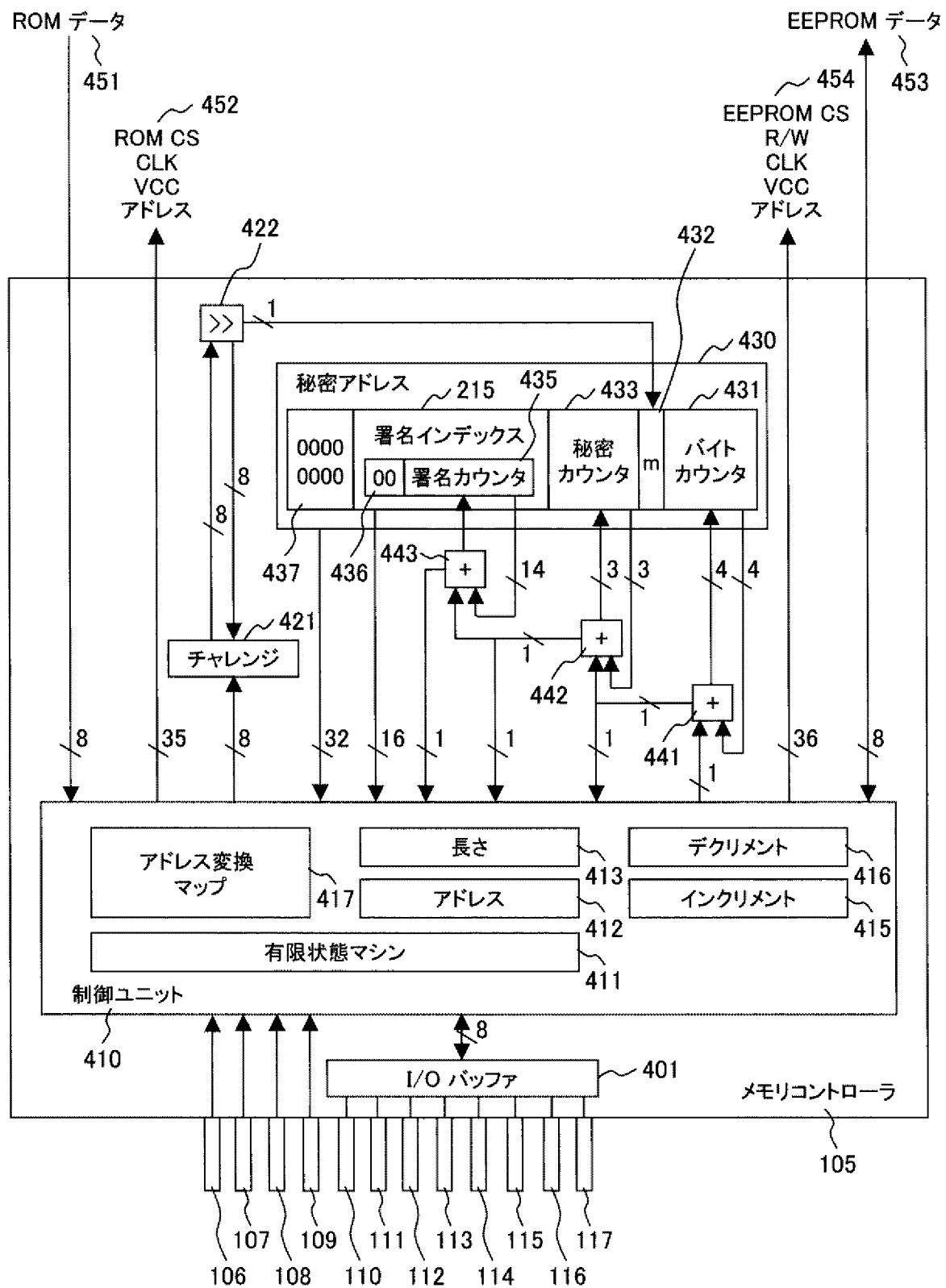
[図3]

図 3



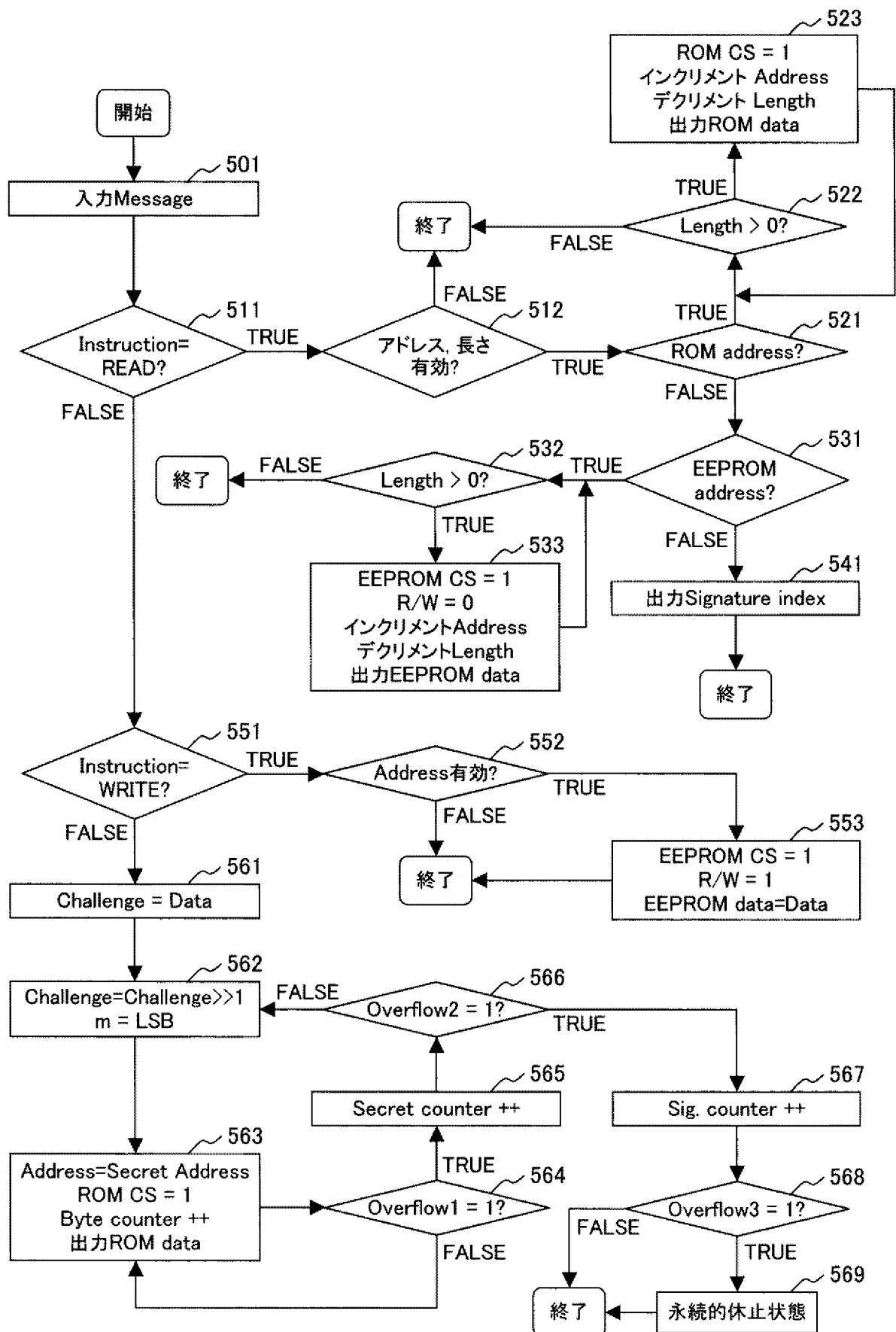
[図4]

図 4



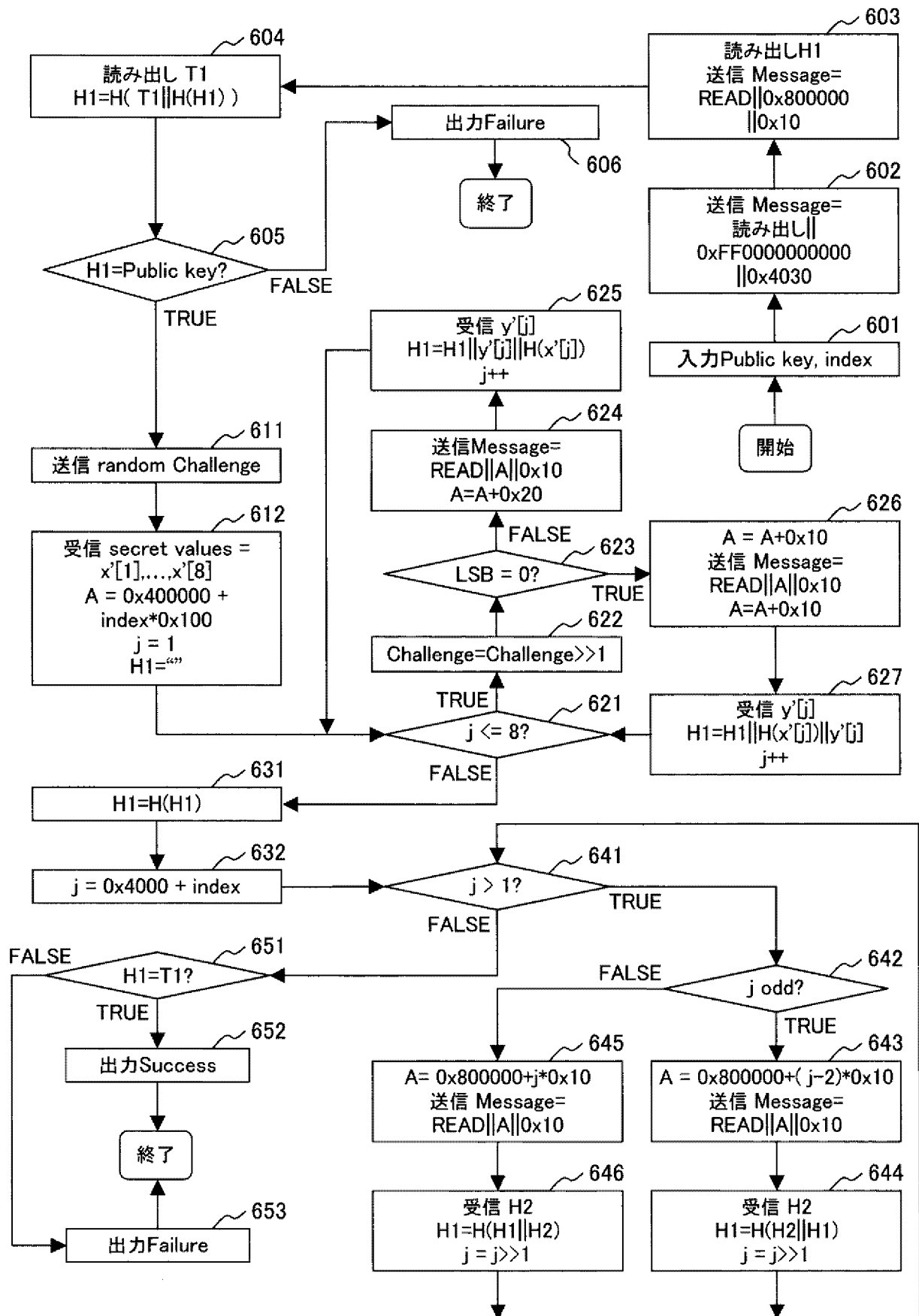
[図5]

図 5



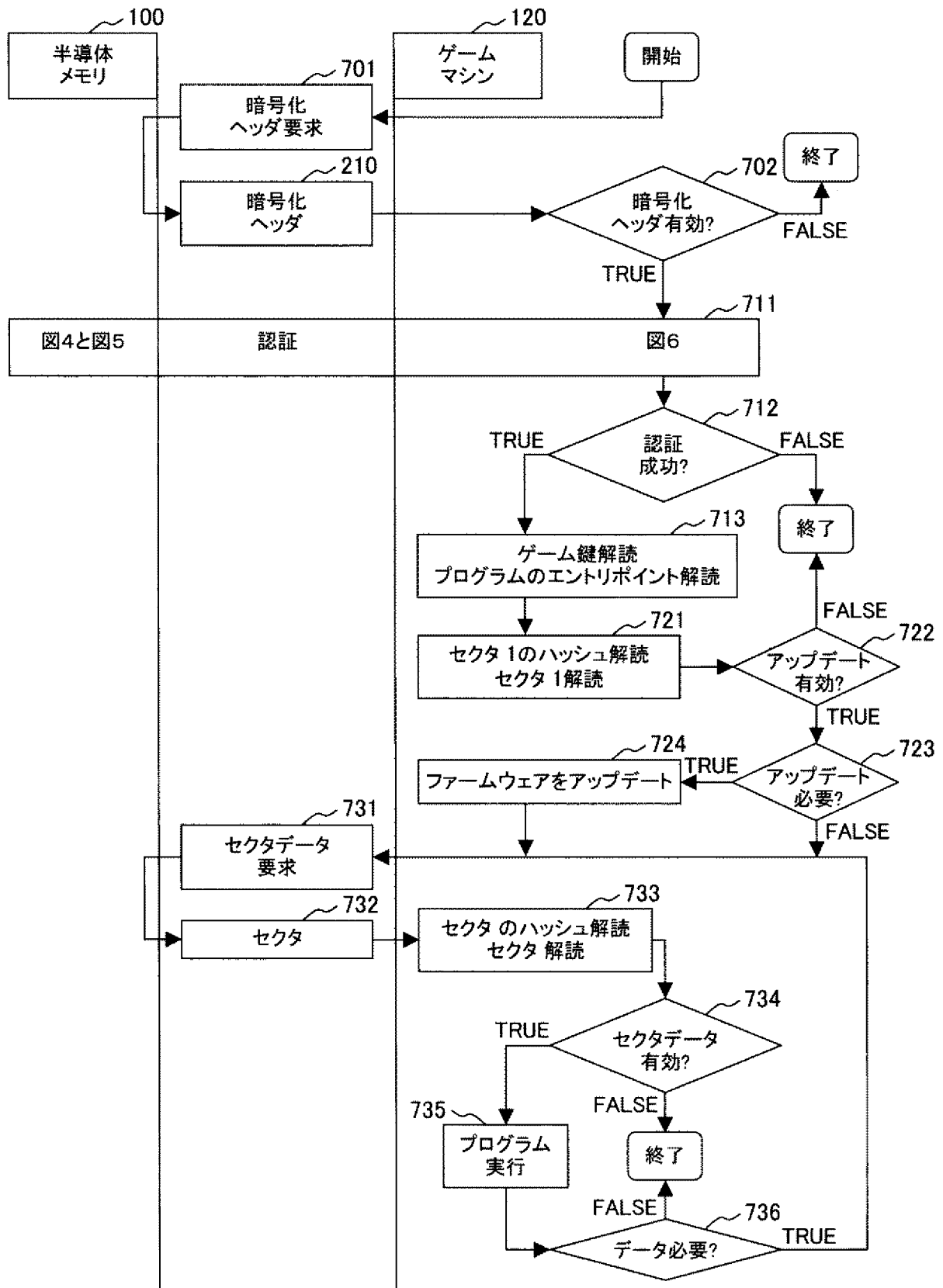
[図6]

図 6



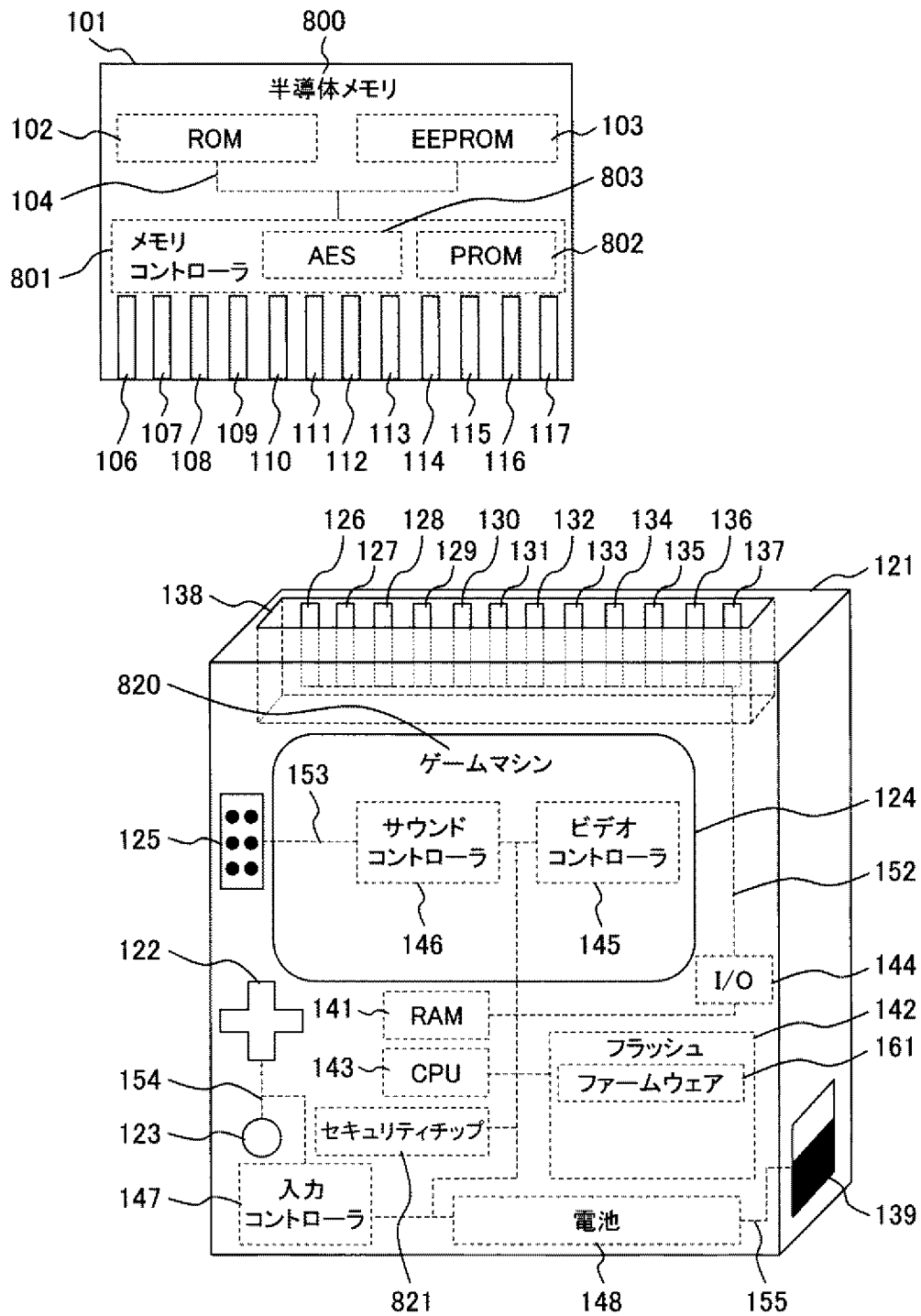
[図7]

図 7



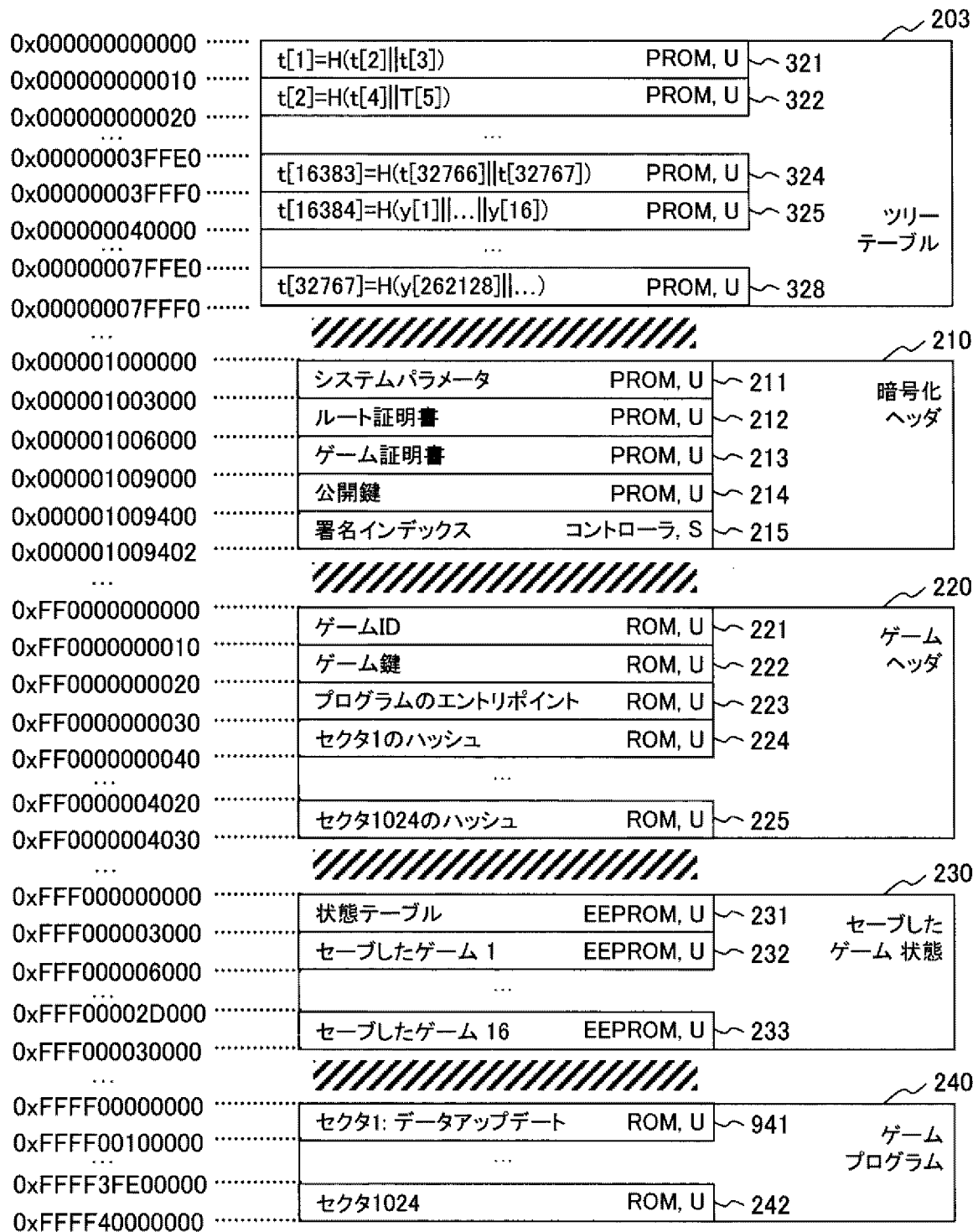
[図8]

図 8



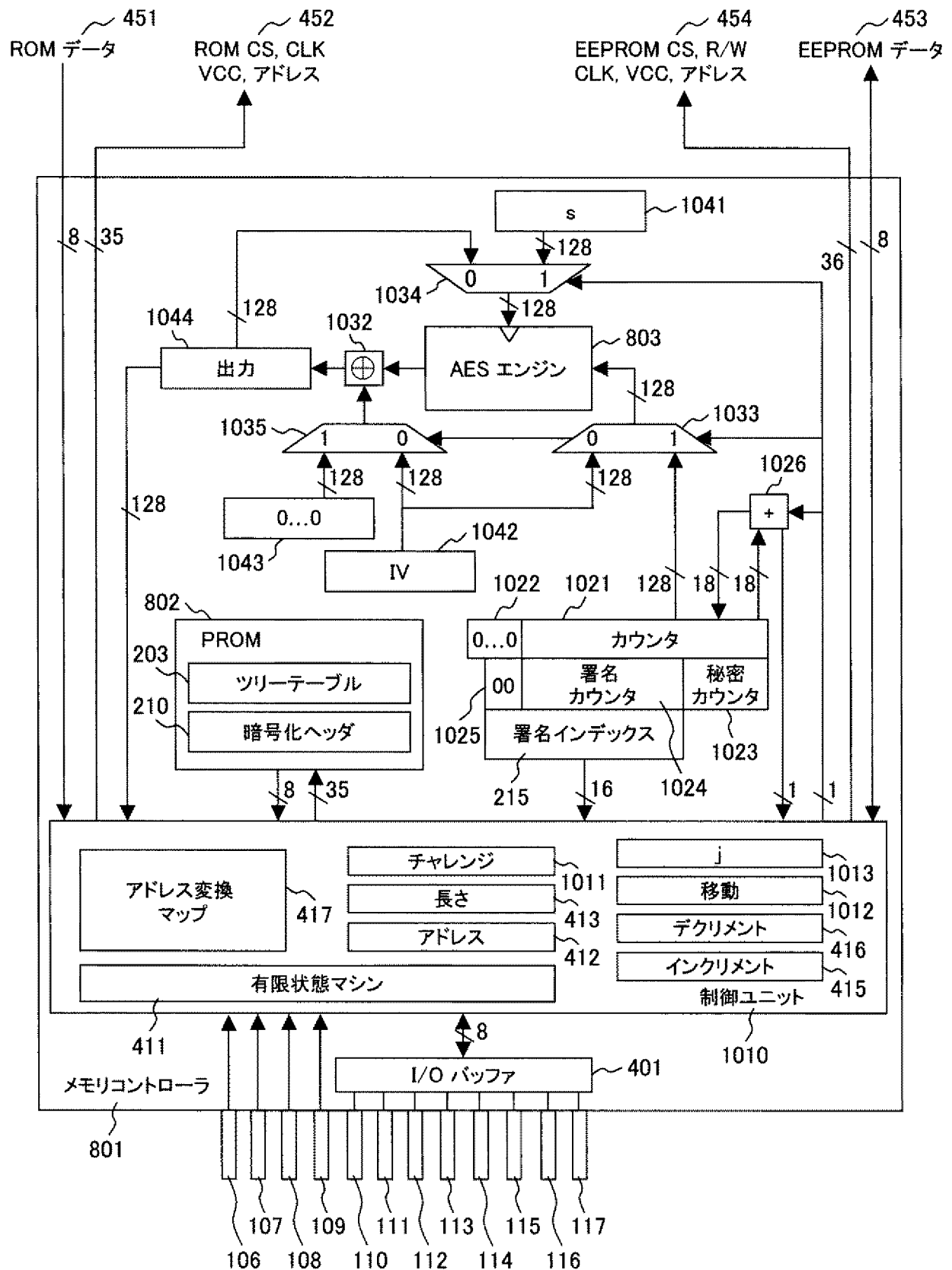
[図9]

図 9



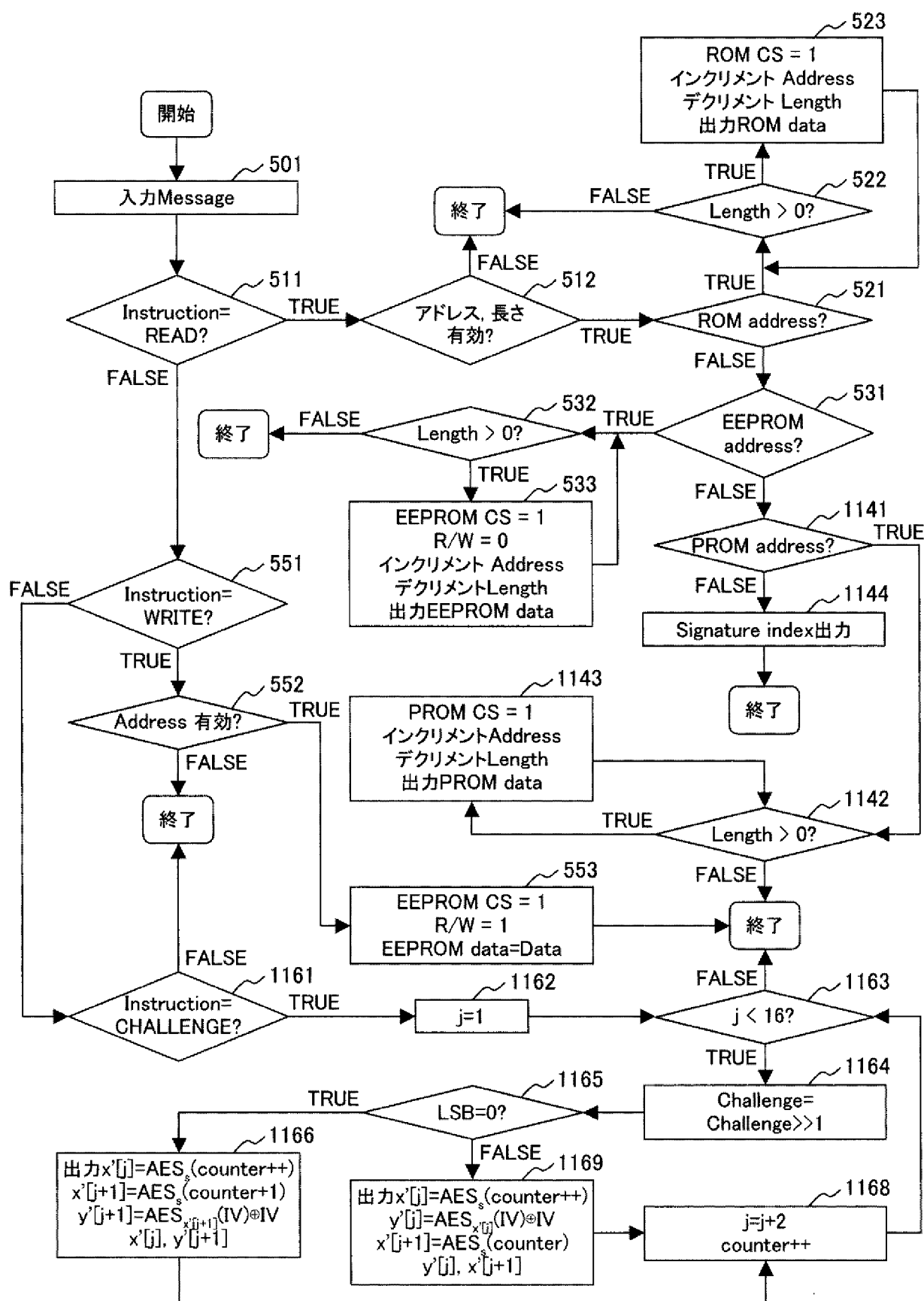
[図10]

図 10



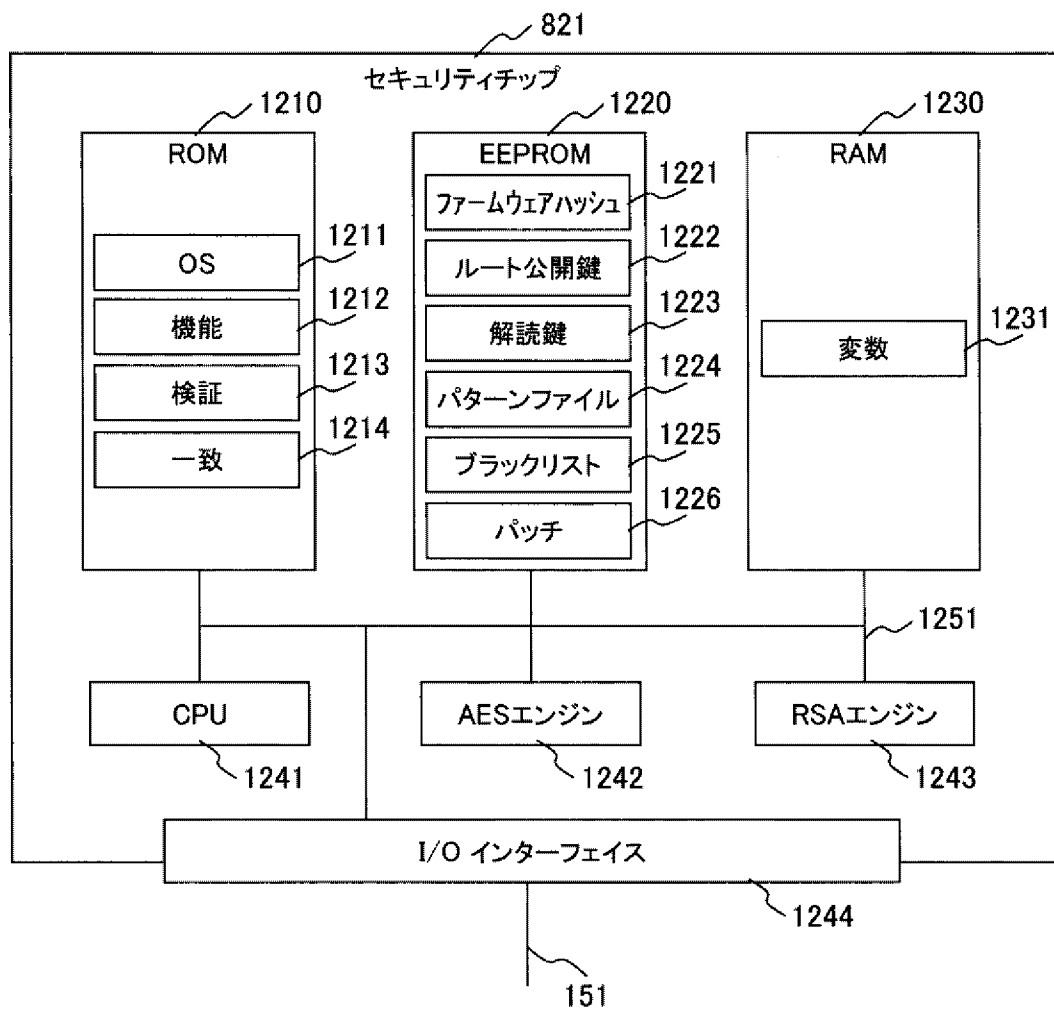
[図11]

図 11



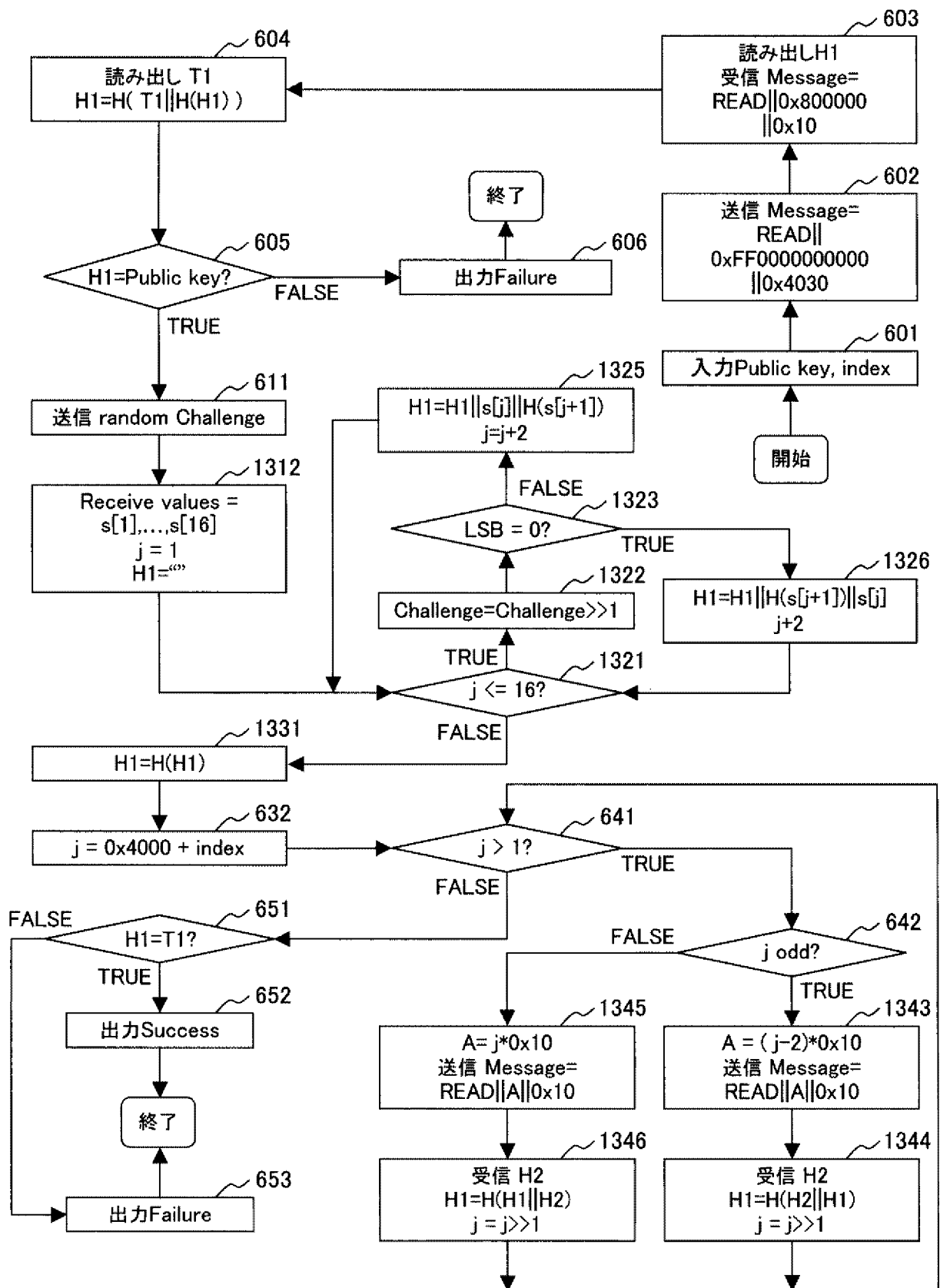
[図12]

図 12



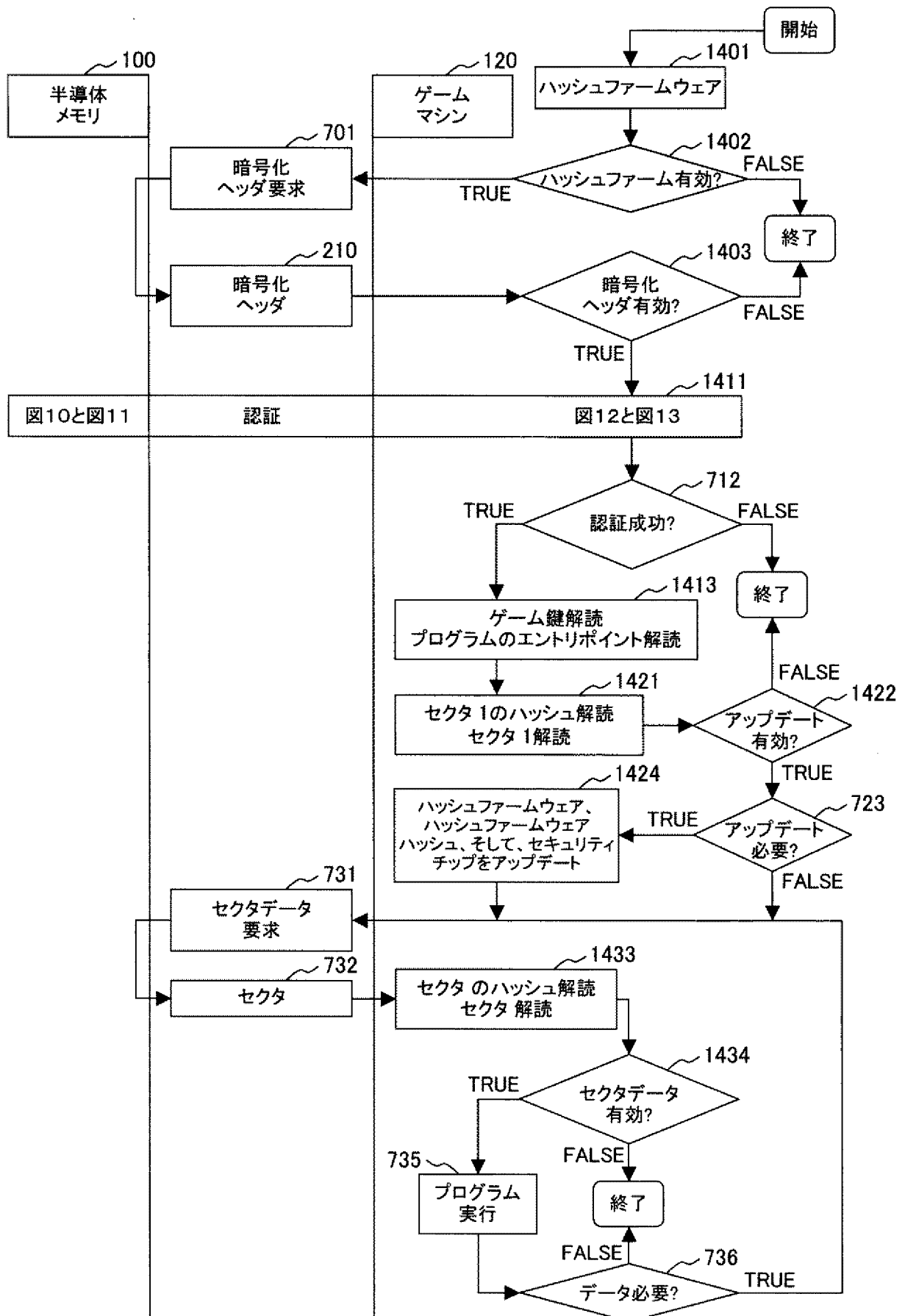
[図13]

図 13



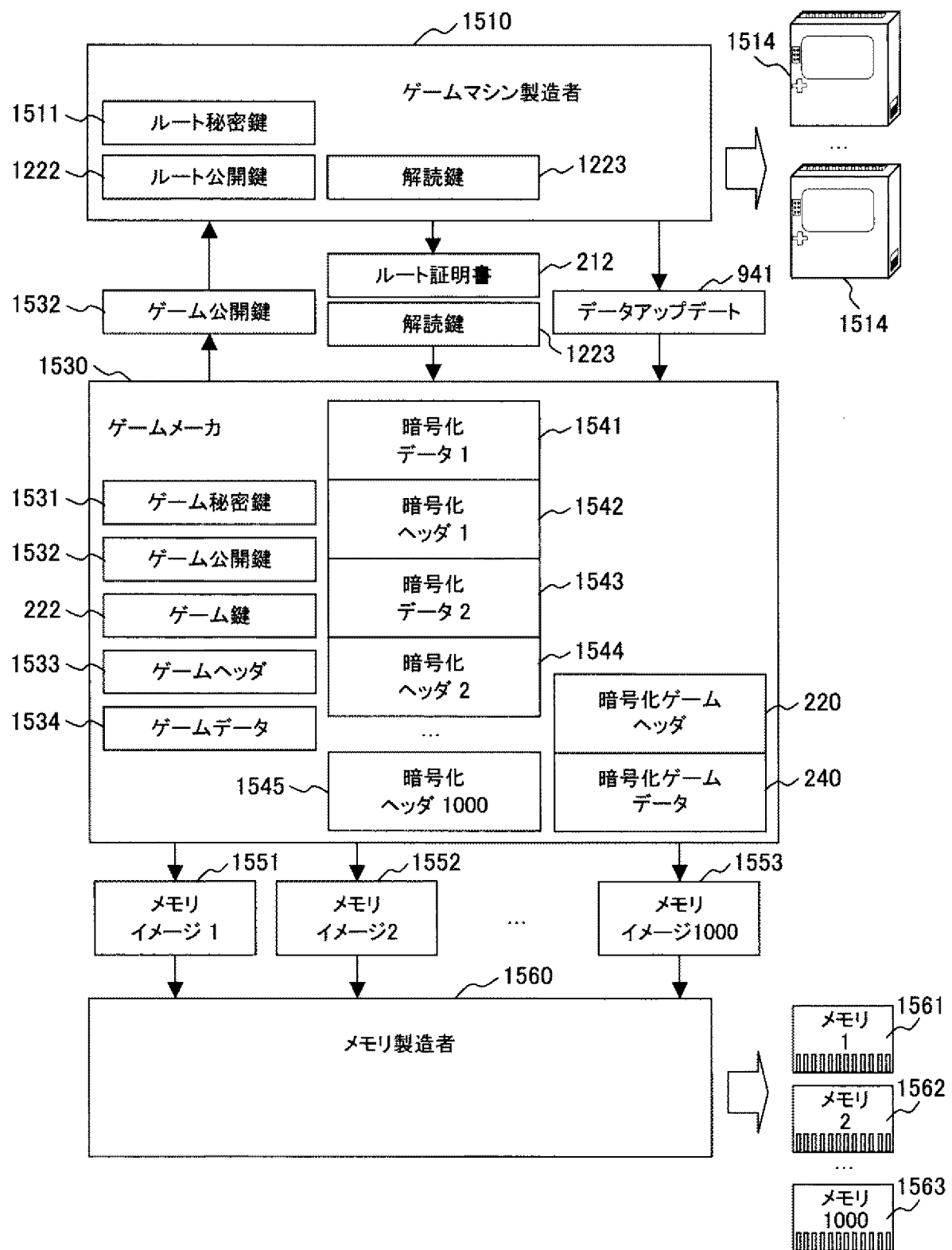
[図14]

図 14



[図15]

図 15



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2009/006158

A. CLASSIFICATION OF SUBJECT MATTER

H04L9/08(2006.01) i, G06F21/24(2006.01) i, H04L9/32(2006.01) i

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)

H04L9/08, G06F21/24, H04L9/32

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

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2010
Kokai Jitsuyo Shinan Koho	1971-2010	Toroku Jitsuyo Shinan Koho	1994-2010

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Camille Vuillaume, Katsuyuki Okeya, Erik Dahmen, "Public Key Authentication out of Memory Operations", 2009 Nen Symposium on Cryptography and Information Security, 20 January 2009 (20.01.2009), 3F3 Kokai Kagi Ango(4), 3F3-1	1-28
Y	JP 2000-76063 A (Open Loop Inc.), 14 March 2000 (14.03.2000), entire text; all drawings (Family: none)	1-28
Y	JP 62-72383 A (Konami Kogyo Kabushiki Kaisha), 02 April 1987 (02.04.1987), entire text; all drawings (Family: none)	1-10

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search
05 February, 2010 (05.02.10)

Date of mailing of the international search report
16 February, 2010 (16.02.10)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

A. 発明の属する分野の分類（国際特許分類（I P C））

Int.Cl. H04L9/08(2006.01)i, G06F21/24(2006.01)i, H04L9/32(2006.01)i

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（I P C））

Int.Cl. H04L9/08, G06F21/24, H04L9/32

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	1 9 2 2 - 1 9 9 6 年
日本国公開実用新案公報	1 9 7 1 - 2 0 1 0 年
日本国実用新案登録公報	1 9 9 6 - 2 0 1 0 年
日本国登録実用新案公報	1 9 9 4 - 2 0 1 0 年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
Y	Camille Vuillaume, Katsuyuki Okeya, Erik Dahmen, "Public Key Authentication out of Memory Operations", 2 0 0 9 年暗号と情報セキュリティシンポジウム, 2009.01.20, 3 F 3 公開鍵暗号(4), 3 F 3 - 1	1 - 2 8
Y	JP 2000-76063 A (株式会社オープンループ) 2000.03.14, 全文, 全図 (ファミリーなし)	1 - 2 8

☒ C 欄の続きにも文献が列挙されている。☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「&」同一パテントファミリー文献

国際調査を完了した日

0 5 . 0 2 . 2 0 1 0

国際調査報告の発送日

1 6 . 0 2 . 2 0 1 0

国際調査機関の名称及びあて先

日本国特許庁（I S A / J P）

郵便番号 1 0 0 - 8 9 1 5

東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）

青木 重徳

5 S

4 2 2 9

電話番号 0 3 - 3 5 8 1 - 1 1 0 1 内線 3 5 4 6

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
Y	JP 62-72383 A (コナミ工業株式会社) 1987. 04. 02, 全文, 全図 (ファミリーなし)	1 - 1 0