(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5398824号
(P5398824)
(45)発行日　平成26年1月29日(2014.1.29)　(24)登録日　平成25年11月1日(2013.11.1)

(51)Int.Cl.　　F I
*G06F 21/12 (2013.01)*　　G06F 21/22 112L
*G06F 9/445 (2006.01)*　　G06F 9/06 640

請求項の数 27　(全 63 頁)

(21)出願番号　特願2011-510771 (P2011-510771)
(86)(22)出願日　平成21年10月9日 (2009.10.9)
(65)公表番号　特表2012-505437 (P2012-505437A)
(43)公表日　平成24年3月1日 (2012.3.1)
(86)国際出願番号　PCT/JP2009/005289
(87)国際公開番号　WO2010/041467
(87)国際公開日　平成22年4月15日 (2010.4.15)
審査請求日　平成24年6月28日 (2012.6.28)
(31)優先権主張番号　特願2008-264530 (P2008-264530)
(32)優先日　平成20年10月10日 (2008.10.10)
(33)優先権主張国　日本国 (JP)
(31)優先権主張番号　特願2008-321540 (P2008-321540)
(32)優先日　平成20年12月17日 (2008.12.17)
(33)優先権主張国　日本国 (JP)

(73)特許権者　000005821
パナソニック株式会社
大阪府門真市大字門真１００６番地
(74)代理人　100109210
弁理士　新居　広守
(72)発明者　ケネス　アレクサンダー　ニコルソン
日本国大阪府門真市大字門真１００６番地
パナソニック株式会社内
(72)発明者　松島　秀樹
日本国大阪府門真市大字門真１００６番地
パナソニック株式会社内
(72)発明者　高山　久
日本国大阪府門真市大字門真１００６番地
パナソニック株式会社内

最終頁に続く

(54)【発明の名称】セキュア処理システムのアプリケーション空間において信頼性を実現するための一時的ＰＣＲ利用

(57)【特許請求の範囲】
【請求項１】
情報処理装置であって、
複数のモジュール各々に先立ってロードされているモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、
前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示す、何れの前記アクティブモジュールも、ロード済みでまだ終了していないモジュールであるアクティブ情報を記録する管理手段と、
次のモジュールをロードする場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、
(v)アクティブモジュールの前記リストが正常に生成された場合には前記次のモジュールをロードし、かつ、
(vi)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモ

ジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御手段とを備える
ことを特徴とする情報処理装置。
【請求項２】
請求項１に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールが終了する場合に、前記次のモジュールはアクティブモジュールでないと示すように前記アクティブ情報を更新するよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項３】
請求項１に記載の情報処理装置であって、
前記管理手段は、有向非巡回グラフを用いて、１以上の前記アクティブモジュールを示す情報を管理する
ことを特徴とする情報処理装置。
【請求項４】
請求項３に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールをロードする場合に、前記次のモジュールと当該次のモジュールの期待プラットフォーム情報とを示すノードを生成し、かつ、従属するそのモジュールに対応する１以上のノードに、前記生成したノードが従属するように、前記生成したノードを前記有向非巡回グラフに付け加えるよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項５】
請求項４に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールがロードされて終了した場合に、前記次のモジュールを示すノードと、前記次のモジュールを示す前記ノードに従属するノード全てとを削除するよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項６】
請求項５に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールを示す前記ノードが従属することになっている親ノードを検索して、各ノードの期待プラットフォーム情報を前記有向非巡回グラフのルートから前記親ノードへ蓄積することによって前記蓄積プラットフォーム情報を生成する
ことを特徴とする情報処理装置。
【請求項７】
請求項１に記載の情報処理装置であって、
前記ロード制御手段は前記蓄積プラットフォーム情報を所定期間の後に削除する
ことを特徴とする情報処理装置。
【請求項８】
請求項６に記載の情報処理装置であって、
前記ロード制御手段は、前記複数のモジュールのうち１つが正常にロードされるたびに前記蓄積プラットフォーム情報を削除し、前記複数のモジュールのうち１つをロードすることになるたびに蓄積プラットフォーム情報を生成する
ことを特徴とする情報処理装置。
【請求項９】
請求項１に記載の情報処理装置であって、
前記複数のモジュールは、それぞれが１以上のモジュールを含む第１モジュールグループと第２モジュールグループとを含み、
前記情報処理装置は、さらに、

前記第１モジュールグループのうち、何れの１以上のモジュールをロードしたかを示す第１蓄積プラットフォーム情報を格納する登録手段を備え、
前記格納手段は、さらに、前記第２モジュールグループのモジュールをロードする前に前記第１モジュールグループのモジュール全てをロードすべきことを示す第１期待プラットフォーム情報を格納し、
前記ロード制御手段は、
前記第１モジュールグループのモジュールに対して、(i)当該モジュールを検証し、(ii)検証が成功した場合には前記モジュールをロードし、かつ、(iii)前記モジュールがロードされた場合には前記モジュールのプラットフォーム情報を前記第１蓄積プラットフォーム情報に蓄積することによって前記第１蓄積プラットフォーム情報を更新し、
前記第２モジュールグループのモジュールをロードする場合には、(i)前記第１期待プラットフォーム情報を前記登録手段に格納された第１蓄積プラットフォーム情報と比較することによって第１モジュールグループのモジュール全てが正常にロードされたことを検証し、
前記第１モジュールグループのモジュール全てが正常にロードされたことが検証された場合には、
前記ロード制御手段は、
(i)第２モジュールグループのモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、
(v)アクティブモジュールの前記リストが正常に生成された場合に、前記次のモジュールをロードし、かつ、
(vi)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項１０】
請求項９に記載の情報処理装置であって、
前記第１モジュールグループは、システム層の１以上のモジュールを含み、前記第２モジュールグループは、アプリケーション層の１以上のモジュールを含む
ことを特徴とする情報処理装置。
【請求項１１】
情報処理装置用の情報処理方法であって、
前記情報処理装置は、
複数のモジュール各々に先立ってロードされていると予期されるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、
前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示す、何れの前記アクティブモジュールも、ロード済みでまだ終了していないモジュールであるアクティブ情報を記録する管理手段とを備え、
前記情報処理方法は、
前記アクティブモジュールに続く次のモジュールをロードする場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、

(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、
(v)アクティブモジュールの前記リストが正常に生成された場合に、前記次のモジュールをロードし、かつ、
(vi)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御ステップを備える
ことを特徴とする情報処理方法。

【請求項１２】
情報処理装置用の記録媒体に記録されたコンピュータプログラムであって、
前記情報処理装置は、
複数のモジュール各々に先立ってロードされていると予期されるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、
前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示す、何れの前記アクティブモジュールも、ロード済みでまだ終了していないモジュールであるアクティブ情報を記録する管理手段とを備え、
前記コンピュータプログラムは、ロード制御ステップを前記情報処理装置に実行させるためのコンピュータプログラムであって、
前記ロード制御ステップでは、前記アクティブモジュールに続く次のモジュールをロードする場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、
(v)アクティブモジュールの前記リストが正常に生成された場合には前記次のモジュールをロードし、かつ、
(vi)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する
ことを特徴とするコンピュータプログラム。

【請求項１３】
情報処理装置に用いられる集積回路デバイスであって、
前記情報処理装置は、
複数のモジュール各々に先立ってロードされていると予期されるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、
前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示す、何れの前記アクティブモジュールも、ロード済みでまだ終了していないモジュールであるアクティブ情報を記録する管理手段とを備え、
前記集積回路デバイスは、
前記アクティブモジュールに続く次のモジュールをロードする場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、

(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、
(v)アクティブモジュールの前記リストが正常に生成された場合には前記次のモジュールをロードし、かつ、
(vi)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御手段を備える
ことを特徴とする集積回路デバイス。
【請求項１４】
請求項１記載の情報処理装置であって、
当該情報処理装置は、サーバに接続され、
前記ロード制御手段は、予期される蓄積プラットフォーム情報を検証する要求を前記サーバから受信した場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、かつ、
(v)アクティブモジュールの前記リストが正常に生成された場合には、前記蓄積プラットフォーム情報を前記サーバに送信する
ことを特徴とする情報処理装置。
【請求項１５】
請求項１４に記載の情報処理装置であって、
前記ロード制御手段は、さらに、
(i)前記蓄積プラットフォーム情報を生成するために、どの期待プラットフォームを用いるかを示す情報を生成し、
(ii)前記情報に基づいて、前記蓄積プラットフォーム情報を検証するために用いる署名情報を生成し、
(iii)前記署名情報が付加されている前記蓄積プラットフォーム情報を送信する
ことを特徴とする情報処理装置。
【請求項１６】
請求項１４に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールが終了する場合に、前記次のモジュールはアクティブモジュールでないと示すように前記アクティブ情報を更新するよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項１７】
請求項１４に記載の情報処理装置であって、
前記管理手段は、有向非巡回グラフを用いて、１以上の前記アクティブモジュールを示す情報を管理する
ことを特徴とする情報処理装置。
【請求項１８】
請求項１７に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールをロードする場合に、前記次のモジュールと当該次のモジュールの前記期待プラットフォーム情報とを示すノードを生成し、かつ、

従属するそのモジュールに対応する１以上のノードに、前記生成したノードが従属するように、前記生成したノードを前記有向非巡回グラフに付け加えるよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項１９】
請求項１８に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールがロードされて終了した場合に、前記次のモジュールを示すノードと、前記次のモジュールを示す前記ノードに従属するノード全てとを削除するよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項２０】
請求項１９に記載の情報処理装置であって、
前記ロード制御手段は、前記次のモジュールを示す前記ノードが従属することになっている親ノードを検索して、各ノードの期待プラットフォーム情報を前記有向非巡回グラフのルートから前記親ノードへ蓄積することによって前記蓄積プラットフォーム情報を生成する
ことを特徴とする情報処理装置。
【請求項２１】
請求項１４に記載の情報処理装置であって、
前記ロード制御手段は前記蓄積プラットフォーム情報を所定期間の後に削除する
ことを特徴とする情報処理装置。
【請求項２２】
請求項２０に記載の情報処理装置であって、
前記ロード制御手段は、前記複数のモジュールのうち１つが正常にロードされるたびに前記蓄積プラットフォーム情報を削除し、前記複数のモジュールのうち１つをロードすることになるたびに蓄積プラットフォーム情報を生成する
ことを特徴とする情報処理装置。
【請求項２３】
請求項１４に記載の情報処理装置であって、
前記複数のモジュールは、それぞれが１以上のモジュールを含む第１モジュールグループと第２モジュールグループとを含み、
前記情報処理装置は、さらに、
前記第１モジュールグループのうち、何れの１以上のモジュールをロードしたかを示す第１蓄積プラットフォーム情報を格納する登録手段を備え、
前記格納手段は、さらに、前記第２モジュールグループのモジュールをロードする前に前記第１モジュールグループのモジュール全てをロードすべきことを示す第１期待プラットフォーム情報を格納し、
前記ロード制御手段は、
前記第１モジュールグループのモジュールに対して、(i)当該モジュールを検証し、(ii)検証が成功した場合には前記モジュールをロードし、かつ、(iii)前記モジュールがロードされた場合には前記モジュールのプラットフォーム情報を前記第１蓄積プラットフォーム情報に蓄積することによって前記第１蓄積プラットフォーム情報を更新し、
前記第２モジュールグループのモジュールをロードする場合には、(i)前記第１期待プラットフォーム情報を前記登録手段に格納された第１蓄積プラットフォーム情報と比較することによって第１モジュールグループのモジュール全てが正常にロードされたことを検証し、
前記第１モジュールグループのモジュール全てが正常にロードされたことが検証された場合には、
前記ロード制御手段は、
(i)第２モジュールグループのモジュールのうち、どれがアクティブモジュールである

かを前記アクティブ情報を用いて判断して、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、
(v)アクティブモジュールの前記リストが正常に生成された場合に、前記次のモジュールをロードし、かつ、
(vi)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する
ことを特徴とする情報処理装置。
【請求項２４】
請求項２３に記載の情報処理装置であって、
前記第１モジュールグループは、システム層の１以上のモジュールを含み、前記第２モジュールグループは、アプリケーション層の１以上のモジュールを含む
ことを特徴とする情報処理装置。
【請求項２５】
請求項１１記載の情報処理方法であって、
前記蓄積プラットフォーム情報を送信する要求をサーバから受信する受信ステップと、
前記受信ステップにおいて前記要求を受信した場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、かつ、
(v)アクティブモジュールの前記リストが正常に生成された場合には、前記蓄積プラットフォーム情報を前記サーバに送信する、送信ステップとを含む
ことを特徴とする情報処理方法。
【請求項２６】
請求項１２記載のコンピュータプログラムであって、
前記蓄積プラットフォーム情報を送信する要求をサーバから受信する受信ステップと、
前記受信ステップにおいて前記要求を受信した場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、かつ、
(v)アクティブモジュールの前記リストが正常に生成された場合には、前記蓄積プラットフォーム情報を前記サーバに送信する送信ステップとを前記情報処理装置に実行させるための
コンピュータプログラム。
【請求項２７】

請求項１３記載の集積回路デバイスであって、
前記蓄積プラットフォーム情報を送信する要求をサーバから受信する受信手段と、
前記受信手段が前記要求を受信した場合に、
(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、
(ii)それぞれの前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、
(iii)前記次のモジュールの前記期待プラットフォーム情報を特定し、
(iv)１以上の前記アクティブモジュールからの１以上のモジュールのリストであって、当該リストの前記蓄積プラットフォーム情報は、前記次のモジュールの前記期待プラットフォーム情報に等しいリストを生成し、かつ、
(v)アクティブモジュールの前記リストが正常に生成された場合には、前記蓄積プラットフォーム情報を前記サーバに送信する、送信手段とを備える
ことを特徴とする集積回路デバイス。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、アクティブモジュールと、アクティブモジュールの次のモジュールとをロードする情報処理装置に関する。
【背景技術】
【０００２】
例えば、非特許文献１、および、非特許文献２のような先導的な先行文献は、装置を、どのようにして、保証、また、信頼できる方法で起動するかを記述している。これらの方法は、信頼性や安全性が、ブートプロセスの間保持されるということを保証するため、徹底的に見直されてきており、その結果、安全にブートができる装置の実装を要求する人々に、有用な基準を提供している。このセキュアブートプロセスのキーコンポーネントは、RIM証明書である。これは、現在期待されるプラットフォームの状態が、どのようなものであるべきかを定義する、署名つきの構造体である。プラットフォーム状態は、プラットフォーム設定レジスタ(PCRs)の集合に対するハッシュによって表され、各ＰＣＲは、一般に、定義されたハッシュ値を含む。これらのPCRは、完全性測定（integrity measurement）情報として用いられるものであり、期待される装置の状態を定義するために、RIM証明書に記録されていてもよい。さらに、RIM証明書はまた、現在の状態が検証される場合に、拡張されるべきPCRを特定する。この拡張の処理では、特定されたPCRの値を取得し、RIM証明書内で定義される新たな既知の値に連結された前のPCR値に基づいて、新たなハッシュ値を算出する。TCGによって定義される典型的なセキュアブートシーケンスは、コアコンポーネントの初期化および自己検証から開始する。ここで、コアコンポーネントとは、検証および測定の信頼性のルート(RTV+RTM)、MTM自体、および関連するコアMTMインターフェースコンポーネント等である。次に、ファームウェアの他の部分をサポートする追加コンポーネントが、信頼できる方法で起動される。ここで、信頼できる方法とは、各コンポーネントが、信頼性のあるコンポーネントから立ち上げられたものであることを保証するために、制御が渡される前に、信頼済みのコンポーネントにより、各コンポーネントを検証してから、自分自身を検証するような方法である。検証→実行→自己検証というこのシーケンスには、システム内において、信頼性のルートから、各コンポーネントへ、動的に信頼性の境界を拡張するという効果がある。そして、最後に、オペレーティングシステムを起動して、MTMサービスへとアクセスするクライアントアプリケーションに対し、安全で信頼性のあるパスを提供する。
【先行技術文献】
【特許文献】
【０００３】
【特許文献１】米国特許出願公開第2006/0212939号明細書

【非特許文献】
【０００４】
【非特許文献１】tTrusted Computing Group (TCG) Mobile Trusted Module (MTM) documents TCG Mobile Reference Architecture version 1.0 12 June 2007
【非特許文献２】TCG Mobile Trusted Module Specification version 1.0 12 June 2007
【非特許文献３】TCG TPM Specification Version 1.2 Revision 103
【発明の概要】
【発明が解決しようとする課題】
【０００５】
しかしながら、セキュアブートプロセスで、PCRへの書き込みが終了すると、問題が生じる。上述したセキュアなブートコンポーネントとは異なり、通常のアプリケーションは、ユーザからの作用、プログラム障害、またはアプリケーション改ざんの検出でさえ原因となって、当然ながら、終了する。非特許文献３では、特別な状況下において、いくつかのPCRをリセットすることが許可されているが、「TCG Mobile Trusted Module Specification v1.0 Revision 1」では、RIM証明書により制御されるPCRは、リセット可能とすべきではないと記載されている。「TCG Mobile Reference Architecture v1.0 Revision 1」の参考コメントでは、この問題に対して、何もしない、１回目の実行に限り拡張する、または１つのPCRを繰り返し拡張するという、３つの解決策が提案されている。何もしないことでは、アプリケーションの安全性または信頼性は改善されない。１回目の実行に限り拡張するということでは、アプリケーションをカバーできるように、信頼性の境界は拡張されるけれども、信頼性がある境界内のアプリケーションを、不正なプロセスが、強制的に終了させて、以前信頼されていたアプリケーションに成り済ますことが可能であることを意味する。最後に、拡張の繰り返しでは、複数のRIM証明書作成および格納によるオーバーヘッドがあり、ランタイムで、要求に応じてRIM証明書を作成するので、システムを攻撃する別の経路が与えられる。さらに、PCRでは、リソースが制限されている。非特許文献２の第50頁第5.3.2項では、セキュアブート中などデバイスメーカー向けに13個のPCRが確保されているが、最悪の場合、アプリケーション向けには、その他のPCRが3つしか残されていないため、複数のアプリケーションが互いに無関係な場合でも、複数のアプリケーション開発者間で、これらのPCRの使用の調整が重大な問題となる。
【０００６】
特許文献１では、指定されたPCRの無限の集合を管理するコンテキストを作成するという手段によってPCR数を増やす方法が開示されているが、RIM証明書の扱い方については考慮されていない。また、全ての仮想PCRを単一の物理的PCRに集約するという開示された方法は、RIM証明書を介して、仮想PCRの一部だけを、どのようにテストするのか、つまりRIM証明書の重要な面について教示していない。さらに、この方法では、単一の物理的PCRへ仮想PCRを集約することが、仮想PCRの存在を認識していないけれども他の目的でその物理的PCRを使用したいアプリケーションと互いに影響し合うことになるという問題をどのように回避するのか教示していない。その上、この方法では、拡張動作をどのようにして効率的に取り消すのか教示していない。ここで、拡張動作を取り消すとは、あるアプリケーションを終了する場合に、このアプリケーションと、この終了されるアプリケーションの従属アプリケーション全てとの周辺に広がる仮想PCRを利用することによって構築された信頼性の境界を、動的に縮小して、信頼性のあるアプリケーションの集合から、これらのアプリケーションを削除するといったことである。代わりに、この方法では、仮想PCRをリセットしてもよいとしか教示していないため、信頼性の境界を再構築する唯一の方法は、従属アプリケーションだけでなく、従属として終了アプリケーションを有する全てのアプリケーションも終了した後に、それら全てを再検証かつ再実行して、信頼性の境界を一から再構築することである。
【０００７】
したがって、１以上のモジュールが終了した後でも、信頼性のある境界に従って動的にPCRの値を変更かつ生成できる装置が必要である。

【０００８】
さらに、「Trusted Computing Group’s (TCG) Trusted Platform Module (TPM) documents」などの先導的な先行文献では、プラットフォームと特定のクライアント両方のリモート認証をどのように確立するか説明されている。セキュアブートプロセスによりプラットフォーム状態が保証されるので、プラットフォームの認証は必ずしも必要ではない。しかしながら、MTMベースのプラットフォーム上で実行するアプリケーションには、認証が対応していなかった。
【０００９】
したがって、動的に変化するPCRsの状態をサーバに認証させることができる装置がさらに必要である。
【課題を解決するための手段】
【００１０】
本発明の第１の形態における情報処理装置は、複数のモジュール各々に先立ってロードされるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示すアクティブ情報を記録する管理手段と、前記アクティブモジュールの次のモジュールをロードする場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記次のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記次のモジュールをロードし、かつ、(iv)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御手段とを備えることを特徴とする情報処理装置である。
【００１１】
本発明は、一時的PCRs（transient PCR）を用いる環境で実行しているクライアントのリモート認証を実装する方法、システムおよびコンピュータプログラム製品に関する。
【００１２】
本発明では、tPCR（transient PCR） RIM証明書を用いる。クライアントは、認証用に用いるためのtPCR値を確立する基礎として、このtPCR RIM証明書を用いてスタートアップ時に自分自身を検証する。
【発明の効果】
【００１３】
この構成によると、情報処理装置は、複数のモジュールのうちで、どれがアクティブモジュールであるかを示す情報を管理し、そして、そのアクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成する。
【００１４】
したがって、当該情報処理装置は、アクティブモジュール全てに対応する蓄積プラットフォーム情報を生成することができる。ゆえに、ロード対象の第１モジュールの期待プラットフォーム情報と蓄積プラットフォーム情報とを比較して、検証を行うことによって、情報処理装置は、第１モジュールを正常にロードする前に、ロード対象の全てのモジュールを検証することができる。さらに、複数のモジュールのうちで、どれがアクティブモジュールであるかを管理することによって、１以上のモジュールが終了した後であっても、情報処理装置が、現在の信頼性の境界に従って（PCR値に対応する）蓄積プラットフォーム情報を動的に生成することができる。
【００１５】
（本願の技術的背景に関する情報）
２００８年１０月１０日に出願された、特願２００８－２６４５３０の日本出願の明細書、図面および特許請求の範囲等における開示の全体は、参照用として、本願に取り込ま

れる。
【００１６】
　また、２００８年１２月１７日に出願された、特願２００８－３２１５４０の日本出願の明細書、図面および特許請求の範囲等における開示の全体も、参照用として、本願に取り込まれる。
【図面の簡単な説明】
【００１７】
【図１】図１は、先行技術を表すブロック図である。
【図１Ａ】図１Ａは、先行技術を表すブロック図である。
【図２】図２は、本発明を表すブロック図である。
【図２Ａ】図２Ａは、本発明を表すブロック図である。
【図３】図３は、PCRアクセスが期待されるモジュールを表すブロック図である。
【図３Ａ】図３Ａは、プラットフォーム状態証明書のツリーにおける関係を表すブロック図である。
【図４】図４は、プラットフォーム状態証明書の構造体を示す。
【図５】図５は、一時的PCRをサポートするための拡張を含んだRIM証明書の構造体を示す。
【図６】図６は、拡張PSCツリーノードの構造体を示す。
【図７】図７は、コンポーネント・PSCマップの構造体を示す。
【図８】図８は、先行技術にかかるプラットフォーム状態証明書のサンプルを示す。
【図９】図９は、本発明の一形態にかかるプラットフォーム状態証明書および図8に基づくプラットフォーム状態証明書の２つのサンプルを示す。
【図１０】図１０は、アプリケーションの起動および終了中のモジュール間通信を示す。
【図１１】図１１は、PSCを拡張するフローチャートである。
【図１２】図１２は、PSCを拡張するフローチャートである。
【図１３】図１３は、PSCを拡張するフローチャートである。
【図１４】図１４は、取り消し前から取り消し後に拡張PSCツリーがどう変化するかを示したものである。
【図１５】図１５は、PSCの拡張を取り消すフローチャートである。
【図１６】図１６は、PSCの拡張を取り消すフローチャートである。
【図１７】図１７は、アプリケーションのリモート認証でのモジュール間通信を示す図である。
【図１８】図１８は、解決される問題を示す図である。
【図１９】図１９は、tPCRを示す図である。
【図２０】図２０は、情報処理装置による処理を示す図である。
【図２１Ａ】図２１Ａは、先行技術を表すブロック図である。
【図２１Ｂ】図２１Ｂは、先行技術を表すブロック図である。
【図２２Ａ】図２２Ａは、本発明を表すブロック図である。
【図２２Ｂ】図２２Ｂは、本発明を表すブロック図である。
【図２３Ａ】図２３Ａは、PCRアクセスが期待されるモジュールを表すブロック図である。
【図２３Ｂ】図２３Ｂは、プラットフォーム状態証明書のツリーにおける関係を表すブロック図である。
【図２４】図２４は、プラットフォーム状態証明書の構造体を示す。
【図２５】図２５は、一時的PCRをサポートするための拡張を含んだRIM証明書の構造体を示す。
【図２６】図２６は、拡張PSCツリーノードの構造体を示す。
【図２７】図２７は、コンポーネント・PSCマップの構造体を示す。
【図２８】図２８は、先行技術にかかるプラットフォーム状態証明書のサンプルを示す。
【図２９】図２９は、本発明の一形態にかかるプラットフォーム状態証明書および図8に

基づくプラットフォーム状態証明書の２つのサンプルを示す。
【図３０】図３０は、アプリケーションの起動および終了中のモジュール間通信を示す。
【図３１】図３１は、PSCを拡張するフローチャートである。
【図３２】図３２は、PSCを拡張するフローチャートである。
【図３３】図３３は、PSCを拡張するフローチャートである。
【図３４】図３４は、取り消し前から取り消し後に拡張PSCツリーがどう変化するかを示したものである。
【図３５】図３５は、PSCの拡張を取り消すフローチャートである。
【図３６】図３６は、PSCの拡張を取り消すフローチャートである。
【図３７Ａ】図３７Ａは、リモート認証の先行技術を表すブロック図である。
【図３７Ｂ】図３７Ｂは、リモート認証の先行技術を表すブロック図である。
【図３８Ａ】図３８Ａは、リモート認証に対する本発明のその他の実施形態を表すブロック図である。
【図３８Ｂ】図３８Ｂは、リモート認証に対する本発明のその他の実施形態を表すブロック図である。
【図３９】図３９は、アプリケーションがリモート認証している間のモジュール間通信を示す。
【図４０】図４０は、Quote情報記録の構造体を示す。
【図４１】図４１は、tPCRのみに対してアプリケーションがリモート認証している間のモジュール間通信を示す。
【発明を実施するための形態】
【００１８】
本発明の第１の形態における情報処理装置（装置１１２０）は、複数のモジュール各々に先立ってロードされるモジュールを示す期待プラットフォーム情報（tPCR値）を前記複数のモジュールごとに格納する格納手段（PSCデータベース1112）と、前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示すアクティブ情報を記録する管理手段（コンポーネント・PSCマップ1202）と、前記アクティブモジュールの次のモジュールをロードする場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報（PSCツリー1206の中にあるtPCR状態1604（図６））を生成し、(ii)前記次のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記次のモジュールをロードし、かつ、(iv)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御手段（OSサポート1200）とを備えることを特徴とする情報処理装置である。
【００１９】
図１８は、第１の形態の情報処理装置により解決される課題を説明する図である。
【００２０】
図１９は、tPCR(Transient PCR)を説明する図である。
【００２１】
図２０は、第１の形態の情報処理装置による処理を説明する図である。
【００２２】
なお、図１８では、遷移図1101Aが示される。また、図１９では、第１の形態の情報処理装置の処理を示す欄1101Baと、従来例の処理を示す欄1101Bbとが示される。
【００２３】
第１の形態の情報処理装置は、上述の通り、格納手段（PSCデータベース1112）と、管理手段（コンポーネント・PSCマップ1202）と、ロード制御手段（OSサポート1200）とを備える。これにより、図１９、図２０により示される機能が実現される。これにより、図

１８により説明される課題が解決される効果が得られる。

【００２４】

他方、第１の形態の情報処理装置と対比される従来例では、上記３つの手段のうちの一部又は全部がない。このため、従来例では、図１９、図２０により示される機能が実現されず、図１８の課題が解決される効果が得られない。

【００２５】

すなわち、上記の構成、作用、効果を有するか否かの点で、第１の形態の情報処理装置は、従来例に対して相違する。

【００２６】

なお、実施形態の説明の第１部分における装置1120（図１）は、後述される、実施形態の説明の第２部分における装置2120（図２１Ａ）に対応する。そして、装置1120のアプリケーション1100は、装置2120のアプリケーション2100に対応する。このように、装置1120の各構成は、その構成が例外的な構成である場合を除いて、装置2120における、その構成と同様の構成に対応する。以下では、このような、構成の間の対応についての詳しい説明は、適宜省略される。

【００２７】

なお、以下では、公知の文献に記載された技術事項については、その技術事項の詳しい説明が、適宜省略される。ここで、公知の文献は、先述された、”Trusted Computing Group’s (TCG) Mobile Trusted Module (MTM) documents TCG Mobile Reference Architecture version 1.0 12 June 2007”、”TCG Mobile Trusted Module Specification version 1.0 12 June 2007”、及びその他の文献を含む。

【００２８】

本発明の第２の形態における情報処理装置は、前記ロード制御手段が、モジュールが終了する場合に、前記モジュールはアクティブモジュールでないと示すように前記アクティブ情報を更新するよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００２９】

この構成によると、第１モジュールが終了する場合に、ロード制御手段は管理手段を制御してアクティブ情報を更新し、その終了モジュールがアクティブモジュールでないことを示す。

【００３０】

したがって、当該情報処理装置は、１以上のモジュールが終了した後であっても、拡張されたプラットフォーム情報を用いて検証することにより、現在ロードされているモジュールに正確に対応するアクティブモジュール全ての蓄積プラットフォーム情報を生成することができる。

【００３１】

本発明の第３の形態における情報処理装置は、さらに、前記アクティブモジュールの次のモジュールをロードする場合に、前記次のモジュールのダイジェスト値（ハッシュ）を算出し、期待ダイジェスト値と算出したダイジェスト値とを比較することで前記次のモジュールが有効であるかどうかを判定する判定手段（抽象化レイヤAPI1108）を備え、前記ロード制御手段は、前記次のモジュールが有効であると判定され、検証手段による検証が成功した場合には、前記次のモジュールをロードし、前記アクティブモジュールのうちの１つが終了した後にアクティブモジュールが少なくとも１つ残っている場合であって、前記少なくとも１つの残ったアクティブモジュールの次のモジュールをロードする場合には、前記少なくとも１つの残ったアクティブモジュールのダイジェスト値の算出をスキップするよう算出手段を制御し、前記少なくとも１つの残ったアクティブモジュールの判定をスキップするよう前記判定手段を制御することを特徴とする情報処理装置である。

【００３２】

この構成によると、情報処理装置は、アクティブモジュールのうちの１つが終了した後にアクティブモジュールが少なくとも１つ残っている場合であって、少なくとも１つの残ったアクティブモジュールの次のモジュールをロードする場合に、アクティブモジュール

のダイジェスト値の算出をスキップして、算出されたダイジェスト値を用いてアクティブモジュールを再び判定することをスキップする。

【００３３】

先行技術では、（この構成の蓄積プラットフォーム情報に対応する）PCR値をリセットまたは蓄積することしかできない。ゆえに、PCRを、それの以前の値に、モジュールが終了した後に戻すために、PCR値がリセットされなければならず、そこで、以前実行された各モジュールについて、それらのダイジェスト値を再計算して、それぞれのダイジェスト値をPCRに蓄積する。

【００３４】

一方、本発明の本形態では、情報処理装置が、どのモジュールがアクティブモジュールであるかを管理し、アクティブモジュールに対応する期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成する。ゆえに、アクティブモジュールのダイジェスト値の算出と、算出されたダイジェスト値を用いた判定とをスキップすることができる。これは、アクティブモジュールに対するこれらの処理が前もって行われており、それ以降はアクティブモジュールは変更されないことになっているからである。したがって、この構成により、第２モジュールのローディング処理を高速化することができる。

【００３５】

本発明の第４の形態における情報処理装置は、前記管理手段が、有向非巡回グラフを用いて前記アクティブモジュールを示す情報を管理することを特徴とする情報処理装置である。

【００３６】

この構成によると、管理手段は、有向非巡回グラフを用いてアクティブモジュールを示す情報を管理する。通常、複数のモジュールは他のモジュールに従属したり従属されたりしてロードされるので、有向非巡回グラフはこの関係を表現するのに適している。したがって、この構成によって、管理手段は、１以上のアクティブモジュールを容易に管理することができる。

【００３７】

本発明の第５の形態における情報処理装置は、前記ロード制御手段が、前記次のモジュールをロードする場合に、前記次のモジュールと当該次のモジュールの期待プラットフォーム情報とを示すノードを生成し、かつ、前記アクティブモジュールに対応するノードに、生成したノードが従属するように、前記生成したノードを前記有向非巡回グラフに付け加えるよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００３８】

この構成によると、情報処理装置は、次のモジュールをロードする場合に、次のモジュールと当該次のモジュールの期待プラットフォーム情報とを示すノードを生成し、かつ、アクティブモジュールに対応するノードに、生成したノードが従属するように、生成したノードを有向非巡回グラフに付け加えるよう、管理手段を制御する。すなわち、非巡回グラフは、どのアクティブモジュールが他のアクティブモジュールに従属しているかを正しく反映するものとなる。したがって、この構成により、管理手段は、アクティブモジュール間の従属関係を正確に管理することができる。

【００３９】

本発明の第６の形態における情報処理装置は、前記ロード制御手段が、前記次のモジュールがロードされて終了した場合に、前記次のモジュールを示すノードと当該ノードに従属するノード全てとを削除するよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００４０】

この構成によると、情報処理装置は、次のモジュールがロードされて終了した場合に、次のモジュールを示すノードと当該ノードに従属するノード全てとを削除するよう、管理手段を制御する。すなわち、終了したモジュールに対応するノードだけでなく、当該ノードに従属するノードも削除される。終了したモジュールのノードに従属するノードは終了

モジュールの子モジュールに対応しており、その親モジュールの終了時に子モジュールは終了する。したがって、この構成により、管理手段は、複数のモジュールのうちどれを本当にロード中なのかを正確に管理することができる。

【００４１】

本発明の第７の形態における情報処理装置は、前記ロード制御手段が、前記次のモジュールを示すノードが従属することになっている親ノードを検索して、各ノードの期待プラットフォーム情報を前記有向非巡回グラフのルートから前記親ノードへ蓄積することによって前記蓄積プラットフォーム情報を生成することを特徴とする情報処理装置である。

【００４２】

この構成によると、ロード制御手段は、次のモジュールを示すノードが従属することになっている親ノードを検索して、各ノードの期待プラットフォーム情報を前記有向非巡回グラフのルートから親ノードへ蓄積することによって蓄積プラットフォーム情報を生成する。この構成によって、情報処理装置は、次のモジュールに先立ってブートするアクティブモジュールを反映した蓄積プラットフォーム情報を間違いなく生成することができる。これは、有向非巡回グラフがアクティブモジュール間の従属関係を反映するからである。

【００４３】

本発明の第８の形態における情報処理装置は、前記ロード制御手段が前記蓄積プラットフォーム情報を所定期間の後に削除することを特徴とする情報処理装置である。

【００４４】

この構成によると、ロード制御手段は、蓄積プラットフォーム情報を所定期間の後に削除する。したがって、この構成により、プラットフォーム情報の存続期間を制限することによって改ざんは困難になる。

【００４５】

本発明の第9の形態における情報処理装置は、前記複数のモジュールが、それぞれが1以上のモジュールを含む第1モジュールグループと第2モジュールグループとを含み、前記情報処理装置は、さらに、前記第1モジュールグループのうち、どのモジュールをロードしたかを示す第1蓄積プラットフォーム情報を格納する登録手段を備え、前記格納手段は、さらに、前記第2モジュールグループのモジュールをロードする前に前記第1モジュールグループのモジュール全てをロードすべきことを示す第1期待プラットフォーム情報を格納し、前記ロード制御手段は、前記第1モジュールグループのモジュールに対して、(i)当該モジュールを検証し、(ii)検証が成功した場合には前記モジュールをロードし、かつ、(iii)前記モジュールがロードされた場合には前記モジュールのプラットフォーム情報を前記第1蓄積プラットフォーム情報に蓄積することによって前記第1蓄積プラットフォーム情報を更新し、前記第2モジュールグループのモジュールをロードする場合には、(i)前記第1期待プラットフォーム情報を前記登録手段に格納された第1蓄積プラットフォーム情報と比較することによって第1モジュールグループのモジュール全てが正常にロードされたことを検証し、前記アクティブモジュールの次の第2モジュールグループのモジュールをロードする場合であって、前記第1モジュールグループのモジュール全てが正常にロードされたことが検証された場合には、前記ロード制御手段は、(i)第2モジュールグループのモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記次のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記次のモジュールをロードし、かつ、(iv)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００４６】

この構成によると、ロード制御手段は、複数のモジュールのうち１つが正常にロードされるたびに蓄積プラットフォーム情報を削除し、複数のモジュールのうち１つをロードす

ることになるたびに蓄積プラットフォーム情報を生成する。したがって、蓄積プラットフォーム情報を改ざんから保護することができる。

【００４７】

本発明の第１０の形態における情報処理装置は、前記複数のモジュールが、それぞれが1以上のモジュールを含む第1モジュールグループと第2モジュールグループとを含み、前記情報処理装置は、さらに、前記第1モジュールグループのうち、どのモジュールをロードしたかを示す第1蓄積プラットフォーム情報を格納する登録手段を備え、前記格納手段は、さらに、前記第2モジュールグループのモジュールをロードする前に前記第1モジュールグループのモジュール全てをロードすべきことを示す第1期待プラットフォーム情報を格納し、前記ロード制御手段は、前記第1モジュールグループのモジュールに対して、(i)当該モジュールを検証し、(ii)検証が成功した場合には前記モジュールをロードし、かつ、(iii)前記モジュールがロードされた場合には前記モジュールのプラットフォーム情報を前記第1蓄積プラットフォーム情報に蓄積することによって前記第1蓄積プラットフォーム情報を更新し、前記第２モジュールグループのモジュールをロードする場合には、(i)前記第１期待プラットフォーム情報を前記登録手段に格納された第１蓄積プラットフォーム情報と比較することによって第１モジュールグループのモジュール全てが正常にロードされたことを検証し、前記アクティブモジュールに続く、第２モジュールグループの１のモジュールをロードする場合であって、前記第１モジュールグループのモジュール全てが正常にロードされたことが検証された場合には、前記ロード制御手段は、(i)第２モジュールグループのモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記１のモジュールの前記期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記１のモジュールをロードし、かつ、(iv)前記１のモジュールがロードされた場合には、前記１のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御することを特徴とする情報処理装置。

【００４８】

この構成によると、ロード制御手段は、(i)第１期待プラットフォーム情報を登録手段に格納された第１蓄積プラットフォーム情報と比較することによって第１モジュールグループのモジュール全てが正常にロードされたことを検証し、(ii)第１モジュールグループのモジュール全てが正常にロードされたことが検証された場合に、第２モジュールグループのモジュールに対して、生成、検証、ロードおよび制御を行う。

【００４９】

この構成により、情報処理装置は、第１モジュールグループに対する検証と第２モジュールグループに対する検証とを別々に行うことができる。したがって、当該情報処理装置は、第２モジュールグループのモジュールがいくつか終了しても、モジュール全ての検証を行う必要がない。

【００５０】

さらに、第２モジュールグループに対する処理は、第１モジュールに対する検証が失敗すれば、開始されない。したがって、たとえ第２モジュールグループに対する検証しか再度行わないとしても、第２モジュールグループのモジュールを、第１モジュールを含むモジュールが正常にロードされている信頼性のある環境上にロードすることができる。

【００５１】

本発明の第１１の形態における情報処理装置は、前記第２モジュールグループのモジュールが終了して第２モジュールグループのモジュールをロードする場合に、前記ロード制御手段は、前記第１期待プラットフォーム情報を前記第１蓄積プラットフォーム情報と比較することによって、前記第１モジュールグループのモジュール全てが正常にロードされ、かつ、終了していないことを検証し、検証が成功した場合には、前記第１モジュールグループのモジュールに対する検証をスキップすることを特徴とする情報処理装置である。

【００５２】
この構成によると、ロード制御手段は記第１期待プラットフォーム情報を第１蓄積プラットフォーム情報と比較することによって、第１モジュールグループのモジュール全てが正常にロードされ、かつ、終了していないことを検証し、検証が成功した場合には、第１モジュールグループのモジュールに対する検証をスキップする。
【００５３】
したがって、当該情報処理装置は、第２モジュールグループの終了されたモジュールを高速にリロードしたり、第２モジュールグループの他のモジュールを高速にロードすることができる。
【００５４】
本発明の第１２の形態における情報処理装置は、前記第１モジュールグループが、システム層のモジュールを含み、前記第２モジュールグループは、アプリケーション層のモジュールを含むことを特徴とする情報処理装置である。
【００５５】
この構成によると、第１モジュールグループは、システム層のモジュールを含み、第２モジュールグループは、アプリケーション層のモジュールを含む。
【００５６】
したがって、アプリケーション層のモジュールなど、頻繁に終了することになるモジュールに対する検証を、たとえそのモジュールが終了した後であっても、高速に再開することができる。
【００５７】
本発明の第１３の形態における情報処理方法は、情報処理装置用の情報処理方法であって、前記情報処理装置は、複数のモジュール各々に先立ってロードされるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示すアクティブ情報を記録する管理手段とを備え、前記情報処理方法は、前記アクティブモジュールの次のモジュールをロードする場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記次のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記次のモジュールをロードし、かつ、(iv)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御ステップを備えることを特徴とする情報処理方法である。
【００５８】
本発明の第１４の形態におけるコンピュータプログラムは、情報処理装置用の記録媒体に記録されたコンピュータプログラムであって、前記情報処理装置は、複数のモジュール各々に先立ってロードされるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示すアクティブ情報を記録する管理手段とを備え、前記コンピュータプログラムは、ロード制御ステップを前記情報処理装置に実行させるコンピュータプログラムであって、前記ロード制御ステップでは、前記アクティブモジュールの次のモジュールをロードする場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記次のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記次のモジュールをロードし、かつ、(iv)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティ

ブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御することを特徴とするコンピュータプログラムである。

【００５９】

本発明の第１５の形態における集積回路デバイスは、情報処理装置に用いられる集積回路デバイスであって、前記情報処理装置は、複数のモジュール各々に先立ってロードされるモジュールを示す期待プラットフォーム情報を前記複数のモジュールごとに格納する格納手段と、前記複数のモジュールのうち、どれがロード済みでまだ終了していないアクティブモジュールであるかを示すアクティブ情報を記録する管理手段とを備え、前記集積回路デバイスは、前記アクティブモジュールの次のモジュールをロードする場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記次のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記次のモジュールをロードし、かつ、(iv)前記次のモジュールがロードされた場合には、前記次のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御する、ロード制御手段を備えることを特徴とする集積回路デバイスである。

【００６０】

本発明の第１６の形態における情報処理装置は、当該情報処理装置は、サーバに接続され、前記ロード制御手段は、前記蓄積プラットフォーム情報を送信する要求を前記サーバから受信した場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、かつ、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記１のモジュールの前記期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)前記検証が成功した場合には、前記蓄積プラットフォーム情報を前記サーバに送信することを特徴とする情報処理装置である。

【００６１】

本発明の第１７の形態における情報処理装置は、前記ロード制御手段は、さらに、(i)前記蓄積プラットフォーム情報を生成するために、どの期待プラットフォームを用いるかを示す情報を生成し、(ii)前記情報に基づいて、前記蓄積プラットフォーム情報を検証するために用いる署名情報を生成し、(iii)前記署名情報が付加されている前記蓄積プラットフォーム情報を送信することを特徴とする情報処理装置である。

【００６２】

本発明の第１８の形態における情報処理装置は、請求項１６に記載の情報処理装置であって、前記ロード制御手段は、１のモジュールが終了する場合に、前記１のモジュールはアクティブモジュールでないと示すように前記アクティブ情報を更新するよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００６３】

本発明の第１９の形態における情報処理装置は、さらに、前記アクティブモジュールに続く１のモジュールをロードする場合に、前記１のモジュールのダイジェスト値を算出し、期待ダイジェスト値と算出したダイジェスト値とを比較することで前記１のモジュールが有効であるかどうかを判定する判定手段を備え、前記ロード制御手段は、前記１のモジュールが有効であると判定され、検証手段による検証が成功した場合には、前記１のモジュールをロードし、前記アクティブモジュールのうちの１つが終了した後にアクティブモジュールが少なくとも１つ残っている場合であって、前記少なくとも１つの残ったアクティブモジュールに続く１のモジュールをロードする場合には、前記少なくとも１つの残ったアクティブモジュールのダイジェスト値の算出をスキップするよう算出手段を制御し、前記少なくとも１つの残ったアクティブモジュールの判定をスキップするよう前記判定手

段を制御することを特徴とする情報処理装置である。

【００６４】

本発明の第２０の形態における情報処理装置は、前記管理手段は、有向非巡回グラフを用いて前記アクティブモジュールを示す情報を管理することを特徴とする情報処理装置である。

【００６５】

本発明の第２１の形態における情報処理装置は、前記ロード制御手段は、前記１のモジュールをロードする場合に、前記１のモジュールと当該１のモジュールの前記期待プラットフォーム情報とを示すノードを生成し、かつ、従属する当該モジュールに対応するノードに、前記生成したノードが従属するように、前記生成したノードを前記有向非巡回グラフに付け加えるよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００６６】

本発明の第２２の形態における情報処理装置は、前記ロード制御手段は、前記１のモジュールがロードされて終了した場合に、前記１のモジュールを示すノードと、前記１のモジュールを示す前記ノードに従属するノード全てとを削除するよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００６７】

本発明の第２３の形態における情報処理装置は、前記ロード制御手段は、前記１のモジュールを示す前記ノードが従属することになっている親ノードを検索して、各ノードの期待プラットフォーム情報を前記有向非巡回グラフのルートから前記親ノードへ蓄積することによって前記蓄積プラットフォーム情報を生成することを特徴とする情報処理装置である。

【００６８】

本発明の第２４の形態における情報処理装置は、前記ロード制御手段は前記蓄積プラットフォーム情報を所定期間の後に削除することを特徴とする情報処理装置である。

【００６９】

本発明の第２５の形態における情報処理装置は、前記ロード制御手段は、前記複数のモジュールのうち１つが正常にロードされるたびに前記蓄積プラットフォーム情報を削除し、前記複数のモジュールのうち１つをロードすることになるたびに蓄積プラットフォーム情報を生成することを特徴とする情報処理装置である。

【００７０】

本発明の第２６の形態における情報処理装置は、前記複数のモジュールは、それぞれが１以上のモジュールを含む第１モジュールグループと第２モジュールグループとを含み、前記情報処理装置は、さらに、前記第１モジュールグループのうち、どのモジュールをロードしたかを示す第１蓄積プラットフォーム情報を格納する登録手段を備え、前記格納手段は、さらに、前記第２モジュールグループのモジュールをロードする前に前記第１モジュールグループのモジュール全てをロードすべきことを示す第１期待プラットフォーム情報を格納し、前記ロード制御手段は、前記第１モジュールグループのモジュールに対して、(i)当該モジュールを検証し、(ii)検証が成功した場合には前記モジュールをロードし、かつ、(iii)前記モジュールがロードされた場合には前記モジュールのプラットフォーム情報を前記第１蓄積プラットフォーム情報に蓄積することによって前記第１蓄積プラットフォーム情報を更新し、前記第２モジュールグループのモジュールをロードする場合には、(i)前記第１期待プラットフォーム情報を前記登録手段に格納された第１蓄積プラットフォーム情報と比較することによって第１モジュールグループのモジュール全てが正常にロードされたことを検証し、前記アクティブモジュールに続く、第２モジュールグループの１のモジュールをロードする場合であって、前記第１モジュールグループのモジュール全てが正常にロードされたことが検証された場合には、前記ロード制御手段は、(i)第２モジュールグループのモジュールのうち、どれがアクティブモジュールであるかを前記アクティブ情報を用いて判断して、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記１のモジュール

の前記期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)検証が成功した場合には前記１のモジュールをロードし、かつ、(iv)前記１のモジュールがロードされた場合には、前記１のモジュールがアクティブモジュールであると示すように前記アクティブ情報を更新するよう、前記管理手段を制御することを特徴とする情報処理装置である。

【００７１】

本発明の第２７の形態における情報処理装置は、前記第２モジュールグループのモジュールが終了して第２モジュールグループのモジュールをロードする場合に、前記ロード制御手段は、前記第１期待プラットフォーム情報を前記第１蓄積プラットフォーム情報と比較することによって、前記第１モジュールグループのモジュール全てが正常にロードされ、かつ、まだ終了していないことを検証し、検証が成功した場合には、前記第１モジュールグループのモジュールに対する検証をスキップすることを特徴とする情報処理装置である。

【００７２】

本発明の第２８の形態における情報処理装置は、前記第１モジュールグループは、システム層のモジュールを含み、前記第２モジュールグループは、アプリケーション層のモジュールを含むことを特徴とする情報処理装置である。

【００７３】

本発明の第２９の形態における情報処理方法は、前記蓄積プラットフォーム情報を送信する要求をサーバから受信する受信ステップと、前記受信ステップにおいて前記要求を受信した場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、かつ、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記１のモジュールの期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)前記検証が成功した場合には、前記蓄積プラットフォーム情報を前記サーバに送信する、送信ステップとを含むことを特徴とする情報処理方法である。

【００７４】

本発明の第３０の形態におけるコンピュータプログラムは、のコンピュータプログラムであって、前記蓄積プラットフォーム情報を送信する要求をサーバから受信する受信ステップと、前記受信ステップにおいて前記要求を受信した場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、かつ、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記１のモジュールの前記期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)前記検証が成功した場合には、前記蓄積プラットフォーム情報を前記サーバに送信する送信ステップとを前記情報処理装置に実行させるためのコンピュータプログラムである。

【００７５】

本発明の第３１の形態における集積回路デバイスは、前記蓄積プラットフォーム情報を送信する要求をサーバから受信する受信部と、前記受信部が前記要求を受信した場合に、(i)前記複数のモジュールのうち、どれがアクティブモジュールであるかを、前記アクティブ情報を用いて判断し、かつ、前記アクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成し、(ii)前記１のモジュールの前記期待プラットフォーム情報を前記蓄積プラットフォーム情報と比較することによって、前記アクティブモジュールが正常にロードされたことを検証し、(iii)前記検証が成功した場合には、前記蓄積プラットフォーム情報を前記サーバに送信する、送信手段とを備えることを特徴とする集積回路デバイスである。

【００７６】

拡張動作を取り消しできるようにすることによって、アプリケーションがPCRの使用を希望する度に新たな証明書を作成するという問題を回避しながら、PCRおよびRIM証明書の手段で信頼性の境界を実行中のアプリケーションに拡張できるようにする方法が必要である。

【００７７】

さらに、既存のRIM証明書および管理ツールを活用するため、TCGが定義したようなRIM証明書の構造体を使用する方法も必要である。

【００７８】

さらに、２つのアプリケーションが気づかずに同じPCRを共有することによって生じる問題を防ぎつつ利用可能なPCR数を拡張する方法も必要である。

【００７９】

さらに、アプリケーションを起動および終了しながら、信頼性の境界を、動的、効果的かつ信頼的に拡大縮小できる方法も必要である。

【００８０】

本発明では、tPCRの状態について安全に問い合わせるRIM証明書をアプリケーションが使用できる、一時的なPCR（tPCR）の概念を実現することにより、当該技術分野の上記の制限に対処している。これらのtPCRの存続期間は、これらを使用するアプリケーションの存続期間ほど長くなく、必要であればより短い期間でもよい。本発明のtPCRは、物理的PCRとは全く異なるものであるため、tPCRを認識していないアプリケーションは、同じ環境で従来どおりに動作してもよい。

【００８１】

好ましい実施形態によると、本発明は、MTMを備えた装置において実現されるが、その他の同類の安全策をMTMの代用としてもよい。必要なキーコンポーネントとは、セキュア処理環境（SPE）と前述したPCRとプラットフォーム状態証明書（PSC）に署名するための検証キーとPSCのデータを認証および処理する方法である。先行技術で説明されたようなRIM証明書が、PSCの具体例である。

【００８２】

本発明の他の好ましい実施形態によると、アプリケーションが実行開始するときに、アプリケーションに関連付けられたPSCを記録し、そして、そのアプリケーションが終了するときに、記録されたPSCおよび従属しているもの全てによりそれらの拡張動作が取り消される。

【００８３】

他の好ましい実施形態によると、拡張動作が行われる場合には、有向非巡回グラフに拡張証明書が記録される。この有向非巡回グラフには、その拡張証明書が子として従属する、拡張済みの他の証明書全てが含まれる。この拡張動作は、拡張動作の有向非巡回グラフから、その拡張動作のレコードと全ての従属レコードとを削除することによって取り消される。

【００８４】

他の好ましい実施形態によると、拡張動作が必要な場合には、この証明書が従属する証明書は、すでに拡張済みの証明書の現在の有向非巡回グラフに基づき動的に評価される。

【００８５】

（第１の実施形態）

以下に、本発明の好ましい実施形態について説明する。

【００８６】

第１の実施形態は、一時的なPCRの利用をサポートするシステムに関し、この一時的なPCRは、証明書によって定められた値以外の定義済み存続期間を有する。前述の追加オペレーティングシステム機能およびPSCの信頼できる検証を行うことによって、SPE付き装置の開発者は、これらのtPCRを扱うシステムを作成できる。使用対象のtPCRについて記述するPSCを提供することによって、このような装置におけるアプリケーションの開発者は、信頼できる実行を柔軟な方法で提供するアプリケーションを作成できる。本発明によると、

アプリケーションは、いずれかのコンポーネントタイプとして定義されるが、独立型のプログラムや独立型のプログラム用プラグインモジュールやプラグイン用ヘルパーモジュールに限定されるものではない。

【００８７】

図１に、「TCG Mobile Reference Architecture」に記載された手段などのシステムのセキュアブートをサポートする場合における先行技術を示す。標準実行環境において、抽象化レイヤAPI1102を使用するアプリケーション1100が存在する。破線は、上方の前述した標準実行モードと、下方のセキュアモードとの間のセキュアモードインターフェース1106を示す。標準実行モードとは、ほとんどのコンピュータシステムが提供するような通常の実行環境である。セキュアモードでは、標準実行環境からはアクセスできないメモリ空間において、許可されたソフトウェアに限り実行可能なセキュア実行環境を提供する。本発明の好ましい実施形態では、セキュアモードしか分からないキーのプライベート部分を有するソフトウェアを暗号化することによって、この許可が行われるが、ホワイトリストや証明書などその他の技術を用いても構わない。この実行環境は、必要に応じて他のモジュールとともに、セキュアブートモジュールとセキュア処理環境1114とを有する。上記のセキュアモードインターフェース1106は、セキュアブートの信頼性の境界1104であって、そのライン以下全てが、信頼性のある環境内となる。そのラインは、「TCG Mobile Trusted Module Specification」で定義されたように、検証および拡張を行うセキュアブートプロセス中に構築される。セキュアモードの抽象化レイヤAPI1108は、サービスに対する、通常モードからの要求を取り扱い、更なる処理のために、抽象化レイヤ1110にこれらの要求をわたす。抽象化レイヤのタスクの１つはPSCデータベース1112を管理することである。また、別のタスクとしては、物理的PCRs1116の操作など、セキュア処理環境1114へのアクセスを含む、セキュアブートコンポーネント1113が提供するサービスの要求を取り扱うことがある。また、セキュアブートコンポーネント1113も、PSCデータベース1112へのアクセスを要求する。セキュア処理環境1114は、ハードウェアまたはソフトウェアのいずれで実現されてもよいが、好ましい実装では、「Trusted Computing Group specifications」で定義されたようなモバイルトラステッドモジュールであり、別の好ましい実装では、トラステッドプラットフォームモジュールである。ソフトウェア処理環境1113は、ハードウェアまたはソフトウェアのいずれか、あるいは双方の組み合わせで実現されてもよい。そして、セカンダリRIC（Runtime Integrity Checker）モニタ1118が存在する。このモニタのタスクには、改ざんが行われたと思われる場合に、アプリケーション1100を終了させることが含まれる。これは、好ましい実装では、PSCに格納された基準ハッシュに対して現在のアプリケーションのハッシュを検証することによって実現される。この検証は、特定のイベントが発生した時あるいは定期的な間隔で行われてもよく、これらのRICモニタに階層があっても構わない。これらのRICモニタは、１以上の子RICモニタの完全性をチェックする各RICモニタレベルを有する。プライマリRICは図示されていないが、そのタスクは、システム全体に対するマスター検証ルートとして、例えばハイパーバイザなどの図示されたシステム外で次のことを実行することである。すなわち、プライマリRICは、１以上のコンポーネントに対する完全性チェックを一定間隔で行い、PSCに格納された基準ハッシュに対してこれらのコンポーネントハッシュを検証する。このプライマリRICモニタのコンポーネントのうち１つは、セカンダリRICを備える必要がある。「TCG Mobile Reference Architecture」では、これをPRMVA、すなわちPrimary Runtime Measurement Verification Agentと称しており、このPRMVAは、セカンダリRICモニタに相当するSRMVA、すなわちSecondary Runtime Measurement Verification Agentを有している。図示されたもの全ては装置1120に配置される。

【００８８】

セキュアモードは、システムプロセッサの単独実行モード、オペレーティングシステムカーネルモード、セキュリティコプロセッサ、仮想マシン、ハイパーバイザ、完全性チェックされたメモリなど、当業者にとって周知な数々の技術により実現されてもよい。各コンポーネントは、本発明の新規性を有する教示と効果を逸脱しない範囲で、１以上の記載

されている技術またはその他の技術によって保護されても構わない。

【００８９】

当該先行技術のその他の実施形態では、セキュアモードインターフェース1106は、抽象化レイヤ1110とセキュア処理環境1114との間に配置される。当業者は、抽象化レイヤの実行に対してセキュアモードの完全な保護を必要とせず、上述したRICモニタが行う完全性保護のみを要するようなかたちで当該抽象化レイヤを実装してもよく、また、本発明を、完全性保護された環境に実装してもよいと分かるであろう。当該完全性保護は、ソフトウェアまたはハードウェア、あるいは双方の組み合わせで行われる。

【００９０】

図１Ａに、セキュアモードのサポートはないが、「TCG Specification Architecture Overview Revision 1.2 28 April 2004」に記載された手段などの独立したセキュア処理環境を有し、セキュアモードインターフェースが存在しない場合における先行技術のその他の実施形態を示す。セキュアモードインターフェースがないため、抽象化レイヤAPI1108は１つしか存在せず、セキュアブートコンポーネントの代わりに、トラステッド/セキュアブートコンポーネント1152が存在する。セカンダリRICモニタ1118は、セキュア処理環境1114以外のシステムにおける全てのコンポーネントをカバーできるように拡張され、このようにトラステッド/セキュアブートの信頼性の境界1150を確立する補助を行う。しかしながら、そうでない場合には、コンポーネントとそれらの役割は図１で説明した通りである。

【００９１】

図２に、図１の先行技術に基づいた本発明を示す。すでに述べたように、標準実行環境における抽象化レイヤAPI1102を使用するアプリケーション1100が存在する。破線は、上方の前述の標準実行モードと、下方のセキュアモードとの間のセキュアモードインターフェース1106を示す。上記のセキュアモードインターフェース1106は、先に述べたような実行前にコンポーネントを検証するプロセスによって構築されたセキュアブートの信頼性の境界1104であって、そのライン以下全てが信頼性のある環境内となる。OSサポート1200モジュールは、標準実行空間にあるが、信頼性のある境界内である。このモジュールは、どのアプリケーションが、起動前の検証にどの証明書を用いたかを示すマッピングを維持するためのコンポーネント・PSCマップ1202を管理する。セキュアモード内に、サービスに対する、通常モードからの要求を取り扱い、更なる処理のために抽象化レイヤ1110に、これらの要求をわたす抽象化レイヤAPI1108がある。抽象化レイヤのタスクの１つは、PSCデータベース1112を管理することであり、また別のタスクには、tPCRサポート1204を実装することがある。このサポートモジュールが、拡張されたけれども取り消されていないPSC全ての有向非巡回グラフを含む拡張PSCツリー1206データを維持する。さらに別のタスクは、物理的PCRs1116の操作など、セキュア処理環境1114へのアクセスを含む、セキュアブートコンポーネント1113が提供するサービスの要求を取り扱うことである。また、セキュアブートコンポーネント1113も、PSCデータベース1112へのアクセスを要求する。そして、セカンダリRIC（Runtime Integrity Checker）モニタ1118が存在し、そのタスクには、改ざんが行われたと思われる場合にアプリケーション1100を終了させることが含まれる。図示されたもの全ては装置1120に配置される。

【００９２】

先行技術と同様に、先行技術の好ましい実施形態では、セキュアモードインターフェース1106は、抽象化レイヤ1110とセキュア処理環境1114との間に配置される。当業者は、抽象化レイヤの実行に対してセキュアモードの完全な保護を必要とせず、上述したRICモニタが行う完全性保護のみを要するようなかたちで当該抽象化レイヤを実装してもよく、また、本発明を、完全性保護された環境に実装してもよいと分かるであろう。当該完全性保護は、ソフトウェアまたはハードウェア、あるいは双方の組み合わせで行われる。

【００９３】

セキュアモードは、システムプロセッサの単独実行モード、オペレーティングシステムカーネルモード、セキュリティコプロセッサ、仮想マシン、ハイパーバイザ、完全性チェ

ック済みメモリなど、当業者にとって周知な数々の技術により実現されてもよい。各コンポーネントは、本発明の新規性を有する教示と効果を逸脱しない範囲で、１以上の記載されている技術またはその他の技術によって保護されても構わない。

【００９４】

また、当業者は、tPCRサポート1204と拡張PSCツリー1206とを、セキュア処理環境1114に移動させたものが、その他の実施形態であると分かるであろう。さらなる実施形態は、抽象化レイヤ1110が、セキュアモードインターフェース1106の外側になって、tPCRサポート1204と拡張PSCツリー1206が、セキュア処理環境1114の内側になるように、これらの他の実施形態を組み合わせたものである。

【００９５】

図２Ａに、セキュアモードのサポートはないが、「TCG Specification Architecture Overview Revision 1.2 28 April 2004」に記載された手段などの独立したセキュア処理環境を有し、セキュアモードインターフェースが存在しない場合における、図１Ａに基づく本発明のその他の実施形態を示す。セキュアモードインターフェースがないため、抽象化レイヤAPI1108は１つしか存在せず、セキュアブートコンポーネントの代わりに、トラスティッド/セキュアブートコンポーネント1152が存在する。セカンダリRICモニタ1118は、セキュア処理環境1114以外のシステム内における全てのコンポーネントをカバーできるように拡張され、このようにして、トラステッド/セキュアブートの信頼性の境界1150を確立する補助を行う。しかしながら、そうでない場合には、コンポーネントとそれらの役割は、図２で説明した通りである。

【００９６】

図３に、本発明にかかるPCRの２タイプの利用パターンを示す。まず、通常アプリケーション空間におけるアプリケーション階層を示す。第１マッシュアップ1300は、第１プラグイン1302と第２プラグイン1304とから、サービスを利用する。これらのプラグインは、第１アプリケーション1306と第２アプリケーション1308とがそれぞれ所有するものである。どちらのアプリケーションも、抽象化レイヤAPI1102からサービスを利用する。次に、セキュアモードインターフェース1106の反対側に、抽象化レイヤAPI1108のセキュアモードのサポートが存在する。これは、抽象化レイヤ1110とやり取りする。抽象化レイヤのタスクの１つは、tPCRサポート1204を実装することである。このサポートモジュールにより、拡張され、かつ、取り消されていないPSC全ての有向非巡回グラフを含む拡張PSCツリー1206データが維持される。また別のタスクは、物理的PCR1116を操作する要求をセキュア処理環境1114に渡すことなどである。これは、ハードウェアまたはソフトウェアのいずれで実現されても構わない。

【００９７】

ところで、物理的PCR1116と一時的PCR1204を利用する典型的なパターンは以下の通りである。物理的PCRの読み取り1310動作は、随時有効である。セキュア処理環境1114でサポートされる全ての関数は、必ず物理的PCRを使用し、決して、一時的PCRを使用しない。しかしながら、もし、上述したように、tPCRサポートコンポーネント1204を、セキュア処理環境1114内に移動したならば、SPEがtPCRを使用することも可能である。物理的PCRへの書き込み1314は、先行技術で教示されたように、主にブート時に行われるが、さらに、アプリケーション空間から物理的PCRの書き込みが可能である1312。どの書き込み動作をアプリケーション空間から行うかを決定するのはシステム設計者または実装者次第である。一時的PCRに対し、読み取り1316と書き込み1318の双方は、アプリケーション空間において、排他的に行われるのが一般的である。一時的PCRの性質により、各実装者にはこれらのtPCRをどのように使用するか選択の自由度がある。しかしながら、図例のような場合には、第１マッシュアップ1300の開発者は、第１プラグイン1302と第２プラグイン1304の開発者達と調整して、どのtPCRをそれぞれ利用する予定か全員が把握できていると確認する必要があるであろう。

【００９８】

図３Ａに、拡張PSCツリー1206を示す。ノードの追加および削除方法については、後で

述べる。図示された有向非巡回グラフは、図２に示したようにtPCRサポートモジュール1204が記録したPSCを表している。図２に示されたモジュールとこの図の証明書との間には1対2の関係があり、信頼性の境界を拡張して、第１アプリケーション1306をカバーするには、先行技術で教示されているように、２つの証明書、つまり第１アプリケーション開始1350および第１アプリケーションロード1354が必要である。第２アプリケーション1308に対しては、第２アプリケーション開始1352および第２アプリケーションロード1356が必要である。第１プラグイン1302に対しては、第１プラグイン開始1358および第１プラグインロード1362、第２プラグイン1304に対しては、第２プラグイン開始1360および第２プラグインロード1364、第１マッシュアップ1300に対しては、第１マッシュアップ開始1366および第１マッシュアップロード1368が必要である。証明書間の矢印は、これらの証明書の従属関係を示している。従属関係は、各証明書の拡張時に各証明書が検出を期待するPCR状態により定義される。ツリーの各ノードの構造体は、図６において後ほど定義する。

【００９９】

図示したように、先行技術によると、各モジュールにはそのモジュールに関連付けられた２つの証明書があり、１つは、起動前にモジュールを検証するためにその親が使用し、もう１つは、期待された環境でモジュールが起動されたことを検証するためにモジュール自身が使用する。当業者は、モジュールごとに２つ以外の証明書を使用しても、本発明の範囲内であると分かるであろう。

【０１００】

図４に、プラットフォーム状態証明書1400（PSC）を示す。これは、そのプラットフォーム状態証明書が定める（物理的または一時的な）PCRで定義されるプラットフォームの状態と、プラットフォームの状態の検証がうまくいった時に、（物理的または一時的な）PCRへ拡張する値とを表す構造体である。これらの構造体を、PSCデータベース1112に格納してもよい。本構造体の第１フィールドは、PSC名1402である。この名前は、PSCデータベース1112へのPSCの格納、および、PSCデータベース1112からのPSCの取り出しに用いられるキーフィールドなので、固有でなければならない。また、好ましい実装では、この名前は、人間が読み取り可能な名前を表すバイト文字列である。アプリケーション開発者が使用する名前を決定してもよい。あるいは、プラットフォームの製造者が、アプリケーション開発者に名前を提供してもよい。当業者は、GUIDなどの他の表示を代わりに用いてもよく、PSC名を選択する方法は、他にもあると分かるであろう。次に、検証用PCR状態1404を表すエントリーリストが存在する。検証対象のPCRごとに、一対の値、つまりPCRインデックス1406およびPCR値1408が存在する。次に、拡張用PCR値が存在する。最初に拡張用PCRインデックス1410そして拡張用値1412である。最後に、セキュア処理環境1114の知る鍵により暗号化されたデータの残りのハッシュを表す暗号署名1414が存在する。この署名鍵は、セキュア処理環境1114へ安全に組み込まれたキーのプライベート部分か、署名PSCに使用するものであるとして上記の組み込まれたキーが直接または間接的に承認したキーのいずれかである。また、署名者は、プラットフォーム開発者またはアプリケーション開発者のエージェント、あるいは有効な署名鍵を発行した他の者でも構わない。

【０１０１】

本発明によると、プラットフォーム状態証明書1400を単に確認するだけでは、物理的PCRなのか一時的PCRなのか判断することはできない。どの種類のPCRをチェックすべきか決定するのは、それを用いる状況である。この利点の１つは、セキュアブート用に証明書を作成する既存のツールを、アプリケーション空間用に証明書を作成するために再利用できることである。

【０１０２】

好ましい実施形態では、検証用PCR状態1404の対リストを、テスト対象であるPCRインデックス1406を表すビットマップとPCR値1408の集合に対するハッシュとで置き換えてもよい。これは、RIM証明書に対して「TCG Mobile Trusted Module Specification」が定義した表示である。より複雑なチェックコードを費やせば、一時的なPCRを検証する証明書を

変更することなく、このような表示を用いることが可能である。しかし、好ましい実施形態では、図５に図示された一時的PCR用RIM証明書1500を用いる。この構造体と図４のプラットフォーム状態証明書とのフィールドの関係および追加フィールドについて詳しく述べる。

【０１０３】

ラベル1502は、PCS名1402に相当する。measurementPCRIndex 1504と、measurementValue 1506とは、拡張用PCRインデックス1410と、拡張用値1412とに相当する。検証用tPCR状態1518、およびPCRインデックス1514とtPCR値1516とを含む対リストは、プラットフォーム状態証明書1400で定義されたフィールドと類似のものである。検証用tPCR状態1518を一時的PCR用RIM証明書1500に関連付けるためには、extensionDigestSize 1508とextensionDigest 1510のフィールドを使用することが必要である。extensionDigestSize 1508は、extensionDigest 1510のバイトサイズであり、extensionDigest 1510には検証用tPCR状態1518の構造体のハッシュが含まれる。state 1512フィールドに設定されたビット数は、テーブルのペア数を示すので、サイズインジケータを格納する必要はない。当業者は、tPCRインデックス系列など、tPCR値1516フィールドに所定順序がある場合には、tPCRインデックス1514フィールドを格納する必要さえないと分かるであろう。

【０１０４】

図６に、拡張PSCツリーノード1600を示す。これは、証明書１つの拡張を記録する構造体である。このノードの構造体では、図３Ａに図示したような各ノードの内容、つまり項目1350～1368を詳しく説明している。拡張PSCツリー1206は、当該技術分野において周知の技術を用いて有向非巡回グラフを実装する。例えば、ブーストC++ライブラリには、ブーストグラフライブラリが含まれている。これは、上述した有向非巡回グラフなど様々なグラフの作成および操作をサポートする。このように、拡張PSCツリーノード1600は、グラフの各頂点に関連付けられる。拡張PSC名1602は、拡張されたプラットフォーム状態証明書1400のPSC名1402のフィールドである。tPCR状態1604は、このノードにおける現在の一時的PCR状態のキャッシュであり、このノードの先祖に当たる拡張済みのPSCから算出される。tPCR状態は、tPCRインデックス1606とtPCR値1608との対リストで構成される。当業者は、このデータの別の表現、例えば、用いたtPCRを表すビットマップでtPCRインデックス1606フィールドを置き換えるなどが存在すると分かるであろう。

【０１０５】

図７に、OSサポートモジュール1200によって維持されたコンポーネント・PSCマップ1202を示す。このコンポーネント・PSCのマップ1202には、コンポーネントをPSCへマッピングするリスト1700が含まれる。このリストの各エントリーには、コンポーネントID1702と、拡張PSC名1704、つまりアプリケーションの起動前に拡張されたプラットフォーム状態証明書1400のPSC名1402フィールドとが含まれる。Windows（登録商標）ベースのプラットフォームにおける本発明の好ましい実施形態では、コンポーネントID1702は２つのフィールドから構成されている。第１フィールドは、プロセスID1706である。このプロセスIDは、GetCurrentProcessId()などのWin32APIによって決定されるような、コンポーネントが属するプロセスを一意的に表す識別子をもつ。第２フィールドはモジュールハンドル1708である。単独で実行可能なコンポーネントに対しては、このフィールドを必ず0に設定する。リンクライブラリとして実装されたコンポーネントに対しては、第１パラメータのDllMainエントリーポイントに渡されるように、ライブラリ用のHMODULEがこのフィールドに含まれる。この構造体の利用法については後で述べる。

【０１０６】

「TCG Mobile Reference Architecture」によると、PCR0は、基礎を成すハードウェアプラットフォームの特徴を表す値を保持する。PCR1は、信頼性のルートを表す値を含む。PCR2は、エンジンロードイベントを表す値を含む。PCR3～6および8～12は、専有手段を含む。PCR13～15はアプリケーションが自由に使用できる。セキュアブートが正しく行われたことをPCR0、1および2が期待通りに示すかどうかのテスト、PCR13が0に設定されたかどうかのテスト、そして、全てが正しければ新たな値をPCR13に拡張することをアプリケー

ションプログラマが希望したとする。図８に、先行技術にかかる、「第１アプリケーション開始」と名付けられたプラットフォーム状態証明書1400のサンプルを示す。証明書の名前は1800に記録される。上述したように、検証用PCR状態1404は、チェックするPCRインデックスとPCR値との対を4つ含み、1802、1804、1806、1808、1810、1812、1814、1816と番号付けする。1802はPCR0を示し、1804の＜ハードウェアプラットフォーム＞は、基礎を成すハードウェアプラットフォームを表す公表値を示す。1806はPCR1を示し、1808の＜信頼性のルート＞は、基礎を成す信頼性のルートを表す公表値を示す。1810はPCR2を示し、1812の＜エンジンロードイベント＞は、PCR2に拡張されたロードイベント値から算出された合成ハッシュを表す公表値を示す。1814はPCR13を示し、1816の0の値は、PCR13がまだ初期状態であるという予測を示す。次に、1818へ拡張するPCRが存在し、そして、そのPCRへ拡張する値1820 が存在する。

【０１０７】

先行技術にかかる図８の証明書に関する問題として次のものがある。他のアプリケーションがPCR13を使用してしまった場合、検証用PCR状態1404はもはや正確ではなくなってしまうという問題がある。アプリケーションが終了して再起動する場合、1818と1820に定義した先に拡張された値によってPCR13は0以外の状態に設定されてしまい、検証用PCR状態1404はもはや正確ではなくなってしまうという問題がある。

【０１０８】

しかしながら、本発明では、図８の証明書と類似の証明書は、アプリケーション自身と同じ時間または別々の時間で対象装置に展開するためにアプリケーション開発者によって、図９に示すような２つの証明書に分けられる。なぜ分けるかというと、既存の証明書により２つの異なるPCRセットを検証するためである。第１のセットにはセキュアブートプロセスの結果、つまり既知の不変な結果が含まれ、第２のセットには動的状態、つまりアプリケーションレベルの状態が含まれる。図８において、項目1802および1804は、ハードウェアプラットフォームを表す既知のセキュアブートPCR0の値を指している。項目1806および1808は、信頼性のルートを表す既知のセキュアブートPCR1の値を指し、項目1810および1812は、エンジンロードイベントを表す既知のセキュアブートPCR2の値を指している。また、項目1814および1816は、期待される前提条件を表す、ブート後のPCR13の望ましい値を指している。このように、アプリケーション開発者は、セキュア処理環境が管理する物理的PCR、すなわち本例ではPCR0、1および2をテストするために用いられる片方の証明書に既知のセキュアブートPCRの値を置くことによって、図８の単一証明書を図９に示した２つの証明書に分けてもよい。第２の証明書は、アプリケーション空間の一時的PCR、すなわち本例ではPCR13用に用いられる。

【０１０９】

「第１アプリケーション開始（セキュアブート）」1900と名付けられた第１証明書によって、セキュア環境が正しいことを確保するため、セキュアブートプロセスがセットアップした物理的PCR（PCR0 1802、PCR1 1806およびPCR2 1810）はテストされる。ここで、拡張用PCRは、拡張がないことを示すために-1 1902に設定され、拡張用値1904はゼロという名目上の値である。この証明書は検証のみに使われる。そして、アプリケーションレベルでは、図３に示すように物理的PCRへの書き込みをやめ、一時的PCRを用いることによってより柔軟性が高まり、前述した当該技術分野における課題を回避することができる。「第１アプリケーション開始（一時的）」1920と名付けられた第２証明書によって、ゼロ 1816である一時的PCR13 1814はテストされ、同じレジスタ1818および1820に値を拡張する。tPCRのインデックス13を選択したことに特に意味はない。先行技術の状況とは異なり、tPCR0またはtPCR99が用いられやすいであろう。

【０１１０】

図１０に、シーケンス図を示す。この図では、アプリケーション起動時には図９の２つの証明書、つまり一方が物理的PCRをテストする証明書であり、もう一方が、一時的PCRをテストする証明書を用い、アプリケーション終了時には、「第１アプリケーション開始（一時的）」1920の第２証明書を用いて、値をtPCRに拡張してからその拡張を取り消すとい

う本発明にかかるイベントシーケンスを示す。相互に作用する６つのオブジェクト、11000、11002、11004、11006、11008および11010を図示している。まず、tPCR13 11000は、この例の状況における一時的PCR13の状態を表す。図６で説明したように、tPCRは、特定のメモリ位置というよりむしろ拡張PSCツリー1206のノードごとに記録されるが、本例においては理解しやすいよう、tPCR13 11000をそのような位置にあるかのごとく表す。次に、tPCRサポート11002が存在する。これは、tPCRを参照するPSCの取得と現在のtPCR状態の検証との処理を行い、有効であれば、証明書は拡張されたと記録する。SPE11004は、先行技術にかかるセキュア処理環境である。好ましい実施形態では、それはMTMである。抽象化レイヤ11006は、通常モードのアプリケーションからの要求を取り扱い、要求を他のモジュールにわたす。OS11008はオペレーティングシステムであり、ここでは、一時的PCRを正確にアップデートするようなアプリケーションの起動および終了の処理に関するシステムである。そして、アプリケーション11010は、任意のタスクを行うサンプルアプリケーションである。図１０のシーケンス図のプロセスは入れ子されてもよいので、アプリケーション11010は、例えば、アプリケーション実行11042が11014から始まるイベントシーケンスに従う別のアプリケーションの開始を含むような、PSCによって保護された他のアプリケーションの開始を要求するなどのアプリケーションであっても構わない。図を単純化するため、エラー処理については図示しないが、本発明から除外されるものではない。

【０１１１】

まず最初に、tPCR13 11000は、0の値11012で始まる。本発明では、拡張PSCツリー1206のルートは、全tPCRを０に設定させた状態で始まるというのがルールである。当業者であれば、セキュアブートの完了後に物理的PCRの値でtPCRを初期化するというような他の初期値候補が可能だと分かるであろう。OS11008は、要求を検出してアプリケーション11010を起動するので、まず、起動しようとするアプリケーションによってどのPSCが用いられるのかを決定する11014。Windowsベースのオペレーティングシステムにおける好ましい実施形態では、実行ファイルに組み込まれたカスタムアセンブリが、使用する２つのPSCを特定する。そのアプリケーションは、改ざんから保護するためにマイクロソフトのストロングネームツールを用いて署名される。次に、11014で特定されたPSC、この図では「第１アプリケーション開始（セキュアブート）」および「第１アプリケーション開始（一時的）」を抽象化レイヤ11006に要求する11016、11018。そして、これら２つのPSCの検証が、抽象化レイヤAPIであるAL_VerifyPSCsAndExtendtPCRを呼び出す11020ことによって行われる。このAPIは、システムが開始したいアプリケーションを検証する２つのPSC、つまり、図９で示したような物理的PCR用のPSCおよび一時的PCR用のPSCを引数とする。まず、図の11024で表されたPSC「第１アプリケーション開始（セキュアブート）」を引数とした、SPEのAPIであるSPE_VerifyPSCStateを呼び出す11022。先行技術によれば、これによって、PSC自身のフォーマットおよび署名のチェックが行われ、そして、PSCの検証用PCRが物理的PCRの現在の値と一致することを検証する。先行技術によれば、PSCがプラットフォームに渡されると、セキュア処理環境1114に組み込まれたキーか、この組み込まれたキーによって正当であると検証されることのできるキーのいずれかでPSCは署名される。そして、有効であれば、セキュア処理環境1114が生成かつ安全に格納した他のキーで再び署名され、証明書はPSCデータベース1112に格納される。

【０１１２】

本発明の好ましい実施形態によると、物理的PCRをチェックするPSCはオプショナルなので、ステップ11016および11018を省略してもよい。その他の好ましい実施形態によると、物理的PCRのチェック用のPSCがアプリケーション全てについて同じである場合、物理的PCR用のPSC１つを２つ以上の異なる一時的PCRのPSCが用いてもよい。

次に、SPEの他のAPI、すなわち、図の11030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定したSPE_VerifyPSCを呼び出す11026。先行技術によると、これによって、物理的レジスタに対してPCRの設定を検証せずとも、PSC自身のフォーマットおよび署名のチェックが行われる。そこで、tPCRサポートモジュール、すなわち、図の11030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定した

APIであるTPCR_VerifyPSCAndExtendを呼び出す11028。このAPIの最初のタスクは、拡張PSCツリー1206内の既存状態に対応する検証用PCR状態1404をチェックすることにより、PSCを拡張できることを検証する11032ことである。この動作の詳細については後で述べる。検証が正常に完了した時点で、このPSCにおける動作の成功が、拡張PSCツリーの正しい位置へその表示を付け加える11034ことによって記録される。この動作の詳細については後に述べる。このPSCをツリーへ付け加えたことによる結果の１つは、tPCR13 11000、つまり拡張対象のレジスタが、以前の値、この場合は０、とPCRから拡張するための値1030、つまり0xABCD1234とを連結したもののハッシュに設定されたレジスタの値を有することである。これは、合成ハッシュと呼ばれ、数式的には、tPCR13 = SHA-1（0xABCD1234と連結されたtPCR13）と書き込まれる。この動作は11036の構文 (+)=で表される。このように、環境が期待された状態であることを検証し、成功したことの記録が生成され、そして、制御がオペレーティングシステムに戻る。先行技術によると、11020におけるPSC検証の前の、更なる安全性の評価として、アプリケーションのハッシュを算出し、拡張するPSCに格納された基準値と比較する。本発明では、この値は、好ましい実施形態において0xABCD1234と表された、拡張用の値として、PSC「第１アプリケーション開始（一時的）」11030に格納される。しかしながら、このステップは図から省略されている。

【０１１３】

アプリケーションを起動する際に、OSは、そのアプリケーションのプロセスIDを取得し、コンポーネント・PSCマップ1202に、この識別子と一時的レジスタ用の対応PSC、つまりPSC「第１アプリケーション開始（一時的）」とを記録する11038。マイクロソフトのWindows環境における好ましい実施形態では、本プロセスは、http://www.codeproject.com/KB/system/InterceptWinAPICalls.aspxのコードプロジェクトにおけるAndriy OriekhovのIntercepting WinAPI callsに記載されたようなプロセス作成プロセスを途中でインターセプトすることによって実装される。取得したプロセスハンドルは、プロセスID1706に変換され、そのフィールドに設定される。そして、モジュールハンドル1708を０に設定する。コンポーネントがダイナミックリンクライブラリである場合、http://www.lenholgate.com/archives/000369.htmlの/* Rambling comments... */におけるLen HolgateAPIのWhy does windows hold the loader lock whilst calling DllMain?に記載されたように、LoadLibrary()およびFreeLibrary()コードをフックし、DllMain()への呼び出しをトラップする。このトラップを設けたまま、プロセスID1706を現在のプロセスIDに設定し、モジュールハンドル1708をDllMain()の第１引数に設定する。

【０１１４】

アプリケーションは起動されると11040、PSCに関連付けられた他のアプリケーションを起動またはtPCR自体を参照する他のPSCを拡張したとしても、プログラムされたように実行し続ける11042。そして、アプリケーションを閉じるとユーザが選択する、クラッシュする、または、アプリケーションを強制的にシャットダウンするセカンダリRICモニタ（この図では図示していない）が改ざんを検出するといういずれかの原因で、アプリケーションは終了する11044。

【０１１５】

アプリケーションが終了するので、OSはアプリケーションのプロセスIDを取得し、そのプロセスIDと0であるモジュールハンドルとを用いて、コンポーネントID1702を作成する。このコンポーネントIDを用いてコンポーネント・PSCマップ1202を検索し、アプリケーションを起動するのに用いられたPSCを見つける11046。これにより、PSC「第１アプリケーション開始（一時的）」11030が返されるので、取り消しを行うため、OSは、抽象化レイヤ11006のAPIであるAL_UndoPSCExtendを、PSCを引数として呼び出す11048。コンポーネントがダイナミックリンクライブラリである場合、上述したようにFreeLibrary() APIをフックするので、そのルーチン内で現在のプロセスIDを問い合わせ、FreeLibrary()のパラメータからモジュールハンドルを取得し、そしてこれら２つのデータ項目を用いて、コンポーネントID1702を作成する。このコンポーネントIDを用いてコンポーネント・PSCマップ1202を検索し、ライブラリを起動するのに用いられたPSCを見つける。前述と同様に

、SPEの他のAPI、すなわち、図の11030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定したSPE_VerifyPSCを呼び出す11026。先行技術によると、これによって、物理的レジスタに対してPCRの設定を検証せずとも、PSC自身のフォーマットおよび署名のチェックが行われる。そこで、tPCRサポートモジュール11002、すなわち、図の11030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定したAPIであるTPCR_VerifyPSCAndExtendを呼び出す11050。このAPIの最初のタスクは、拡張PSCツリー1206にそのPSCがすでに存在するかどうか確認することによって、PSCが拡張されたことを検証する11052ことである。検証が正常に完了した時点で、これは、11028における拡張動作を取り消すことが可能であることを意味する。これは、拡張PSCツリー11034から、PSCを表すノードおよびそれに従属する他のノード全てを削除することによって実現される。この動作の詳細は後で述べる。ツリーからこのPSCを削除することの結果の１つは、tPCR13 11000、つまり取り消す対象のレジスタが効率よくその状態を0にリセットされる11056ことである。このように、環境が期待した状態であることを検証し、拡張PSCツリーからノードを削除する11034ことによって先に拡張された動作を取り消し、制御がオペレーティングシステムに戻り、本システムは他の動作をすぐに実行することができる。当業者であれば、これらの他の実行動作の１つとして、終了したアプリケーションを再起動することもあると分かるであろう。tPCR13が11056において00…00、つまり11012で示されたtPCR用の開始値にリセットされた後は、アプリケーションの開始PSC11030の検証は２回目も正常に再実行されるので、本発明によると、アプリケーションを再起動することができる。

【０１１６】

図１０に、実行ファイルの起動および終了時に信頼性の境界を動的に拡張かつ縮小するシーケンス図を示す。注釈を用いて、好ましい実施形態に沿ったダイナミックリンクライブラリに対する同様のタスクをWindows Portable Executable形式のモジュールでどのように行うかを示す。Java ArchiveモジュールやJARなどのPortable Executableベースでないモジュール形式に対しては、モジュールをロードかつアンロードするエンジンによって同様の手法が用いられるか、代わりに、モジュール自身によって抽象化レイヤ11006への明示的な呼び出しが行われるかして、信頼性の境界を拡張かつ縮小してもよいと当業者は分かるであろう。先行技術によると、JARは、Javaバイトコードベースのモジュールだけでなく、例えばECMAスクリプト（Javaスクリプト）など、その他の言語モジュールを含んでもよい。これらを、http://java.sun.com/j2se/1.3/docs/tooldocs/win32/jarsigner.htmlのSunからjarsignerツールを用いて署名し、その署名を、http://java.sun.com/j2se/1.4.2/docs/api/java/util/jar/JarFile.htmlに記載されたようなjava.util.jar.JarFileクラスを用いて検証してもよい。この場合、好ましい実施形態において、図７に示されたモジュールハンドル1708は、コンポーネントを含むJARファイルを参照するハンドルである。

【０１１７】

図１１に、tPCRサポートモジュール11002のAPI、TPCR_VerifyPSCAndExtend 11028の詳細に関するフローチャートを示す。本関数は、検証および記録するためのPSCをパラメータとしてわたして11100で開始し、次の図１２で示す、拡張PSCツリーの各ノードに対してtPCR状態を算出する11102サブルーチンを呼び出す。次に、現在のtPCR状態とソリューションリストの変数とを、NULLに初期化する11104。これら２つの変数の利用法については図１３で説明する。そして、渡されたPSCのtPCR値に、現在の拡張PSCツリーに記述された状態から達することができることを検証するサブルーチンが呼び出される11106。戻りコードをテストすることで、本関数が、拡張PSCツリー内で、渡されたPSCが保持する検証用の状態にtPCRを設定したノードを１以上見つけたかどうか分かる11108。この設定された親が見つからなかった場合、プロセスは、呼び出しルーチンへ拡張できなかったことを示すエラーコードを返す11110。見つかった場合は、渡されたPSCを、ソリューションリストに記述されたノードに設定した先のPSCを有する拡張PSCツリーに付け加え11112、プロセスは、呼び出しルーチンにサクセスコードを返す11114。

【０１１８】
図１２に、図６における拡張PSCツリーの各ノードのtPCR状態1604をどのように算出するか説明するフローチャートを示す。当該フローチャートは引数なしに呼び出され11200、行きがけ順に拡張PSCツリーの走査を行う11202ことで処理が開始する。この走査は、ツリーのノードを続く処理にとって望ましい順序で収集するためにルートから始まる。好ましい実装では、ブーストグラフライブラリ関数breadth_first_search()を用いてこれらのノードを収集する。次に、記録されている各ノード11204に対し、深さ優先走査によって現在のtPCR状態1604をNULLに設定する11206。セキュアブート完了の際に定義されたtPCRの開始値、好ましい実施形態では０の値、に初期化されたtPCRを検証する証明書の特別処理が必要なので、現在の証明書に格納された検証用PCR状態1404のtPCR対ごとに、その値をセキュアブート完了の際に定義されたtPCR初期値、好ましい実施形態では０の値と照合する11210。それらが等しい場合は、tPCRのインデックスおよび値の対をこのノードのtPCR状態に付け加え11212、次のtPCRに対してループを続ける。そうでない場合は、ノードのtPCR状態を付け加えずにループを続ける。各tPCRをチェックすると、本関数は、現在のノードの親それぞれに対するループへと移行する11214。親ノードが拡張PSCツリーのルートノードであれば11216、前回のループがこの特殊ケースを処理したため、何も行われない。そうでない場合には、親ノードのtPCR状態1604を問い合わせ、コピーする11218。親の拡張PSC名1602により参照されるPSCによって定義された拡張動作が親の状態のコピー11220において行われ、その結果のtPCR状態が現在のノードのtPCR状態に付加される11222。この拡張PSCツリー構築方法により、２つの親が同じtPCRに対して異なる値を指定する事態は決しておこらない。そのため、当業者は、この状況をチェックする必要はないが、検証目的で実装してもよいと分かるであろう。tPCR状態の収集は、前述したように、全ての親に対して繰り返され、終了すれば、走査中の次のノード11204が処理される。このように各ノードが処理されると、各ノードのtPCR状態を返してプロセスは終了する11224。

【０１１９】
当業者は、例えばbfs_visitor’s examine_vertex()でステップ11204から11222を実行するなど、上記アルゴリズムを実行する別の方法が存在するので、ノードの別々のリストの必要性はなくなると分かるであろう。さらに、この関数は、PSC関連の動作が行われるたびに呼び出されるが、その値をキャッシュして再計算にかかる労力を削減してもよい。

【０１２０】
図１３に、ノードごとのtPCR状態が算出された後、あるPSCに定義された状態へtPCRを設定する拡張済みのPSC全てのリストを見つけることによって、どのようにしてそのPSCを検証するか説明するフローチャートを示す。この図で説明するルーチンは、再帰的なルーチンであることに注意されたい。このルーチンへのエントリーポイントは、引数として、リスト形式の検証用tPCR状態と、現在のtPCRの一致の状況と、tPCR状態によってPSCの親だと判明した拡張PSCツリーのノードのリストとをとる11300。ソリューション方法の概要は、tPCRのインデックスおよび値の対ごとに、望ましい値を現在のtPCRに拡張する証明書であり、かつ、このソリューションの一部である別のtPCRに拡張する他の証明書と互換性をもつ証明書を見つけようと試みることである。候補が見つかれば、ルーチンを再帰的に呼び出して、PSCの一部である別のtPCRを一致の状態へ拡張する他の証明書を見つける。

【０１２１】
最初のステップは、一致するtPCRのリストをチェックすることである。これが空である場合11302、ルーチンはリストの最後に正常に再帰されたので、呼び出し側に成功を示すためFOUND値を返す11304。そうでない場合は、tPCRリストの先頭を削除し、これを現在のtPCRとして用いて親の証明書を見つけようと試みる11306。ソリューションリストにまだ割り当てられていない拡張PSCツリーの各ノードを親になる候補として選択する11308。先行技術に従ってこれらのノードをどのように取得できるかは図１２の説明で示した。まず、２つの構造体において一致するtPCRインデックスが同じtPCR値であることを検証することによって、このノードのtPCR状態を、渡された現在の一致tPCR状態と互換性があるかどうかチェックする11312。値が一致しなければ11316、ルーチンは、拡張PSCツリーの次の

ノードへ移行する11308。一致する場合には、ノードに格納された拡張PSC名1602を用いて検証用ノードのPSCを検索し11314、拡張用PSCのインデックス1410を11306で取り出した現在のtPCRのインデックスと比較する。インデックスが一致しない場合11316、ルーチンは、拡張PSCツリーの次のノードへ移行する11308。インデックスが一致する場合は、親ノード候補が見つかったので、このノードをソリューションリストに付け加え11318、このノードの状態を拡張して現在のtPCRリストにマージする。tPCR検証ルーチンを、短縮したtPCRリストと現在のtPCR状態とソリューションリストとを引数にして、再帰的に呼び出す11332。再帰的呼び出しが成功すれば11324、呼び出し側にその成功を示すためFOUND値を返す11326。失敗した場合は、現在のノードをソリューションリストから削除して11320での状態のマージを取り消し11328、そして、ツリーの次のノードを調べるためにループを続ける。全てのノードが調べられ、一致がなかった場合は、NOTFOUNDを返す11310。

【０１２２】

図１４に、取り消し前と取り消し後のサンプル拡張PSCツリー1206の状態を示す。取り消し前の状態11400は図３Ａで述べたとおり、図３のモジュールツリーから構築された状態である。ところで、図１０で示されたように、ユーザからの作用、プログラムバグ、または改ざん検出により第１プラグインが終了すれば、オペレーションシステムは第１プラグインの終了を検出して、「第１プラグイン開始」証明書1358は起動時にテストされたPSCであったと判断するので、証明書の拡張を取り消す必要がある。「第１プラグイン開始」1358とともに、それに従属する全ての証明書、すなわち、「第１プラグインロード」1362、「第１マッシュアップ開始」1366および「第１マッシュアップロード」1368も拡張PSCツリー1206から削除しなければならない。その結果、取り消し後の状態11402の拡張PSCツリー1206となる。

【０１２３】

図１５に、拡張プロセスをどのようにして取り消すか説明するフローチャートを示す。本関数は、引数として、取り消し対象PSCをとる11500。まず、本関数は、上記の図１２で示された、拡張PSCツリーのノードごとにtPCR状態を算出する11502サブルーチンを呼び出す。次に、対象PSCのPSC名1402とツリー内の各ノードの拡張PSC名1602との一致を探すことで、拡張PSCツリー内で対象PSCへの参照を検索する11504。一致するノードが見つからない場合11506、エラーコードを呼び出し側に返して取り消すことができなかったことを示す11512。次に、対象PSCの検証用PCR状態1404を見つかったノードのtPCR状態1604とを比較する。それらの状態が等しくない場合11510、エラーコードを呼び出し側に返して取り消すことができなかったことを示す11512。状態が等しい場合は、見つかったノードとそれに従属するもの全てを削除する関数を呼び出し、その関数がサクセスコードを呼び出し側に返す11516。

【０１２４】

図１６に、取り消し処理が拡張PSCツリーからどのようにしてノードを削除するか説明するフローチャートを示す。本関数は、引数として、ツリーから削除するノードをとる11600。まず、このノードの子PSC全てに対してループし11602、再帰的に自分自身を呼び出してその子それぞれを順に削除する11604。全ての子が削除された時点で、ノード自身を削除し11606、本関数はリターンする11608。このようにして、終了中のモジュールおよびそれに従属する信頼性のあるモジュール全てをカバーする、以前の拡張処理によって確立された信頼性の境界は、終了する必要のないモジュールをカバーしたまま、装置のアプリケーション空間における信頼性レベルを落とすことなく、終了するモジュールを除外するように縮小される。

【０１２５】

図１７に、アプリケーションがリモート認証する間のモジュール間通信について説明するシーケンス図を示す。相互に作用する6つのオブジェクト、11004、11002、11006、11008、11010および11700を図示している。まず、SPE11004は先行技術に係るセキュア処理環境である。好ましい実施形態では、それはMTMである。次に、tPCRサポート11002が存在する。これは、tPCRを参照するPSCを取って、任意のPSCに基づくtPCR状態を検証する処理を

行い、有効であれば、証明書を拡張したことを記録する。抽象化レイヤ11006は、通常モードのアプリケーションからの要求を取り扱い、要求を他のモジュールにわたす。OS11008はオペレーティングシステムであり、ここでは、一時的PCRを正確にアップデートするようなアプリケーションの起動および終了の処理に関するシステムである。アプリケーション11010は、この例における認証を含む任意のタスクを行うサンプルアプリケーションである。そして、サーバ11700はリモート認証を行う。

【０１２６】

まず、アプリケーション11010は、抽象化レイヤ11006にクライアントノンス$N_c$を要求し11701、このランダムに生成された値が返ってくる11702。この値は、アプリケーション11010がサーバ11700に認証を要求する場合に用いられる11703。例えば、保護されたサービスへの、アプリケーション11010によるアクセスを許可する前に、サーバ11700は、アプリケーション11010が期待された環境で動作しているか確認する必要があるため、アプリケーション11010は、サーバ11700からこの許可を得るため認証手順を開始する。アプリケーション11010は、生成されたクライアントノンス$N_c$をサーバ11700に渡す。このクライアントノンスは、リプレイ攻撃やその他の攻撃に対する保護を実現するための値である。サーバ11700は、クエリ用の、サーバノンス$N_s$とランダムに生成されたChallengeと物理的PCRセットとを含むメッセージを付けた認証要求を送信することによって応答する11704。このメッセージは、先行技術で述べられているような匿名証明プロトコルなどでクライアントとサーバとの間において先に確立されたAIKを用いて署名される。ここで、好ましい実施形態では、このメッセージフォーマットはTCGで指定されたものと同じである。アプリケーション11010は、この認証要求の処理11706をOS11008に託す。OS11008は、図１０で述べたようなプロセス空間の情報を用いて、どのアプリケーションまたはダイナミックロードライブラリが関数を呼び出したのかと、モジュールがどのRIM証明書を使ってそのモジュール自身の検証を行ったのかとを判断し11708、アプリケーションのリモート認証用の、先に確立されたAIKを決定する11709。検索されたRIM証明書は、モジュールへ認証要求するために、他の認証パラメータとともに抽象化レイヤ11006へ渡される11710。ここで、認証が開始可能になる。まず、認証用の、サーバノンスとランダムChallengeと物理的PCRとを含むメッセージの署名が、先行技術に基づき先に確立されたAIKを用いてSPE11004によって検証される11712。次に、SPE11004を再び用いて、今度は、アプリケーション11010に対するRIM証明書の完全性を検証し11714、そして、tPCRサポート11002を用いて前記RIM証明書内のtPCRセットの検証を行う11716。これらのチェックが正常に行われたとすると、抽象化レイヤ11006はSPE_Quoteで用いられるハッシュ値を作成する11718。このハッシュは、先にサーバへ送信されたクライアントノンスとサーバノンスと11710で渡されたChallengeとを連結したものおよびアプリケーション11010のRIM証明書に格納されている一時的PCRハッシュから算出される。SPE11004が、サーバから受信したPCRセレクション11704に対応したPCRセットと、11718で算出されたハッシュ値とを含み、確立されたAIKのプライベート部分を用いて署名された署名済みハッシュの生成を要求される11722。SPE11004がMTMである好ましい実施形態において、SPE_QuoteはTPM_Quoteのエイリアスであり、TCG仕様書で定義されたように動作する。この得られた署名値は、SPE11004からサーバ11700へ、11722、11724、11726そして11728と遡って渡される。サーバ11700は、渡された結果が期待される結果と等しいかどうかを検証し11730、等しければ、アプリケーション11010に認証が正常に完了したことを通知する11732。

【０１２７】

好ましい実施形態では、アプリケーション11010およびサーバ11700間の通信（11703、11704、11728、11730および11732）は、インターネットを介したワイヤレスリンクで行われるが、固定リンクや無線リンクを用いる実施形態でも可能であると当業者は分かるであろう。この通信のプロトコルは、メッセージコンテンツを暗号化する必要がないように設計されているが、SSLなどの暗号化プロトコルを用いる実施形態でも可能であると当業者は分かるであろう。

【０１２８】

本発明は上記の実施形態に基づいて述べられているが、本発明は明らかにそのような実施形態に限定されるものではない。下記のケースもまた本発明に含まれる。

【０１２９】

(1)上述の実施形態では、MTM仕様書と同様の方法で検証は行われた。しかしながら、本発明を他の検証システムに適用してもよい。ただし、コンポーネントをチェーンのように検証する検証方法で検証システムがシステムのコンポーネントを検証できる場合（例えば、あるコンポーネントがそのコンポーネントの後に起動する他のコンポーネントを検証する）に限る。例えば、ハッシュ値のMTMへの拡張などの動作はTCG仕様書に固有なものであるため、省略してもよい。

【０１３０】

(２)上述の実施形態では、証明書(RIM証明書)内でハッシュ値を使用することにより検証は行われた。しかしながら、ハッシュ値を使用しない他の検証方法が本発明に適用されてもよい。

【０１３１】

従来のチェックサムや、コンポーネントから抽出される別のデータ(例えば、コンポーネントから抽出される第１所定ビット)が、検証を行うために使用されてもよい。また、証明書は、完全性チェック値を含むデータの集合に置き換えられてもよい。

【０１３２】

さらに、検証方法は、コンポーネントから抽出される値と期待される値が等しいか否かをチェックすることに限定されない。例えば、コンポーネントのサイズを確認し、もしサイズが所定量より大きいか、または小さいなら、コンポーネントは検証されたと判断されるとしてもよい。これらの検証方法は、ハッシュ値と、その期待値とを比較するほど厳密なものではないが、それよりも高速に行われる。

【０１３３】

（３）上記の各装置は、具体的には、マイクロプロセッサ、ＲＯＭ、ＲＡＭ、ハードディスクユニット、ディスプレイユニット、キーボード、マウスなどから構成されるコンピュータシステムである。ＲＡＭまたはハードディスクユニットには、コンピュータプログラムが記憶されている。マイクロプロセッサが、コンピュータプログラムにしたがって動作することにより、各装置は、その機能を達成する。ここでコンピュータプログラムは、コンピュータに対する指令を示す命令コードが複数個組み合わされて構成されたものである。

【０１３４】

（４）上記の各装置を構成する構成要素の一部または全部は、１個のシステムＬＳＩ（Large Scale Integration：大規模集積回路）から構成されているとしてもよい。システムＬＳＩは、複数の構成部を１個のチップ上に集積して製造された超多機能ＬＳＩであり、具体的には、マイクロプロセッサ、ＲＯＭ、ＲＡＭなどを含んで構成されるコンピュータシステムである。ＲＡＭには、コンピュータプログラムが記憶されている。マイクロプロセッサが、コンピュータプログラムにしたがって動作することにより、システムLSIは、その機能を達成する。

【０１３５】

さらに、各装置を構成する構成部品の各ユニットは、別個の個別チップとして、または、一部もしくは全てを含む単一のチップとして作られてもよい。

【０１３６】

さらに、ここでは、LSIが述べられているが、集積化の程度の違いにより、指定IC、LSI、スーパLSI、ウルトラLSIが使用される場合もある。

【０１３７】

さらに、回路の集積化の手段は、LSIに限定されるものではなく、専用回路、もしくは汎用プロセッサによる実装も利用できる。さらに、LSIが製造された後にプログラム可能なフィールドプログラムゲートアレイ(FPGA)を使用することもまた可能であり、またLSI内で回路セルの接続および設定を再構成可能なリコンフィギュアラブルプロセッサも使用

可能である。

【０１３８】

さらに、もしLSIに置き換わる集積回路技術が半導体技術もしくは他の派生する技術の進歩を通じて現われるのであれば、その技術は当然に構成要素の集積化を実現するために使用することができる。バイオ技術の適用が予想される。

【０１３９】

（５）上記の各装置を構成する構成要素の一部または全部は、各装置に脱着可能なＩＣカードまたは単体のモジュールから構成されているとしてもよい。ＩＣカードまたはモジュールは、マイクロプロセッサ、ＲＯＭ、ＲＡＭなどから構成されるコンピュータシステムである。ＩＣカードまたはモジュールは、上記の超多機能ＬＳＩに含まれるとしてもよい。マイクロプロセッサが、コンピュータプログラムにしたがって動作することにより、ＩＣカードまたはモジュールは、その機能を達成する。このＩＣカードまたはこのモジュールは、耐タンパ性を有するとしてもよい。

【０１４０】

（６）本発明は、上記に示す方法をコンピュータにより実現するコンピュータプログラムであるとしてもよいし、コンピュータプログラムなどのデジタル信号であるとしてもよい。

【０１４１】

また、本発明は、コンピュータプログラムまたはデジタル信号をコンピュータ読み取り可能な記録媒体、例えば、フレキシブルディスク、ハードディスク、ＣＤ－ＲＯＭ、ＭＯ、ＤＶＤ、ＤＶＤ－ＲＯＭ、ＤＶＤ－ＲＡＭ、ＢＤ（Blu-ray Disc）、半導体メモリなどに記録したものとしてもよい。また、これらの記録媒体に記録されているデジタル信号であるとしてもよい。

【０１４２】

また、本発明は、コンピュータプログラムまたはデジタル信号を、電気通信回線、無線または有線通信回線、インターネットを代表とするネットワーク、データ放送等を経由して伝送するものとしてもよい。

【０１４３】

また、本発明は、マイクロプロセッサとメモリを備えたコンピュータシステムであって、メモリは、上記コンピュータプログラムを記憶しており、マイクロプロセッサは、コンピュータプログラムにしたがって動作するとしてもよい。

【０１４４】

また、プログラムまたはデジタル信号を記録媒体に記録して移送することにより、またはプログラムまたはデジタル信号を、ネットワーク等を経由して移送することにより、独立した他のコンピュータシステムにより実施するとしてもよい。

【０１４５】

（７）本技術分野の当業者にとっては、説明した実施形態の、本発明固有の教示および利点から原理的に離れることのない多くの変形がありえることを容易に理解することができるだろう。また、上記変更および実施形態の任意の組み合わせは、本発明の範囲内に含まれる。

【０１４６】

（第２の実施形態）

以下に、本発明の好ましい実施形態について説明する。

【０１４７】

第２の実施形態は一時的なPCRの利用をサポートするシステムに関し、この一時的なPCRは、証明書によって定められた値以外の定義済み存続期間を有する。前述の追加オペレーティングシステム機能およびPSCの信頼できる検証を行うことによって、SPE付き装置の開発者はこれらのtPCRを扱うシステムを作成できる。使用対象のtPCRについて記述するPSCを提供することによって、このような装置におけるアプリケーションの開発者は、信頼できる実行を柔軟な方法で提供するアプリケーションを作成できる。本発明によると、アプ

リケーションはいずれかのコンポーネントタイプとして定義されるが、独立型のプログラムや独立型のプログラム用プラグインモジュールやプラグイン用ヘルパーモジュールに限定されるものではない。

【０１４８】

図２１Ａに、「TCG Mobile Reference Architecture」に記載された手段などのシステムのセキュアブートをサポートする場合における先行技術を示す。標準実行環境において、抽象化レイヤAPI2102を使用するアプリケーション2100が存在する。破線は、上方の前述した標準実行モードと下方のセキュアモードとの間のセキュアモードインターフェース2106を示す。標準実行モードとは、ほとんどのコンピュータシステムが提供するような通常の実行環境である。セキュアモードでは、標準実行環境からはアクセスできないメモリ空間において、許可されたソフトウェアに限り実行可能なセキュア実行環境を提供する。本発明の好ましい実施形態では、セキュアモードしか分からないキーのプライベート部分を有するソフトウェアを暗号化することによってこの許可が行われるが、ホワイトリストや証明書などその他の技術を用いても構わない。この実行環境は、必要に応じて他のモジュールとともに、セキュアブートモジュールとセキュア処理環境2114とを有する。上記のセキュアモードインターフェース2106はセキュアブートの信頼性の境界2104であって、そのライン以下全てが信頼性のある環境内となる。そのラインは、「TCG Mobile Trusted Module Specification」で定義されたように検証および拡張を行うセキュアブートプロセス中に構築される。セキュアモードの抽象化レイヤAPI2108は、サービスに対する通常モードからの要求を取り扱い、更なる処理のために抽象化レイヤ2110にこれらの要求をわたす。抽象化レイヤのタスクの１つはPSCデータベース2112を管理することである。また別のタスクとしては、物理的PCRs2116の操作など、セキュア処理環境2114へのアクセスを含む、セキュアブートコンポーネント2113が提供するサービスの要求を取り扱うことがある。また、セキュアブートコンポーネント2113もPSCデータベース2112へのアクセスを要求する。セキュア処理環境2114は、ハードウェアまたはソフトウェアのいずれで実現されてもよいが、好ましい実装では、「Trusted Computing Group specifications」で定義されたようなモバイルトラステッドモジュールであり、別の好ましい実装では、トラステッドプラットフォームモジュールである。ソフトウェア処理環境2113は、ハードウェアまたはソフトウェアのいずれか、あるいは双方の組み合わせで実現されてもよい。そして、セカンダリRIC（Runtime Integrity Checker）モニタ2118が存在する。このモニタのタスクには、改ざんが行われたと思われる場合にアプリケーション2100を終了させることが含まれる。これは、好ましい実装では、PSCに格納された基準ハッシュに対して現在のアプリケーションのハッシュを検証することによって実現される。この検証は、特定のイベントが発生した時あるいは定期的な間隔で行われてもよく、これらのRICモニタに階層があっても構わない。これらのRICモニタは１以上の子RICモニタの完全性をチェックする各RICモニタレベルを有する。プライマリRICは図示されていないが、そのタスクは、システム全体に対するマスター検証ルートとして、例えばハイパーバイザなどの図示されたシステム外で次のことを実行することである。すなわち、プライマリRICは、１以上のコンポーネントに対する完全性チェックを一定間隔で行い、PSCに格納された基準ハッシュに対してこれらのコンポーネントハッシュを検証する。このプライマリRICモニタのコンポーネントのうち１つはセカンダリRICを備える必要がある。「TCG Mobile Reference Architecture」では、これをPRMVA、すなわちPrimary Runtime Measurement Verification Agentと称しており、このPRMVAは、セカンダリRICモニタに相当するSRMVA、すなわちSecondary Runtime Measurement Verification Agentを有している。図示されたもの全ては装置2120に配置される。

【０１４９】

セキュアモードは、システムプロセッサの単独実行モード、オペレーティングシステムカーネルモード、セキュリティコプロセッサ、仮想マシン、ハイパーバイザ、完全性チェックされたメモリなど、当業者にとって周知な数々の技術により実現されてもよい。各コンポーネントは、本発明の新規性を有する教示と効果を逸脱しない範囲で、１以上の記載

されている技術またはその他の技術によって保護されても構わない。

【０１５０】

当該先行技術のその他の実施形態では、セキュアモードインターフェース2106は抽象化レイヤ2110とセキュア処理環境2114との間に配置される。当業者は、抽象化レイヤの実行に対してセキュアモードの完全な保護を必要とせず、上述したRICモニタが行う完全性保護のみを要するようなかたちで当該抽象化レイヤを実装してもよく、また、本発明を、完全性保護された環境に実装してもよいと分かるであろう。当該完全性保護は、ソフトウェアまたはハードウェア、あるいは双方の組み合わせで行われる。

【０１５１】

図２１Ｂに、セキュアモードのサポートはないが、「TCG Specification Architecture Overview Revision 1.2 28 April 2004」に記載された手段などの独立したセキュア処理環境を有し、セキュアモードインターフェースが存在しない場合における先行技術のその他の実施形態を示す。セキュアモードインターフェースがないため、抽象化レイヤAPI2108は１つしか存在せず、セキュアブートコンポーネントの代わりに、トラステッド/セキュアブートコンポーネント2152が存在する。セカンダリRICモニタ2118は、セキュア処理環境2114以外のシステムにおける全てのコンポーネントをカバーできるように拡張され、このようにトラステッド/セキュアブートの信頼性の境界2150を確立する補助を行う。しかしながら、そうでない場合には、コンポーネントとそれらの役割は図２１Ａで説明した通りである。

【０１５２】

図２２Ａに、図２１Ｂなどの先行技術に基づいた本発明の第２の実施形態を示す。すでに述べたように、標準実行環境における抽象化レイヤAPI2102を使用するアプリケーション2100が存在する。破線は、上方の前述の標準実行モードと下方のセキュアモードとの間のセキュアモードインターフェース2106を示す。上記のセキュアモードインターフェース2106は、先に述べたような実行前にコンポーネントを検証するプロセスによって構築されたセキュアブートの信頼性の境界2104であって、そのライン以下全てが信頼性のある環境内となる。OSサポート2200モジュールは、標準実行空間にあるが、信頼性のある境界内である。このモジュールは、どのアプリケーションが起動前の検証にどの証明書を用いたかを示すマッピングを維持するためのコンポーネント・PSCマップ2202を管理する。セキュアモード内に、サービスに対する通常モードからの要求を取り扱い、更なる処理のために抽象化レイヤ2110にこれらの要求をわたす抽象化レイヤAPI2108がある。抽象化レイヤのタスクの１つは、PSCデータベース2112を管理することであり、また別のタスクにはtPCRサポート2204を実装することがある。このサポートモジュールが、拡張されたけれども取り消されていないPSC全ての有向非巡回グラフを含む拡張PSCツリー2206データを維持する。さらに別のタスクは、物理的PCRs2116の操作など、セキュア処理環境2114へのアクセスを含む、セキュアブートコンポーネント2113が提供するサービスの要求を取り扱うことである。また、セキュアブートコンポーネント2113もPSCデータベース2112へのアクセスを要求する。そして、セカンダリRIC（Runtime Integrity Checker）モニタ2118が存在し、そのタスクには、改ざんが行われたと思われる場合にアプリケーション2100を終了させることが含まれる。図示されたもの全ては装置2120に配置される。

【０１５３】

先行技術と同様に、先行技術の好ましい実施形態では、セキュアモードインターフェース2106は抽象化レイヤ2110とセキュア処理環境2114との間に配置される。当業者は、抽象化レイヤの実行に対してセキュアモードの完全な保護を必要とせず、上述したRICモニタが行う完全性保護のみを要するようなかたちで当該抽象化レイヤを実装してもよく、また、本発明を、完全性保護された環境に実装してもよいと分かるであろう。当該完全性保護は、ソフトウェアまたはハードウェア、あるいは双方の組み合わせで行われる。

【０１５４】

セキュアモードは、システムプロセッサの単独実行モード、オペレーティングシステムカーネルモード、セキュリティコプロセッサ、仮想マシン、ハイパーバイザ、完全性チェ

ック済みメモリなど、当業者にとって周知な数々の技術により実現されてもよい。各コンポーネントは、本発明の新規性を有する教示と効果を逸脱しない範囲で、１以上の記載されている技術またはその他の技術によって保護されても構わない。

【０１５５】

また、当業者は、tPCRサポート2204と拡張PSCツリー2206とをセキュア処理環境2114に移動させたものがその他の実施形態であると分かるであろう。さらなる実施形態は、抽象化レイヤ2110がセキュアモードインターフェース2106の外側になって、tPCRサポート2204と拡張PSCツリー2206がセキュア処理環境2114の内側になるように、これらの他の実施形態を組み合わせたものである。

【０１５６】

図２２Ｂに、セキュアモードのサポートはないが、”TCG Specification Architecture Overview Revision ”に記載された手段などの独立したセキュア処理環境を有し、セキュアモードインターフェースが存在しない場合における、図２１Ｂに基づいた本発明の第２の実施形態の変形例を示す。セキュアモードインターフェースがないため、抽象化レイヤAPI2108は１つしか存在せず、セキュアブートコンポーネントの代わりに、トラスティッド/セキュアブートコンポーネント2152が存在する。セカンダリRICモニタ2118は、セキュア処理環境2114以外のシステム内における全てのコンポーネントをカバーできるように拡張され、このようにしてトラステッド/セキュアブートの信頼性の境界2150を確立する補助を行う。しかしながら、そうでない場合には、コンポーネントとそれらの役割は図２２Ａで説明した通りである。

【０１５７】

図２３Ａに、本発明にかかるPCRの2タイプの利用パターンを示す。まず、通常アプリケーション空間におけるアプリケーション階層を示す。第１マッシュアップ2300は、第１プラグイン2302と第２プラグイン2304とからサービスを利用する。これらのプラグインは第１アプリケーション2306と第２アプリケーション2308とがそれぞれ所有するものである。どちらのアプリケーションも抽象化レイヤAPI2102からサービスを利用する。次に、セキュアモードインターフェース2106の反対側に、抽象化レイヤAPI2108のセキュアモードのサポートが存在する。これは抽象化レイヤ2110とやり取りする。抽象化レイヤのタスクの１つは、tPCRサポート2204を実装することである。このサポートモジュールにより、拡張され、かつ、取り消されていないPSC全ての有向非巡回グラフを含む拡張PSCツリー2206データが維持される。また別のタスクは、物理的PCR2116を操作する要求をセキュア処理環境2114に渡すことなどである。これは、ハードウェアまたはソフトウェアのいずれで実現されても構わない。

【０１５８】

ところで、物理的PCR2116と一時的PCR2204を利用する典型的なパターンは以下の通りである。物理的PCRの読み取り2310動作は随時有効である。セキュア処理環境2114でサポートされる全ての関数は必ず物理的PCRを使用し、決して一時的PCRを使用しない。しかしながら、もし上述したように、tPCRサポートコンポーネント2204をセキュア処理環境2114内に移動したならば、SPEがtPCRを使用することも可能である。物理的PCRへの書き込み2314は、先行技術で教示されたように主にブート時に行われるが、さらに、アプリケーション空間から物理的PCRの書き込みが可能である2312。どの書き込み動作をアプリケーション空間から行うかを決定するのはシステム設計者または実装者次第である。一時的PCRに対し、読み取り2316と書き込み2318の双方はアプリケーション空間において排他的に行われるのが一般的である。一時的PCRの性質により、各実装者にはこれらのtPCRをどのように使用するか選択の自由度がある。しかしながら、図例のような場合には、第１マッシュアップ2300の開発者は、第１プラグイン2302と第２プラグイン2304の開発者達と調整して、どのtPCRをそれぞれ利用する予定か全員が把握できていると確認する必要があるであろう。

【０１５９】

図２３Ｂに、拡張PSCツリー2206を示す。ノードの追加および削除方法については、後

で述べる。図示された有向非巡回グラフは、図２２Ａに示したように拡張されているとしてtPCRサポートモジュール2204が記録したPSCを表している。図２２Ａに示されたモジュールとこの図の証明書との間には1対2の関係があり、信頼性の境界を拡張して第１アプリケーション2306をカバーするには、先行技術で教示されているように、２つの証明書、つまり第１アプリケーション開始2350および第１アプリケーションロード2354が必要である。第２アプリケーション2308に対しては、第２アプリケーション開始2352および第２アプリケーションロード2356が必要である。第１プラグイン2302に対しては、第１プラグイン開始2358および第１プラグインロード2362、第２プラグイン2304に対しては、第２プラグイン開始2360および第２プラグインロード2364、第１マッシュアップ2300に対しては、第１マッシュアップ開始2366および第１マッシュアップロード2368が必要である。証明書間の矢印は、これらの証明書の従属関係を示している。従属関係は、各証明書の拡張時に各証明書が検出を期待するPCR状態により定義される。ツリーの各ノードの構造体は、図２６において後ほど定義する。

【０１６０】

図示したように、先行技術によると、各モジュールにはそのモジュールに関連付けられた２つの証明書があり、１つは、起動前にモジュールを検証するためにその親が使用し、もう１つは、期待された環境でモジュールが起動されたことを検証するためにモジュール自身が使用する。当業者は、モジュールごとに２つ以外の証明書を使用しても本発明の範囲内であると分かるであろう。

【０１６１】

図２４に、プラットフォーム状態証明書2400（PSC）を示す。これは、そのプラットフォーム状態証明書が定める（仮想的または一時的な）PCRで定義されるプラットフォームの状態と、プラットフォームの状態の検証がうまくいった時に（仮想的または一時的な）PCRへ拡張する値とを表す構造体である。これらの構造体をPSCデータベース2112に格納してもよい。本構造体の第１フィールドはPSC名2402である。この名前は、PSCデータベース2112へのPSCの格納およびPSCデータベース2112からのPSCの取り出しに用いられるキーフィールドなので固有でなければならない。また、好ましい実装では、この名前は、人間が読み取り可能な名前を表すバイト文字列である。アプリケーション開発者が使用する名前を決定してもよい。あるいは、プラットフォームの製造者がアプリケーション開発者に名前を提供してもよい。当業者は、GUIDなどの他の表示を代わりに用いてもよく、PSC名を選択する方法は他にもあると分かるであろう。次に、検証用PCR状態2404を表すエントリーリストが存在する。検証対象のPCRごとに、一対の値、つまりPCRインデックス2406およびPCR値2408が存在する。次に、拡張用PCR値が存在する。最初に拡張用PCRインデックス2410そして拡張用値2412である。最後に、セキュア処理環境2114の知る鍵により暗号化されたデータの残りのハッシュを表す暗号署名2414が存在する。この署名鍵は、セキュア処理環境2114へ安全に組み込まれたキーのプライベート部分か、署名PSCに使用するものであるとして上記の組み込まれたキーが直接または間接的に承認したキーのいずれかである。また、署名者は、プラットフォーム開発者またはアプリケーション開発者のエージェント、あるいは有効な署名鍵を発行した他の者でも構わない。

【０１６２】

本発明によると、プラットフォーム状態証明書2400を単に確認するだけでは、物理的PCRなのか一時的PCRなのか判断することはできない。どの種類のPCRをチェックすべきか決定するのは、それを用いる状況である。この利点の１つは、セキュアブート用に証明書を作成する既存のツールを、アプリケーション空間用に証明書を作成するために再利用できることである。

【０１６３】

好ましい実施形態では、検証用PCR状態2404の対リストを、テスト対象であるPCRインデックス2406を表すビットマップとPCR値2408の集合に対するハッシュとで置き換えてもよい。これは、RIM証明書に対して「TCG Mobile Trusted Module Specification」が定義した表示である。より複雑なチェックコードを費やせば、一時的なPCRを検証する証明書を

変更することなく、このような表示を用いることが可能である。しかし、好ましい実施形態では、図２５に図示された一時的PCR用RIM証明書2500を用いる。この構造体と図２４のプラットフォーム状態証明書とのフィールドの関係および追加フィールドについて詳しく述べる。

【０１６４】

ラベル2502はPCS名2402に相当する。measurementPCRIndex 2504とmeasurementValue 2506とは、拡張用PCRインデックス2410と拡張用値2412とに相当する。検証用tPCR状態2518、およびPCRインデックス2514とtPCR値2516とを含む対リストは、プラットフォーム状態証明書2400で定義されたフィールドと類似のものである。検証用tPCR状態2518を一時的PCR用RIM証明書2500に関連付けるためには、extensionDigestSize 2508とextensionDigest 2510のフィールドを使用することが必要である。extensionDigestSize 2508は、extensionDigest 2510のバイトサイズであり、extensionDigest 2510には検証用tPCR状態2518の構造体のハッシュが含まれる。state 2512フィールドに設定されたビット数はテーブルのペア数を示すので、サイズインジケータを格納する必要はない。当業者は、tPCRインデックス系列など、tPCR値2516フィールドに所定順序がある場合には、tPCRインデックス2514フィールドを格納する必要さえないと分かるであろう。

【０１６５】

図２６に、拡張PSCツリーノード2600を示す。これは、証明書１つの拡張を記録する構造体である。このノードの構造体では、図２３Ｂに図示したような各ノードの内容、つまり項目2350～2368を詳しく説明している。拡張PSCツリー2206は、当該技術分野において周知の技術を用いて有向非巡回グラフを実装する。例えば、ブーストC++ライブラリにはブーストグラフライブラリが含まれている。これは、上述した有向非巡回グラフなど様々なグラフの作成および操作をサポートする。このように、拡張PSCツリーノード2600はグラフの各頂点に関連付けられる。拡張PSC名2602は、拡張されたプラットフォーム状態証明書2400のPSC名2402のフィールドである。tPCR状態2604は、このノードにおける現在の一時的PCR状態のキャッシュであり、このノードの先祖に当たる拡張済みのPSCから算出される。tPCR状態は、tPCRインデックス2606とtPCR値2608との対リストで構成される。当業者は、このデータの別の表現、例えば、用いたtPCRを表すビットマップでtPCRインデックス2606フィールドを置き換えるなどが存在すると分かるであろう。

【０１６６】

図２７に、OSサポートモジュール2200によって維持されたコンポーネント・PSCマップ2202を示す。このコンポーネント・PSCのマップ2202には、コンポーネントをPSCへ2700マッピングするリストが含まれる。このリストの各エントリーには、コンポーネントID2702と拡張PSC名2704、つまりアプリケーションの起動前に拡張されたプラットフォーム状態証明書2400のPSC名2402フィールドとが含まれる。Windows（登録商標）ベースのプラットフォームにおける本発明の好ましい実施形態では、コンポーネントID2702は２つのフィールドから構成されている。第１フィールドは、プロセスID2706である。このプロセスIDは、GetCurrentProcessId()などのWin32APIによって決定されるような、コンポーネントが属するプロセスを一意的に表す識別子をもつ。第２フィールドはモジュールハンドル2708である。単独で実行可能なコンポーネントに対しては、このフィールドを必ず0に設定する。リンクライブラリとして実装されたコンポーネントに対しては、第１パラメータのDllMainエントリーポイントに渡されるように、ライブラリ用のHMODULEがこのフィールドに含まれる。この構造体の利用法については後で述べる。

【０１６７】

「TCG Mobile Reference Architecture」によると、PCR0は、基礎を成すハードウェアプラットフォームの特徴を表す値を保持する。PCR1は、信頼性のルートを表す値を含む。PCR2は、エンジンロードイベントを表す値を含む。PCR3～6および8～12は、専有手段を含む。PCR13～15はアプリケーションが自由に使用できる。セキュアブートが正しく行われたことをPCR0、1および2が期待通りに示すかどうかのテスト、PCR13が0に設定されたかどうかのテスト、そして、全てが正しければ新たな値をPCR13に拡張することをアプリケー

ションプログラマが希望したとする。図２８に、先行技術にかかる、「第１アプリケーション開始」と名付けられたプラットフォーム状態証明書2400のサンプルを示す。証明書の名前は2800に記録される。上述したように、検証用PCR状態2404は、チェックするPCRインデックスとPCR値との対を4つ含み、2802、2804、2806、2808、2810、2812、2814、2816と番号付けする。2802はPCR0を示し、2804の＜ハードウェアプラットフォーム＞は、基礎を成すハードウェアプラットフォームを表す公表値を示す。2806はPCR1を示し、2808の＜信頼性のルート＞は、基礎を成す信頼性のルートを表す公表値を示す。2810はPCR2を示し、2812の＜エンジンロードイベント＞は、PCR2に拡張されたロードイベント値から算出された合成ハッシュを表す公表値を示す。2814はPCR13を示し、2816の0の値は、PCR13がまだ初期状態であるという予測を示す。次に、2818へ拡張するPCRが存在し、そして、そのPCRへ拡張する値820 が存在する。

【０１６８】

先行技術にかかる図２８の証明書に関する問題として次のものがある。他のアプリケーションがPCR13を使用してしまった場合、検証用PCR状態2404はもはや正確ではなくなってしまうという問題がある。アプリケーションが終了して再起動する場合、2818と2820に定義した先に拡張された値によってPCR13は0以外の状態に設定されてしまい、検証用PCR状態2404はもはや正確ではなくなってしまうという問題がある。

【０１６９】

しかしながら、本発明では、図２８の証明書と類似の証明書は、アプリケーション自身と同じ時間または別々の時間で対象装置に展開するためにアプリケーション開発者によって、図２９に示すような２つの証明書に分けられる。なぜ分けるかというと、既存の証明書により２つの異なるPCRセットを検証するためである。第１のセットにはセキュアブートプロセスの結果、つまり既知の不変な結果が含まれ、第２のセットには動的状態、つまりアプリケーションレベルの状態が含まれる。図２８において、項目2802および2804は、ハードウェアプラットフォームを表す既知のセキュアブートPCR0の値を指している。項目2806および2808は、信頼性のルートを表す既知のセキュアブートPCR1の値を指し、項目2810および2812は、エンジンロードイベントを表す既知のセキュアブートPCR2の値を指している。また、項目2814および2816は、期待される前提条件を表す、ブート後のPCR13の望ましい値を指している。このように、アプリケーション開発者は、セキュア処理環境が管理する物理的PCR、すなわち本例ではPCR0、1および2をテストするために用いられる片方の証明書に既知のセキュアブートPCRの値を置くことによって、図２８の単一証明書を図２９に示した２つの証明書に分けてもよい。第２の証明書は、アプリケーション空間の一時的PCR、すなわち本例ではPCR13用に用いられる。

【０１７０】

「第１アプリケーション開始（セキュアブート）」2900と名付けられた第１証明書によって、セキュア環境が正しいことを確保するため、セキュアブートプロセスがセットアップした物理的PCR（PCR0 2802、PCR1 2806およびPCR2 2810）はテストされる。ここで、拡張用PCRは、拡張がないことを示すために-1 2902に設定され、拡張用値2904はゼロという名目上の値である。この証明書は検証のみに使われる。そして、アプリケーションレベルでは、図２３Ａに示すように物理的PCRへの書き込みをやめ、一時的PCRを用いることによってより柔軟性が高まり、前述した当該技術分野における課題を回避することができる。「第１アプリケーション開始（一時的）」2920と名付けられた第２証明書によって、ゼロ 2816である一時的PCR13 2814はテストされ、同じレジスタ2818および2820に値を拡張する。tPCRのインデックス13を選択したことに特に意味はない。先行技術の状況とは異なり、tPCR0またはtPCR99が用いられやすいであろう。

【０１７１】

図３０に、シーケンス図を示す。この図では、アプリケーション起動時には図２９の２つの証明書、つまり一方が物理的PCRをテストする証明書でありもう一方が一時的PCRをテストする証明書を用い、アプリケーション終了時には「第１アプリケーション開始（一時的）」2920の第２証明書を用いて、値をtPCRに拡張してからその拡張を取り消すという本

発明にかかるイベントシーケンスを示す。相互に作用する6つのオブジェクト、21000、21002、21004、21006、21008および21010を図示している。まず、tPCR13 21000は、この例の状況における一時的PCR13の状態を表す。図２６で説明したように、tPCRは、特定のメモリ位置というよりむしろ拡張PSCツリー2206のノードごとに記録されるが、本例においては理解しやすいよう、tPCR13 21000をそのような位置にあるかのごとく表す。次に、tPCRサポート21002が存在する。これは、tPCRを参照するPSCの取得と現在のtPCR状態の検証との処理を行い、有効であれば、証明書は拡張されたと記録する。SPE21004は、先行技術にかかるセキュア処理環境である。好ましい実施形態では、それはMTMである。抽象化レイヤ21006は、通常モードのアプリケーションからの要求を取り扱い、要求を他のモジュールにわたす。OS21008はオペレーティングシステムであり、ここでは、一時的PCRを正確にアップデートするようなアプリケーションの起動および終了の処理に関するシステムである。そして、図３０のシーケンス図のプロセスは入れ子されてもよいので、アプリケーション21010は、例えば、アプリケーション実行21042が21014から始まるイベントシーケンスに従う別のアプリケーションの開始を含むような、PSCによって保護された他のアプリケーションの開始を要求するアプリケーションであっても構わない。図を単純化するため、エラー処理については図示しないが、本発明から除外されるものではない。

【０１７２】

まず最初に、tPCR13 21000は、0の値21012で始まる。本発明では、拡張PSCツリー2206のルートは、全tPCRを０に設定させた状態で始まるというのがルールである。当業者であれば、セキュアブートの完了後に物理的PCRの値でtPCRを初期化するというような他の初期値候補が可能だと分かるであろう。OS21008は、要求を検出してアプリケーション21010を起動するので、まず、起動しようとするアプリケーションによってどのPSCが用いられるのかを決定する21014。Windowsベースのオペレーティングシステムにおける好ましい実施形態では、実行ファイルに組み込まれたカスタムアセンブリが、使用する２つのPSCを特定する。そのアプリケーションは、改ざんから保護するためにマイクロソフトのストロングネームツールを用いて署名される。次に、21014で特定されたPSC、この図では「第１アプリケーション開始（セキュアブート）」および「第１アプリケーション開始（一時的）」を抽象化レイヤ21006に要求する21016、21018。そして、これら２つのPSCの検証が、抽象化レイヤAPIであるAL_VerifyPSCsAndExtendtPCRを呼び出す21020ことによって行われる。このAPIは、システムが開始したいアプリケーションを検証する２つのPSC、つまり、図２９で示したような物理的PCR用のPSCおよび一時的PCR用のPSCを引数とする。まず、図の21024で表されたPSC「第１アプリケーション開始（セキュアブート）」を引数とした、SPEのAPIであるSPE_VerifyPSCStateを呼び出す21022。先行技術によれば、これによって、PSC自身のフォーマットおよび署名のチェックが行われ、そして、PSCの検証用PCRが物理的PCRの現在の値と一致することを検証する。先行技術によれば、PSCがプラットフォームに渡されると、セキュア処理環境2114に組み込まれたキーか、この組み込まれたキーによって正当であると検証されることのできるキーのいずれかでPSCは署名される。そして、有効であれば、セキュア処理環境2114が生成かつ安全に格納した他のキーで再び署名され、証明書はPSCデータベース2112に格納される。

【０１７３】

本発明の好ましい実施形態によると、物理的PCRをチェックするPSCはオプショナルなので、ステップ21016および21018を省略してもよい。その他の好ましい実施形態によると、物理的PCRのチェック用のPSCがアプリケーション全てについて同じである場合、物理的PCR用のPSC１つを２つ以上の異なる一時的PCRのPSCが用いてもよい。

【０１７４】

次に、SPEの他のAPI、すなわち、図の21030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定したSPE_VerifyPSCを呼び出す21026。先行技術によると、これによって、物理的レジスタに対してPCRの設定を検証せずとも、PSC自身のフォーマットおよび署名のチェックが行われる。そこで、tPCRサポートモジュール、すなわち、図の21030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定した

APIであるTPCR_VerifyPSCAndExtendを呼び出す21028。このAPIの最初のタスクは、拡張PSCツリー2206内の既存状態に対応する検証用PCR状態2404をチェックすることにより、PSCを拡張できることを検証する21032ことである。この動作の詳細については後で述べる。検証が正常に完了した時点で、このPSCにおける動作の成功が、拡張PSCツリーの正しい位置へその表示を付け加える21034ことによって記録される。この動作の詳細については後に述べる。このPSCをツリーへ付け加えたことによる結果の１つは、tPCR13 21000、つまり拡張対象のレジスタが、以前の値、この場合は０、とPCRから拡張するための値1030、つまり0xABCD1234とを連結したもののハッシュに設定されたレジスタの値を有することである。これは、合成ハッシュと呼ばれ、数式的には、tPCR13 = SHA-1（0xABCD1234と連結されたtPCR13）と書き込まれる。この動作は21036の構文 (+)=で表される。このように、環境が期待された状態であることを検証し、成功したことの記録が生成され、そして、制御がオペレーティングシステムに戻る。先行技術によると、21020におけるPSC検証の前の、更なる安全性の評価として、アプリケーションのハッシュを算出し、拡張するPSCに格納された基準値と比較する。本発明では、この値は、好ましい実施形態において0xABCD1234と表された、拡張用の値として、PSC「第１アプリケーション開始（一時的）」21030に格納される。しかしながら、このステップは図から省略されている。

【０１７５】

アプリケーションを起動する際に、OSは、そのアプリケーションのプロセスIDを取得し、コンポーネント・PSCマップ2202に、この識別子と一時的レジスタ用の対応PSC、つまりPSC「第１アプリケーション開始（一時的）」とを記録する21038。マイクロソフトのWindows環境における好ましい実施形態では、本プロセスは、http://www.codeproject.com/KB/system/InterceptWinAPICalls.aspxのコードプロジェクトにおけるAndriy OriekhovのIntercepting WinAPI callsに記載されたようなプロセス作成プロセスを途中でインターセプトすることによって実装される。取得したプロセスハンドルは、プロセスID2706に変換され、そのフィールドに設定される。そして、モジュールハンドル2708を０に設定する。コンポーネントがダイナミックリンクライブラリである場合、http://www.lenholgate.com/archives/000369.htmlの/* Rambling comments... */におけるLen HolgateAPIのWhy does windows hold the loader lock whilst calling DllMain?に記載されたように、LoadLibrary()およびFreeLibrary()コードをフックし、DllMain()への呼び出しをトラップする。このトラップを設けたまま、プロセスID2706を現在のプロセスIDに設定し、モジュールハンドル2708をDllMain()の第１引数に設定する。

【０１７６】

アプリケーションは起動されると21040、PSCに関連付けられた他のアプリケーションを起動またはtPCR自体を参照する他のPSCを拡張したとしても、プログラムされたように実行し続ける21042。そして、アプリケーションを閉じるとユーザが選択する、クラッシュする、または、アプリケーションを強制的にシャットダウンするセカンダリRICモニタ（この図では図示していない）が改ざんを検出するといういずれかの原因で、アプリケーションは終了する21044。

【０１７７】

アプリケーションが終了するので、OSはアプリケーションのプロセスIDを取得し、そのプロセスIDと0であるモジュールハンドルとを用いて、コンポーネントID2702を作成する。このコンポーネントIDを用いてコンポーネント・PSCマップ2202を検索し、アプリケーションを起動するのに用いられたPSCを見つける21046。これにより、PSC「第１アプリケーション開始（一時的）」21030が返されるので、取り消しを行うため、OSは、抽象化レイヤ21006のAPIであるAL_UndoPSCExtendを、PSCを引数として呼び出す21048。コンポーネントがダイナミックリンクライブラリである場合、上述したようにFreeLibrary() APIをフックするので、そのルーチン内で現在のプロセスIDを問い合わせ、FreeLibrary()のパラメータからモジュールハンドルを取得し、そしてこれら２つのデータ項目を用いて、コンポーネントID2702を作成する。このコンポーネントIDを用いてコンポーネント・PSCマップ2202を検索し、ライブラリを起動するのに用いられたPSCを見つける。前述と同様に

、SPEの他のAPI、すなわち、図の21030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定したSPE_VerifyPSCを呼び出す21026。先行技術によると、これによって、物理的レジスタに対してPCRの設定を検証せずとも、PSC自身のフォーマットおよび署名のチェックが行われる。そこで、tPCRサポートモジュール21002、すなわち、図の21030で表されたPSC「第１アプリケーション開始（一時的）」にパラメータを設定したAPIであるTPCR_VerifyPSCAndExtendを呼び出す21050。このAPIの最初のタスクは、拡張PSCツリー2206にそのPSCがすでに存在するかどうか確認することによって、PSCが拡張されたことを検証する21052ことである。検証が正常に完了した時点で、これは、21028における拡張動作を取り消すことが可能であることを意味する。これは、拡張PSCツリー21034から、PSCを表すノードおよびそれに従属する他のノード全てを削除することによって実現される。この動作の詳細は後で述べる。ツリーからこのPSCを削除することの結果の１つは、tPCR13 21000、つまり取り消す対象のレジスタが効率よくその状態を0にリセットされる21056ことである。このように、環境が期待した状態であることを検証し、拡張PSCツリーからノードを削除する21034ことによって先に拡張された動作を取り消し、制御がオペレーティングシステムに戻り、本システムは他の動作をすぐに実行することができる。当業者であれば、これらの他の実行動作の１つとして、終了したアプリケーションを再起動することもあると分かるであろう。tPCR13が21056において00…00、つまり21012で示されたtPCR用の開始値にリセットされた後は、アプリケーションの開始PSC21030の検証は２回目も正常に再実行されるので、本発明によると、アプリケーションを再起動することができる。

【０１７８】

図３０に、実行ファイルの起動および終了時に信頼性の境界を動的に拡張かつ縮小するシーケンス図を示す。注釈を用いて、好ましい実施形態に沿ったダイナミックリンクライブラリに対する同様のタスクをWindows Portable Executable形式のモジュールでどのように行うかを示す。Java ArchiveモジュールやJARなどのPortable Executableベースでないモジュール形式に対しては、モジュールをロードかつアンロードするエンジンによって同様の手法が用いられるか、代わりに、モジュール自身によって抽象化レイヤ21006への明示的な呼び出しが行われるかして、信頼性の境界を拡張かつ縮小してもよいと当業者は分かるであろう。先行技術によると、JARは、Javaバイトコードベースのモジュールだけでなく、例えばECMAスクリプト（Javaスクリプト）など、その他の言語モジュールを含んでもよい。これらを、http://java.sun.com/j2se/1.3/docs/tooldocs/win32/jarsigner.htmlのSunからjarsignerツールを用いて署名し、その署名を、http://java.sun.com/j2se/1.4.2/docs/api/java/util/jar/JarFile.htmlに記載されたようなjava.util.jar.JarFileクラスを用いて検証してもよい。この場合、好ましい実施形態において、図２７に示されたモジュールハンドル2708は、コンポーネントを含むJARファイルを参照するハンドルである。

【０１７９】

図３１に、tPCRサポートモジュール21002のAPI、TPCR_VerifyPSCAndExtend 21028の詳細に関するフローチャートを示す。本関数は、検証および記録するためのPSCをパラメータとしてわたして21100で開始し、次の図３２で示す、拡張PSCツリーの各ノードに対してtPCR状態を算出する21102サブルーチンを呼び出す。次に、現在のtPCR状態とソリューションリストの変数とを、NULLに初期化する21104。これら２つの変数の利用法については図３３で説明する。そして、渡されたPSCのtPCR値に、現在の拡張PSCツリーに記述された状態から達することができることを検証するサブルーチンが呼び出される21106。戻りコードをテストすることで、本関数が、拡張PSCツリー内で、渡されたPSCが保持する検証用の状態にtPCRを設定したノードを１以上見つけたかどうか分かる21108。この設定された親が見つからなかった場合、プロセスは、呼び出しルーチンへ拡張できなかったことを示すエラーコードを返す21110。見つかった場合は、渡されたPSCを、ソリューションリストに記述されたノードに設定した先のPSCを有する拡張PSCツリーに付け加え21112、プロセスは、呼び出しルーチンにサクセスコードを返す21114。

【０１８０】
図３２に、図２６における拡張PSCツリーの各ノードのtPCR状態2604をどのように算出するか説明するフローチャートを示す。当該フローチャートは引数なしに呼び出され21200、行きがけ順に拡張PSCツリーの走査を行う21202ことで処理が開始する。この走査は、ツリーのノードを続く処理にとって望ましい順序で収集するためにルートから始まる。好ましい実装では、ブーストグラフライブラリ関数breadth_first_search()を用いてこれらのノードを収集する。次に、記録されている各ノード21204に対し、深さ優先走査によって現在のtPCR状態2604をNULLに設定する21206。セキュアブート完了の際に定義されたtPCRの開始値、好ましい実施形態では０の値、に初期化されたtPCRを検証する証明書の特別処理が必要なので、現在の証明書に格納された検証用PCR状態2404のtPCR対ごとに、その値をセキュアブート完了の際に定義されたtPCR初期値、好ましい実施形態では０の値と照合する21210。それらが等しい場合は、tPCRのインデックスおよび値の対をこのノードのtPCR状態に付け加え21212、次のtPCRに対してループを続ける。そうでない場合は、ノードのtPCR状態を付け加えずにループを続ける。各tPCRをチェックすると、本関数は、現在のノードの親それぞれに対するループへと移行する21214。親ノードが拡張PSCツリーのルートノードであれば21216、前回のループがこの特殊ケースを処理したため、何も行われない。そうでない場合には、親ノードのtPCR状態2604を問い合わせ、コピーする21218。親の拡張PSC名2602により参照されるPSCによって定義された拡張動作が親の状態のコピー21220において行われ、その結果のtPCR状態が現在のノードのtPCR状態に付加される21222。この拡張PSCツリー構築方法により、２つの親が同じtPCRに対して異なる値を指定する事態は決しておこらない。そのため、当業者は、この状況をチェックする必要はないが、検証目的で実装してもよいと分かるであろう。tPCR状態の収集は、前述したように、全ての親に対して繰り返され、終了すれば、走査中の次のノード21204が処理される。このように各ノードが処理されると、各ノードのtPCR状態を返してプロセスは終了する21224。

【０１８１】
当業者は、例えばbfs_visitor’s examine_vertex()でステップ21204から21222を実行するなど、上記アルゴリズムを実行する別の方法が存在するので、ノードの別々のリストの必要性はなくなると分かるであろう。さらに、この関数は、PSC関連の動作が行われるたびに呼び出されるが、その値をキャッシュして再計算にかかる労力を削減してもよい。

【０１８２】
図３３に、ノードごとのtPCR状態が算出された後、あるPSCに定義された状態へtPCRを設定する拡張済みのPSC全てのリストを見つけることによって、どのようにしてそのPSCを検証するか説明するフローチャートを示す。この図で説明するルーチンは、再帰的なルーチンであることに注意されたい。このルーチンへのエントリーポイントは、引数として、リスト形式の検証用tPCR状態と、現在のtPCRの一致の状況と、tPCR状態によってPSCの親だと判明した拡張PSCツリーのノードのリストとをとる21300。ソリューション方法の概要は、tPCRのインデックスおよび値の対ごとに、望ましい値を現在のtPCRに拡張する証明書であり、かつ、このソリューションの一部である別のtPCRに拡張する他の証明書と互換性をもつ証明書を見つけようと試みることである。候補が見つかれば、ルーチンを再帰的に呼び出して、PSCの一部である別のtPCRを一致の状態へ拡張する他の証明書を見つける。

【０１８３】
最初のステップは、一致するtPCRのリストをチェックすることである。これが空である場合21302、ルーチンはリストの最後に正常に再帰されたので、呼び出し側に成功を示すためFOUND値を返す21304。そうでない場合は、tPCRリストの先頭を削除し、これを現在のtPCRとして用いて親の証明書を見つけようと試みる21306。ソリューションリストにまだ割り当てられていない拡張PSCツリーの各ノードを親になる候補として選択する21308。先行技術に従ってこれらのノードをどのように取得できるかは図３２の説明で示した。まず、２つの構造体において一致するtPCRインデックスが同じtPCR値であることを検証することによって、このノードのtPCR状態を、渡された現在の一致tPCR状態と互換性があるかどうかチェックする21312。値が一致しなければ21316、ルーチンは、拡張PSCツリーの次の

ノードへ移行する21308。一致する場合は、ノードに格納された拡張PSC名2602を用いて検証用ノードのPSCを検索し21314、拡張用PSCのインデックス2410を21306で取り出した現在のtPCRのインデックスと比較する。インデックスが一致しない場合21316、ルーチンは、拡張PSCツリーの次のノードへ移行する21308。インデックスが一致する場合は、親ノード候補が見つかったので、このノードをソリューションリストに付け加え21318、このノードの状態を拡張して現在のtPCRリストにマージする。tPCR検証ルーチンを、短縮したtPCRリストと現在のtPCR状態とソリューションリストとを引数にして、再帰的に呼び出す21332。再帰的呼び出しが成功すれば21324、呼び出し側にその成功を示すためFOUND値を返す21326。失敗した場合は、現在のノードをソリューションリストから削除して21320での状態のマージを取り消し21328、そして、ツリーの次のノードを調べるためにループを続ける。全てのノードが調べられ、一致がなかった場合は、NOTFOUNDを返す21310。

【０１８４】

図３４に、取り消し前と取り消し後のサンプル拡張PSCツリー2206の状態を示す。取り消し前の状態21400は図２３Ｂで述べたとおり、図２３Ａのモジュールツリーから構築された状態である。ところで、図３０で示されたように、ユーザからの作用、プログラムバグ、または改ざん検出により第１プラグインが終了すれば、オペレーションシステムは第１プラグインの終了を検出して、「第１プラグイン開始」証明書2358は起動時にテストされたPSCであったと判断するので、証明書の拡張を取り消す必要がある。「第１プラグイン開始」2358とともに、それに従属する全ての証明書、すなわち、「第１プラグインロード」2362、「第１マッシュアップ開始」2366および「第１マッシュアップロード」2368も拡張PSCツリー2206から削除しなければならない。その結果、取り消し後の状態21402の拡張PSCツリー2206となる。

【０１８５】

図３５に、拡張プロセスをどのようにして取り消すか説明するフローチャートを示す。本関数は、引数として、取り消し対象PSCをとる21500。まず、本関数は、上記の図３２で示された、拡張PSCツリーのノードごとにtPCR状態を算出する21502サブルーチンを呼び出す。次に、対象PSCのPSC名2402とツリー内の各ノードの拡張PSC名2602との一致を探すことで、拡張PSCツリー内で対象PSCへの参照を検索する21504。一致するノードが見つからない場合21506、エラーコードを呼び出し側に返して取り消すことができなかったことを示す21512。次に、対象PSCの検証用PCR状態2404を見つかったノードのtPCR状態2604とを比較する。それらの状態が等しくない場合21510、エラーコードを呼び出し側に返して取り消すことができなかったことを示す21512。状態が等しい場合は、見つかったノードとそれに従属するもの全てを削除する関数を呼び出し、その関数がサクセスコードを呼び出し側に返す21516。

【０１８６】

図３６に、取り消し処理が拡張PSCツリーからどのようにしてノードを削除するか説明するフローチャートを示す。本関数は、引数として、ツリーから削除するノードをとる21600。まず、このノードの子PSC全てに対してループし21602、再帰的に自分自身を呼び出してその子それぞれを順に削除する21604。全ての子が削除された時点で、ノード自身を削除し21606、本関数はリターンする21608。このようにして、終了中のモジュールおよびそれに従属する信頼性のあるモジュール全てをカバーする、以前の拡張処理によって確立された信頼性の境界は、終了する必要のないモジュールをカバーしたまま、装置のアプリケーション空間における信頼性レベルを落とすことなく、終了するモジュールを除外するように縮小される。

【０１８７】

（第３の実施形態）

本発明の第３の実施形態は、リモート認証のためのものである。先行技術によると、リモート認証のプロセスには異なる段階が２つある。まず、共有されるAIK、すなわち、認証識別キー（Attestation Identity Key）が、TPM v1.2にある匿名認証（Direct Anonymous Attestation）プロトコルをおそらく用いて、装置上のクライアントとリモートサーバ

との間で確立される。次のステップは、このAIKを用いて特定のデバイス構成の認証を行うことである。図３７Ａおよび図３７Ｂに、当該リモート認証の先行技術を示す。装置2120とその構成要素とは、図１１Ａおよび図１１Ｂで示されたものに、セキュア処理環境2114内に格納されたAIK21710を加えたものである。図３７Ａに、当該システムのセキュアブートをサポートする場合における先行技術を示し、本発明にとって重要な２つの要素を含むサーバ21700も示す。認証プロセスを制御する認証器21702があり、これは、先に確立されており、装置2120のAIK21710と関連付けられているAIK証明書21704を用いる。認証器21702は、認証要求21706を生成し、認証を要求したアプリケーション2100にその認証要求を送信する。アプリケーション2100における認証プロセスでは、認証要求21708をセキュアブートコンポーネント2113に送信し、セキュアブートコンポーネント2113はセキュア処理環境2114と連携して先行技術で定義したような認証を実行する。

【０１８８】

図３７Ｂも同様に、セキュアモードのサポートはないが、「TCG Specification Architecture Overview Revision 」に記載した手段などの独立したセキュア処理環境2114を有し、セキュアモードインターフェースが存在しない場合における先行技術を示す。前述したように、本発明にとって重要な２つの要素を含むサーバ21700も存在する。認証プロセスを制御する認証器21702があり、これは、先に確立されており、装置2120のAIK21710と関連付けられているAIK証明書21704を用いる。認証器21702は、認証要求21706を生成し、認証を要求したアプリケーション2100にその認証要求を送信する。アプリケーション2100における認証プロセスでは、認証要求21705をトラステッド/セキュアブートコンポーネント2152に送信し、セキュアブートコンポーネントはセキュア処理環境2114と連携して先行技術で定義したような認証を実行する。

【０１８９】

図３８Ａに、図３７Ａの先行技術に基づいた、一時的PCRに対するリモート認証のための本発明における第３の実施形態を示す。サーバ21700は前述したものであり、前述した認証要求21706は、アプリケーション2100に対して行われる。しかしながら、第３の実施形態では、セキュアブートコンポーネント2113へ認証要求21708を直接行う代わりに、サーバへ返される認証情報にtPCRサポート2204からの情報も含めることができるよう、システムのレイヤ全てを介して認証要求21800を行う。

【０１９０】

図３８Ｂに、図３７Ｂの先行技術に基づいた、一時的PCRに対するリモート認証のための本発明における第３の実施形態を示す。サーバ21700は前述したものであり、前述した認証要求21706は、アプリケーション2100に対して行われる。しかしながら、第３の実施形態では、トラステッド/セキュアブートコンポーネント2152へ認証要求21708を直接行う代わりに、サーバに返される認証情報にtPCRサポート2204からの情報も含めることができるよう、システムのレイヤ全てを介して認証要求21850を行う。

【０１９１】

図３９に、アプリケーションがリモート認証する間のモジュール間通信について説明するシーケンス図を示す。相互に作用する6つのオブジェクト、21004、21002、21006、21008、21010および21900を図示している。まず、SPE21004は先行技術に係るセキュア処理環境である。好ましい実施形態では、それはMTMである。次に、tPCRサポート21002が存在する。これは、tPCRを参照するPSCをとって任意のPSCに基づくtPCR状態を検証する処理を行い、有効であれば、証明書を拡張したことを記録する。抽象化レイヤ21006は、通常モードのアプリケーションからの要求を取り扱い、要求を他のモジュールにわたす。OS21008はオペレーティングシステムであり、ここでは、一時的PCRを正確にアップデートするようなアプリケーションの起動および終了の処理に関するシステムである。アプリケーション21010は、リモート認証を要求するアプリケーションである。そして、サーバ21900はリモート認証を行う。

【０１９２】

まず、アプリケーション21010は、抽象化レイヤ21006にクライアントノンス$N_c$ を要求

し21901、このランダムに生成された値が返ってくる21902。この値は、アプリケーション21010がサーバ21900に認証を要求する場合に用いられる21903。例えば、アプリケーション21010が保護されたサービスへのアクセスを許可する前に、サーバ21900はアプリケーション21010が期待された環境で動作しているか確認する必要があるため、アプリケーション21010はサーバ21900からこの許可を得るため認証手順を開始する。アプリケーション21010は、生成されたクライアントノンス$N_c$をサーバ21900に渡す。このクライアントノンスは、アプリケーション21010およびサーバ21900間の通信ストリームに対するリプレイ攻撃やその他の攻撃に対する保護を実現するための値である。サーバ21900は、クエリ用のサーバノンス$N_s$とランダムに生成されたChallengeと物理的PCRセットとを含むメッセージを付けた認証要求を送信することによって応答する21904。このメッセージは、先行技術で述べられているような匿名証明プロトコルなどでクライアントとサーバとの間において先に確立されたAIKを用いて署名される。ここで、好ましい実施形態では、このメッセージフォーマットはTCGで指定されたものと同じである。アプリケーション21010は、この認証要求の処理21906をOS21008に託す。OS21008は、図３０で述べたようなプロセス空間の情報を用いて、どのアプリケーションまたはダイナミックロードライブラリが関数を呼び出したのかと、モジュールがどのPSCを使ってそのモジュール自身の検証を行ったのかとを判断し21908、アプリケーションのリモート認証用の先に確立されたAIKを決定する21909。検索されたPSCは、モジュールへ認証要求するために、他の認証パラメータとともに抽象化レイヤ21006へ渡される21910。ここで、認証が開始可能になる。まず、認証用のサーバノンスとランダムChallengeと物理的PCRとを含むメッセージの署名が、先行技術に基づき先に確立されたAIKを用いてSPE21004によって検証される21912。次に、SPE21004を再び用いて、今度は、アプリケーション21010に対するPSCの完全性を検証し21914、そして、tPCRサポート21002を用いて前記PSC内のtPCRセットの検証を行う21916。これらのチェックが正常に行われたとすると、抽象化レイヤ21006はSPE_Quoteで用いられるハッシュ値を作成する21918。このハッシュは、先にサーバへ送信されたクライアントノンスとサーバノンスと21910で渡されたChallengeとを連結したものおよびアプリケーション21010のPSCに格納されている一時的PCRハッシュから算出される。SPE21004が、サーバから受信したPCRセレクション21904に対応したPCRセットと、21918で算出されたハッシュ値とを含み、確立されたAIKのプライベート部分を用いて署名された署名済みハッシュの生成を要求される21922。SPE21004がMTMである第３の実施形態において、SPE_QuoteはTPM_Quoteのエイリアスであり、TCG仕様書で定義されたように動作する。この得られた署名値は、SPE21004からサーバ21900へ、21922、21924、21926そして21928と遡って渡される。サーバ21900は、渡された結果が期待される結果と等しいかどうかを検証し21930、等しければ、アプリケーション21010に認証が正常に完了したことを通知する21932。

【０１９３】

第３の実施形態では、アプリケーション21010およびサーバ21900間の通信（21903、21904、21928、21930および21932）は、インターネットを介したワイヤレスリンクで行われるが、固定リンクや無線リンクを用いる実施形態でも可能であると当業者は分かるであろう。この通信のプロトコルは、メッセージコンテンツを暗号化する必要がないように設計されているが、SSLなどの暗号化プロトコルを用いる実施形態でも可能であると当業者は分かるであろう。

【０１９４】

もう1つの方法として、tPCR値に対するリモート認証のみを要求しても構わない。図４０に、この場合のリモート認証で用いられるQuote情報記録22000の構造体を示す。versionフィールド22002には、定値1.1.0.0と定義されたバージョン表示が含まれる。fixedフィールド22004には、定値「QUOT」と定義された構造体タイプの識別子が含まれる。digestValueフィールド22006には、認証されるtPCRダイジェスト値が含まれる。externalDataフィールド22008には、クライアントノンスとサーバノンスとChallenge値とを連結したもののハッシュとしてリモート認証プロトコルにより定義された外部データが含まれる。そして、signatureフィールド22010には、上記のフィールドの暗号化署名が含まれる。この署

名は、先に確立されたAIKとしてリモート認証プロトコルにより定義された、署名生成ルーチンに渡されたキーリファレンスを用いて生成される。

【０１９５】

図４１に、tPCRのみに対し、アプリケーションが図４０で示したQuote情報構造体22000を用いてリモート認証する間のモジュール間通信を示す。この通信シーケンスの第１段階は、図３９で示したシーケンスと全く同じである。前述したように、抽象化レイヤ21006は、SPE21004を用いて、認証要求の署名を検証し21912、一時的PCRのPSCの完全性を検証する21914。そして、tPCRサポートを用いてPSC内の実tPCR値を検証する21916。前述した検証が正常に行われた場合は、ここから、図３９とシーケンスが異なってくる。抽象化レイヤ21008は、Q1と名付けられたQuote情報構造体22000を作成し22100、versionフィールド22002およびfixedフィールド22004をそれぞれ予め定義した値へ初期化する22102。次に、digestValueフィールドを先に検証されたPSC内のフィールドのダイジェスト値へ設定する22106。このダイジェスト値は、検証用PCR状態2404のPCR値$_n$フィールド2408全てのハッシュ値をまとめて計算することによって算出される。次に、externalDataフィールド22008を、クライアントノンスとサーバノンスとChallenge値とを連結したもののハッシュ値へ設定する22106。これらのうちサーバノンスとChallenge値は、21910で抽象化レイヤから渡されたものである。全てのデータが正しく設定されると、抽象化レイヤ21006は、Quote情報22000の最初の4つのフィールドを暗号化署名用のデータとして、最後のフィールドを署名用のキーとしてAIKを用いて署名を保存する場所であるsignature 22010として、SPE21004のSPE_Sign関数を呼び出す。SPE_Sign関数22108の動作は、TPM_Sign APIの先行技術において詳述されたとおりである。この結果は、signatureフィールド22010に置かれ、抽象化レイヤ21006に返される22110。そして、このQuote情報構造体22000一式は、抽象化レイヤ21006からサーバ21900へ、22112、22114、22116へと遡って渡される。サーバ21900は、渡された結果が期待される結果と等しいかどうかを検証し22118、等しければ、アプリケーション21010に認証が正常に完了したことを通知する21932。

【０１９６】

本発明は上記の実施形態に基づいて述べられているが、本発明は明らかにそのような実施形態に限定されるものではない。下記のケースもまた本発明に含まれる。

【０１９７】

(1)上述の実施形態では、MTM仕様書と同様の方法で検証は行われた。しかしながら、本発明を他の検証システムに適用してもよい。ただし、コンポーネントをチェーンのように検証する検証方法で検証システムがシステムのコンポーネントを検証できる場合（例えば、あるコンポーネントがそのコンポーネントの後に起動する他のコンポーネントを検証する）に限る。例えば、ハッシュ値のMTMへの拡張などの動作はTCG仕様書に固有なものであるため、省略してもよい。

【０１９８】

(２)上述の実施形態では、証明書(RIM証明書)内でハッシュ値を使用することにより検証は行われた。しかしながら、ハッシュ値を使用しない他の検証方法が本発明に適用されてもよい。

【０１９９】

従来のチェックサムや、コンポーネントから抽出される別のデータ(例えば、コンポーネントから抽出される第１所定ビット)が、検証を行うために使用されてもよい。また、証明書は、完全性チェック値を含むデータの集合に置き換えられてもよい。

【０２００】

さらに、検証方法は、コンポーネントから抽出される値と期待される値が等しいか否かをチェックすることに限定されない。例えば、コンポーネントのサイズを確認し、もしサイズが所定量より大きいか、または小さいなら、コンポーネントは検証されたと判断されるとしてもよい。これらの検証方法は、ハッシュ値と、その期待値とを比較するほど厳密なものではないが、それよりも高速に行われる。

【０２０１】

（３）上記の各装置は、具体的には、マイクロプロセッサ、ＲＯＭ、ＲＡＭ、ハードディスクユニット、ディスプレイユニット、キーボード、マウスなどから構成されるコンピュータシステムである。ＲＡＭまたはハードディスクユニットには、コンピュータプログラムが記憶されている。マイクロプロセッサが、コンピュータプログラムにしたがって動作することにより、各装置は、その機能を達成する。ここでコンピュータプログラムは、コンピュータに対する指令を示す命令コードが複数個組み合わされて構成されたものである。

【０２０２】

（４）上記の各装置を構成する構成要素の一部または全部は、１個のシステムＬＳＩ（Large Scale Integration：大規模集積回路）から構成されているとしてもよい。システムＬＳＩは、複数の構成部を１個のチップ上に集積して製造された超多機能ＬＳＩであり、具体的には、マイクロプロセッサ、ＲＯＭ、ＲＡＭなどを含んで構成されるコンピュータシステムである。ＲＡＭには、コンピュータプログラムが記憶されている。マイクロプロセッサが、コンピュータプログラムにしたがって動作することにより、システムLSIは、その機能を達成する。

【０２０３】

さらに、各装置を構成する構成部品の各ユニットは、別個の個別チップとして、または、一部もしくは全てを含む単一のチップとして作られてもよい。

【０２０４】

さらに、ここでは、LSIが述べられているが、集積化の程度の違いにより、指定IC、LSI、スーパLSI、ウルトラLSIが使用される場合もある。

【０２０５】

さらに、回路の集積化の手段は、LSIに限定されるものではなく、専用回路、もしくは汎用プロセッサによる実装も利用できる。さらに、LSIが製造された後にプログラム可能なフィールドプログラムゲートアレイ(FPGA)を使用することもまた可能であり、またLSI内で回路セルの接続および設定を再構成可能なリコンフィギュアラブルプロセッサも使用可能である。

【０２０６】

さらに、もしLSIに置き換わる集積回路技術が半導体技術もしくは他の派生する技術の進歩を通じて現われるのであれば、その技術は当然に構成要素の集積化を実現するために使用することができる。バイオ技術の適用が予想される。

【０２０７】

（５）上記の各装置を構成する構成要素の一部または全部は、各装置に脱着可能なＩＣカードまたは単体のモジュールから構成されているとしてもよい。ＩＣカードまたはモジュールは、マイクロプロセッサ、ＲＯＭ、ＲＡＭなどから構成されるコンピュータシステムである。ＩＣカードまたはモジュールは、上記の超多機能ＬＳＩに含まれるとしてもよい。マイクロプロセッサが、コンピュータプログラムにしたがって動作することにより、ＩＣカードまたはモジュールは、その機能を達成する。このＩＣカードまたはこのモジュールは、耐タンパ性を有するとしてもよい。

【０２０８】

（６）本発明は、上記に示す方法をコンピュータにより実現するコンピュータプログラムであるとしてもよいし、コンピュータプログラムなどのデジタル信号であるとしてもよい。

【０２０９】

また、本発明は、コンピュータプログラムまたはデジタル信号をコンピュータ読み取り可能な記録媒体、例えば、フレキシブルディスク、ハードディスク、ＣＤ－ＲＯＭ、ＭＯ、ＤＶＤ、ＤＶＤ－ＲＯＭ、ＤＶＤ－ＲＡＭ、ＢＤ（Blu-ray Disc）、半導体メモリなどに記録したものとしてもよい。また、これらの記録媒体に記録されているデジタル信号であるとしてもよい。

【０２１０】

また、本発明は、コンピュータプログラムまたはデジタル信号を、電気通信回線、無線または有線通信回線、インターネットを代表とするネットワーク、データ放送等を経由して伝送するものとしてもよい。

【０２１１】

また、本発明は、マイクロプロセッサとメモリを備えたコンピュータシステムであって、メモリは、上記コンピュータプログラムを記憶しており、マイクロプロセッサは、コンピュータプログラムにしたがって動作するとしてもよい。

【０２１２】

また、プログラムまたはデジタル信号を記録媒体に記録して移送することにより、またはプログラムまたはデジタル信号を、ネットワーク等を経由して移送することにより、独立した他のコンピュータシステムにより実施するとしてもよい。

【０２１３】

（７）本技術分野の当業者にとっては、説明した実施形態の、本発明固有の教示および利点から原理的に離れることのない多くの変形がありえることを容易に理解することができるだろう。また、上記変更および実施形態の任意の組み合わせは、本発明の範囲内に含まれる。

【産業上の利用可能性】

【０２１４】

この構成によると、情報処理装置は、複数のモジュールのうちどれがアクティブモジュールであるかを示す情報を管理し、そして、そのアクティブモジュールの期待プラットフォーム情報を蓄積することによって蓄積プラットフォーム情報を生成する。

【０２１５】

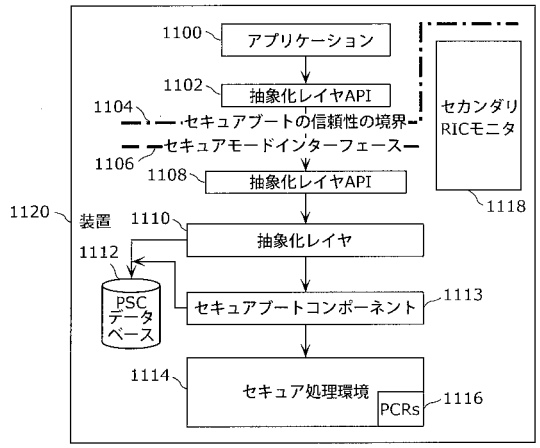
したがって、当該情報処理装置は、アクティブモジュール全てに対応する蓄積プラットフォーム情報を生成することができる。ゆえに、ロード対象の第１モジュールの期待プラットフォーム情報と蓄積プラットフォーム情報を比較して検証を行うことによって、情報処理装置は、第１モジュールを正常にロードする前にロード対象の全てのモジュールを検証することができる。さらに、複数のモジュールのうちどれがアクティブモジュールであるかを管理することによって、１以上のモジュールが終了した後であっても、情報処理装置が、現在の信頼性の境界に従って（PCR値に対応する）蓄積プラットフォーム情報を動的に生成することができる。

【符号の説明】

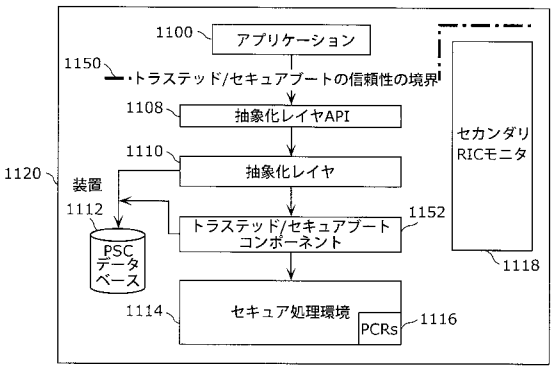
【０２１６】

1100　アプリケーション
1102，1108　抽象化レイヤAPI
1104　セキュアブートの信頼性の境界
1106　セキュアモードインターフェース
1110　抽象化レイヤ
1112　PSCデータベース
1113　セキュアブートコンポーネント
1114　セキュア処理環境
1116　物理的PCRs
1118　セカンダリRICモニタ
1120　装置
1200　OSサポート
1202　コンポーネント・PSCマップ
1204　tPCRサポート
1206　拡張PSCツリー

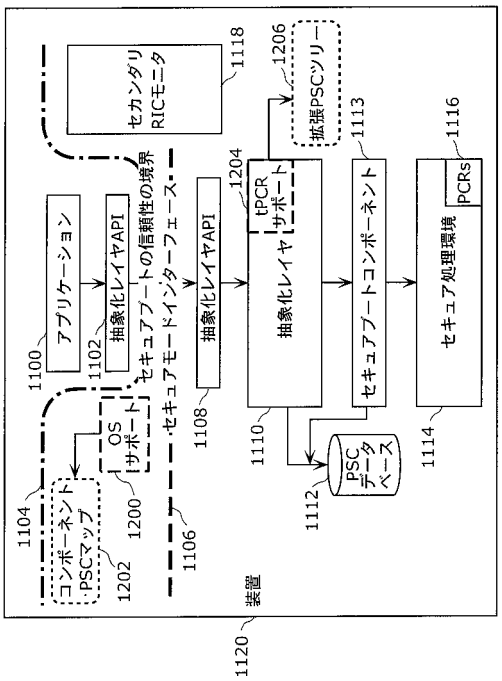
【図１】



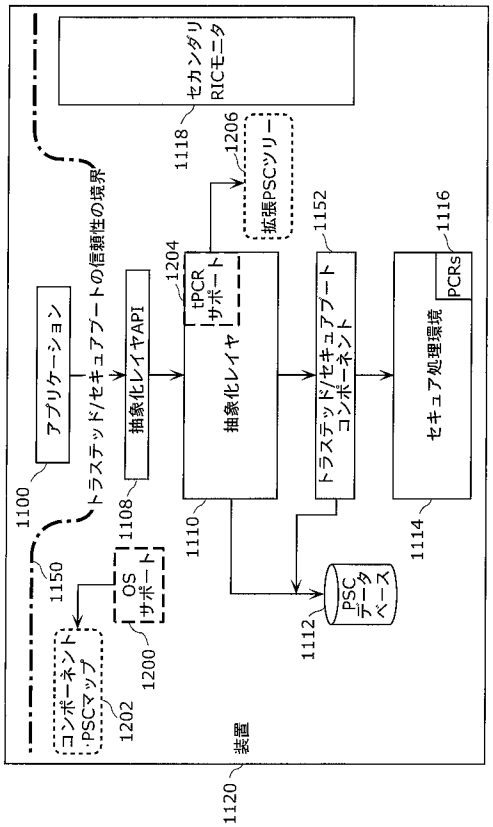

【図１Ａ】



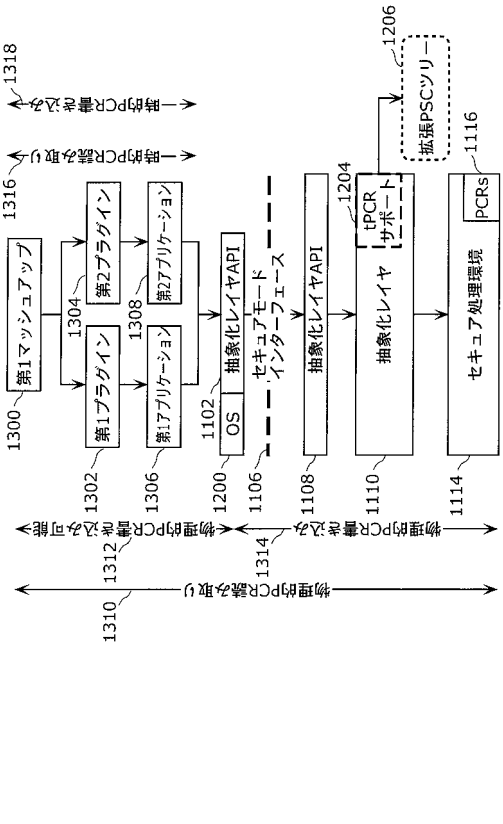

【図２】



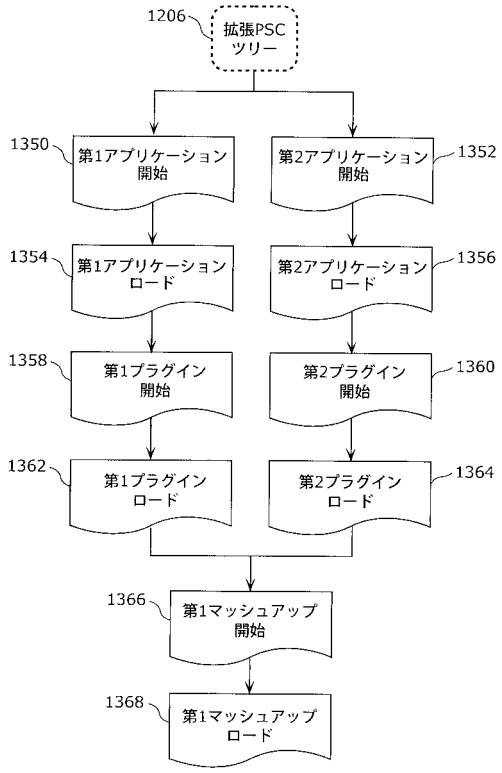

【図２Ａ】



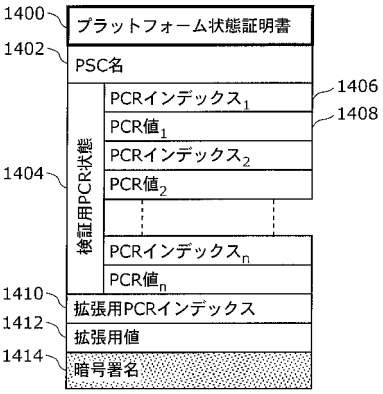

【図３】



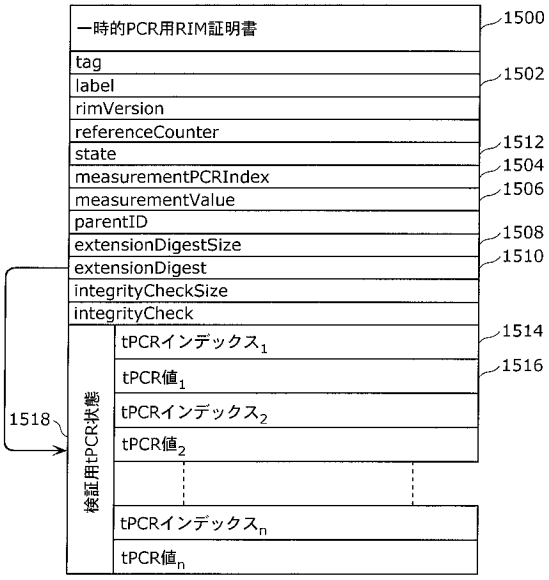
【図３Ａ】



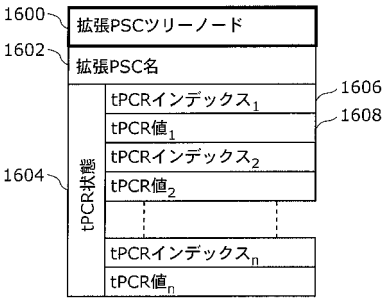

【図４】



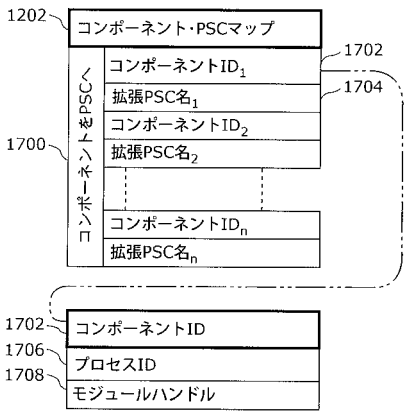

【図５】



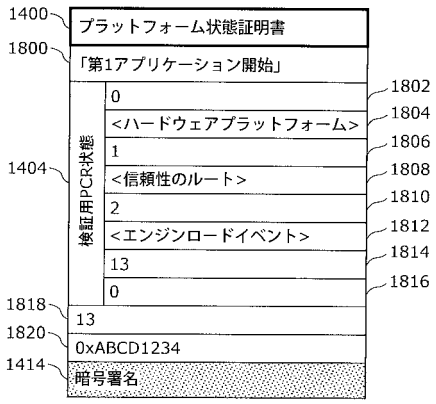

【図６】



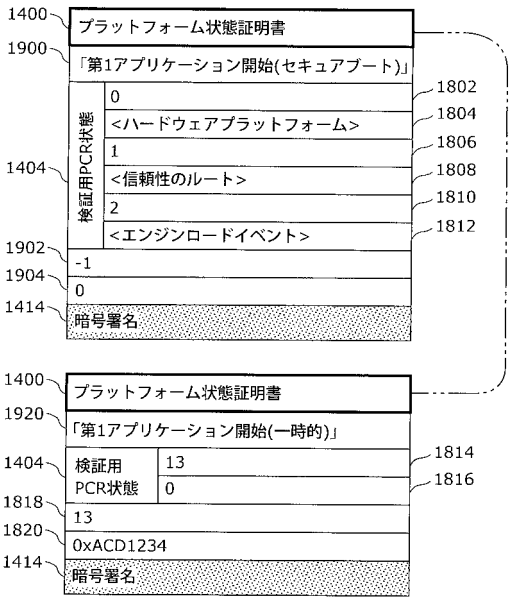

【図７】



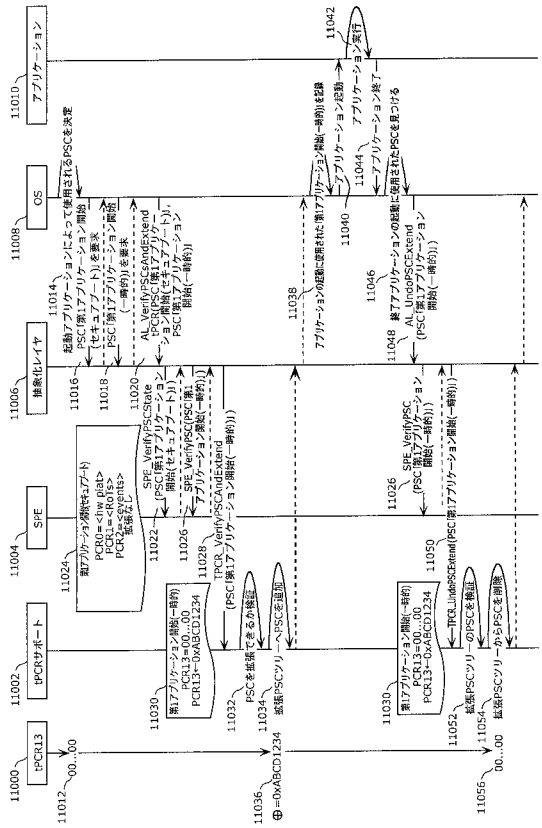
【図８】


1400
プラットフォーム状態証明書
1800
「第1アプリケーション開始」
検証用PCR状態
0
<ハードウェアプラットフォーム>
1
<信頼性のルート>
2
<エンジンロードイベント>
13
0
1802
1804
1806
1808
1810
1812
1814
1816
1404
1818
13
1820
0xABCD1234
1414
暗号署名

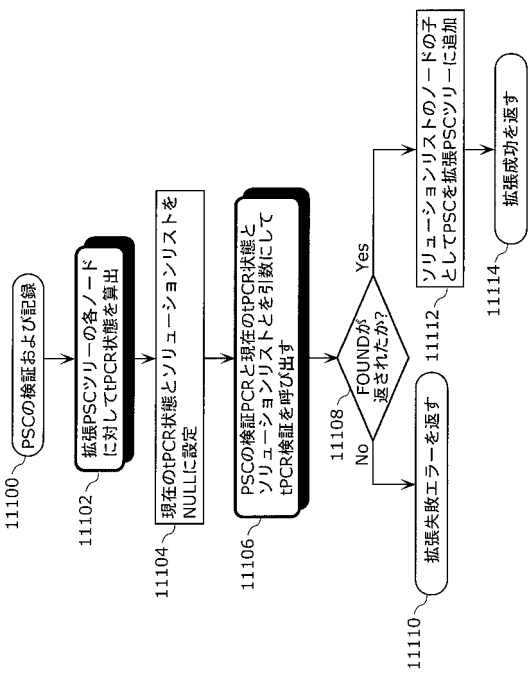
【図９】


1400
プラットフォーム状態証明書
1900
「第1アプリケーション開始(セキュアブート)」
検証用PCR状態
0
<ハードウェアプラットフォーム>
1
<信頼性のルート>
2
<エンジンロードイベント>
1802
1804
1806
1808
1810
1812
1404
1902
-1
1904
0
1414
暗号署名
1400
プラットフォーム状態証明書
1920
「第1アプリケーション開始(一時的)」
検証用
PCR状態
13
0
1814
1816
1404
1818
13
1820
0xACD1234
1414
暗号署名

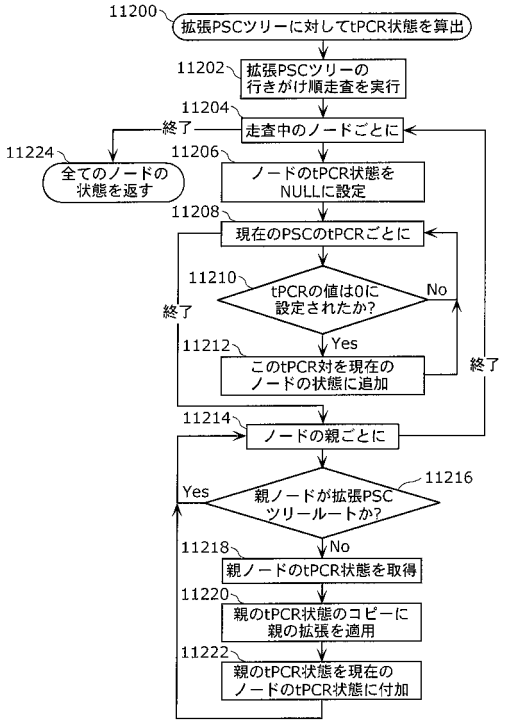
【図１０】


アプリケーション
OS
抽象化レイヤ
S/E
tPCRサポート
tPCR13
11010
11008
11006
11004
11002
11000
11042
アプリケーション実行
アプリケーション起動
アプリケーション終了
11040
11044
11038
アプリケーションの起動に関連付けられた第1アプリケーション開始(一時的)を記録
11046
終了アプリケーションの起動に関連付けられたPSCを見つける
11048
AL_UndoPSCExtend
(PSC「第1アプリケーション開始(一時的)」)
11014
起動アプリケーションによって使用されるPSCを決定
11016
PSC「第1アプリケーション開始(セキュアブート)」を要求
11018
PSC「第1アプリケーション開始(一時的)」を要求
11020
AL_VerifyPSCAndExtend
(PSC「第1アプリケーション開始(セキュアブート)」,
PSC「第1アプリケーション開始(一時的)」)
11022
SPE_VerifyPSC
(PSC「第1アプリケーション開始(セキュアブート)」)
11024
第1アプリケーション開始(セキュアブート)
PCR0=<hw plat>
PCR1=<RoIs>
PCR2=<Elements>
拡張なし
11026
SPE_VerifyPSC
(PSC「第1アプリケーション開始(一時的)」)
11028
TPCR_VerifyPSCAndExtend
(PSC「第1アプリケーション開始(一時的)」)
11030
第1アプリケーション開始(一時的)
PCR13=00...00
PCR13=0xABCD1234
11032
PSCを拡張できるか検証
11034
拡張PSCツリーへPSCを追加
11036
⊕=0xABCD1234
11050
TPCR_UndoPSCExtend(PSC「第1アプリケーション開始(一時的)」)
11052
拡張PSCツリーのPSCを検証
11054
拡張PSCツリーからPSCを削除
11056
00...00
11012
00...00

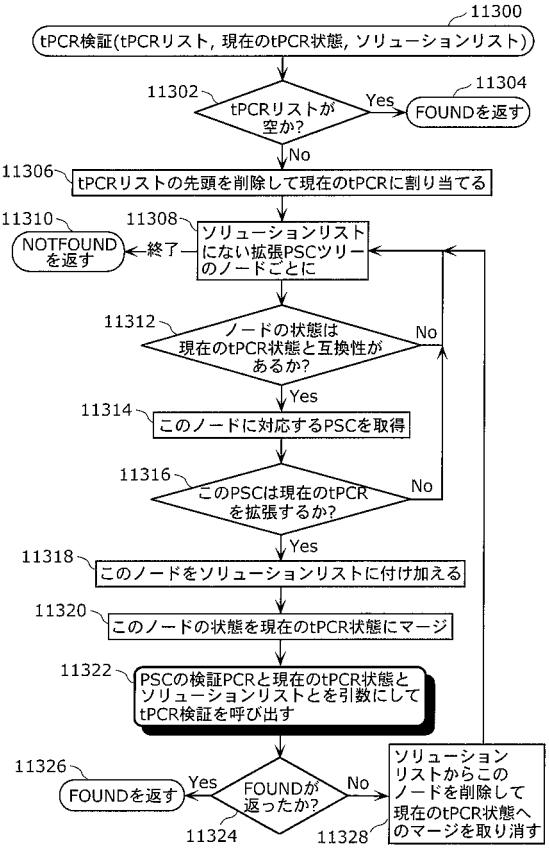
【図１１】


11100
PSCの検証および記録
11102
拡張PSCツリーの各ノードに対してtPCR状態を算出
11104
現在のtPCR状態とソリューションリストをNULLに設定
11106
PSCの検証PCRと現在のtPCR状態とソリューションリストとを引数にしてtPCR検証を呼び出す
11108
FOUNDが返されたか?
Yes
No
11112
ソリューションリストのノードの子としてPSCを拡張PSCツリーに追加
11114
拡張成功を返す
11110
拡張失敗エラーを返す

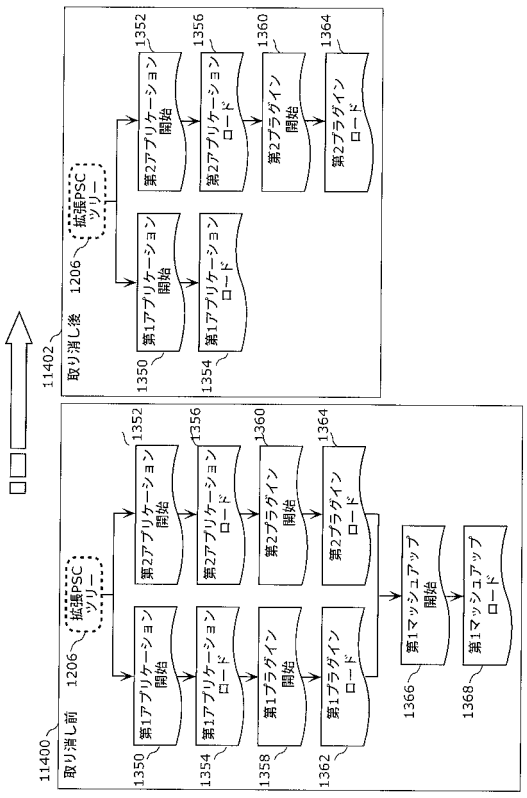
【図１２】



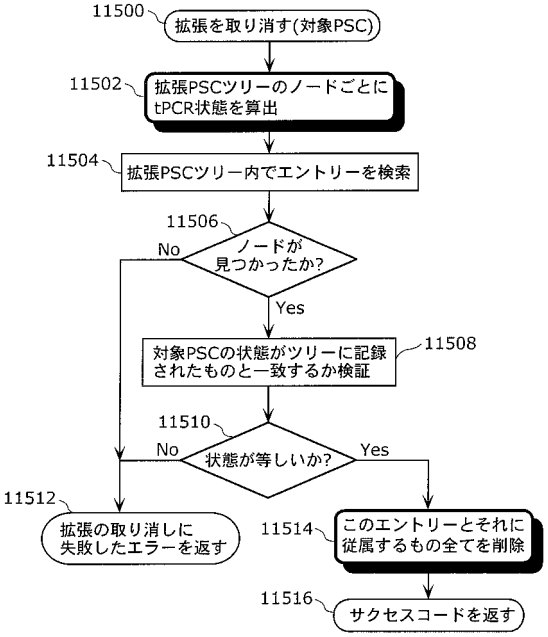

【図１３】



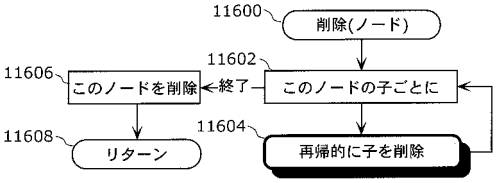

【図１４】



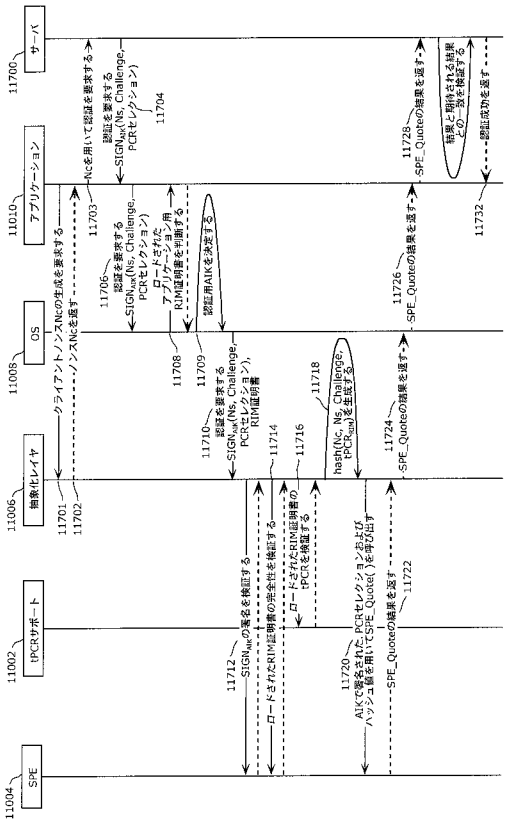

【図１５】



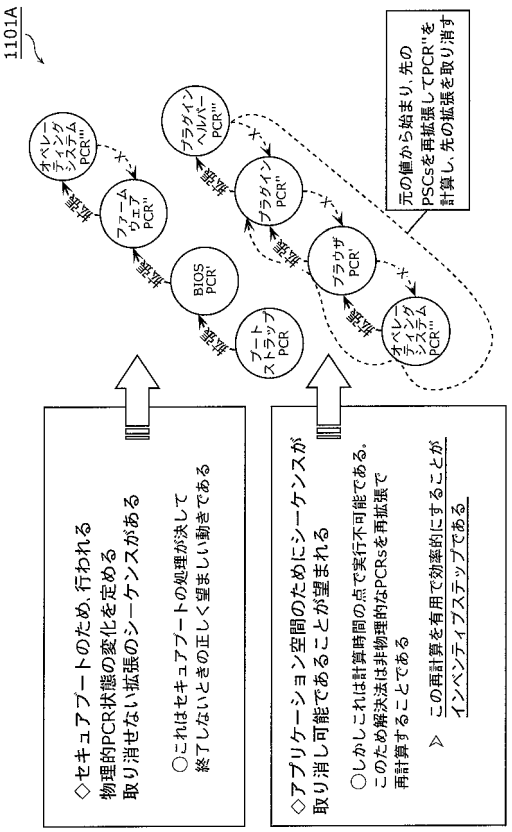

【図１６】



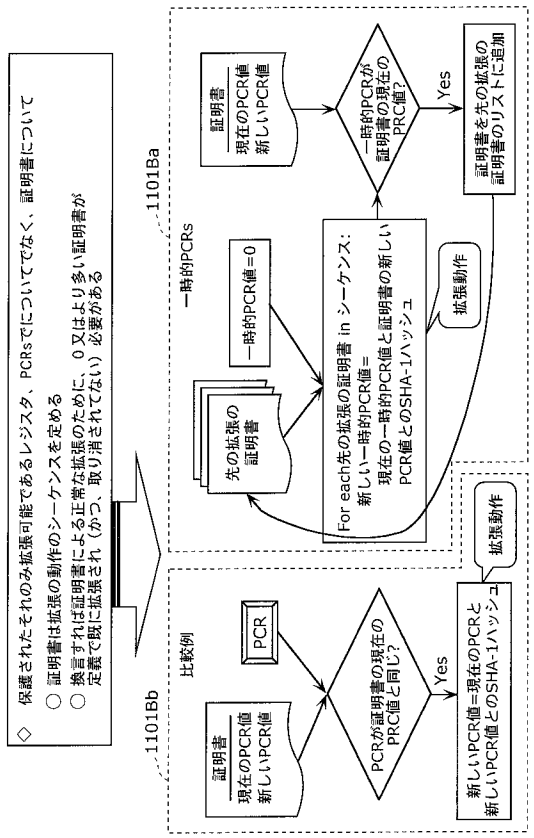
【図１７】



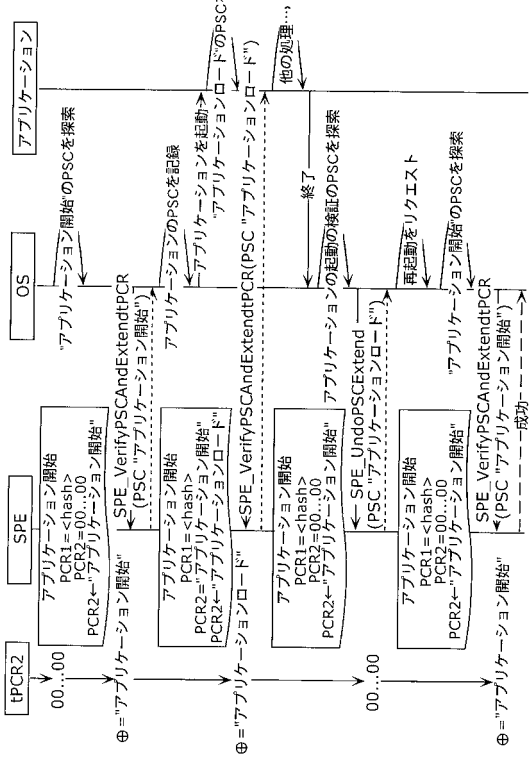

【図１８】



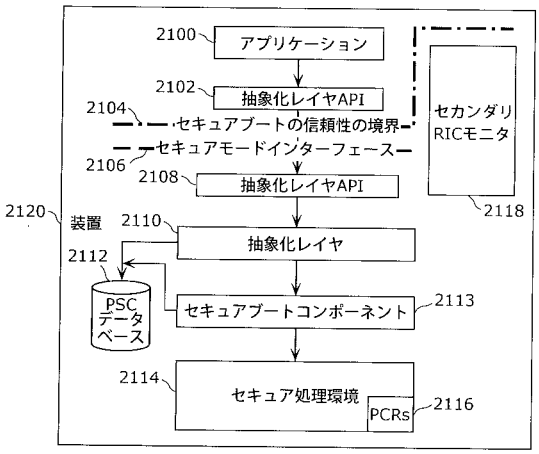

【図１９】



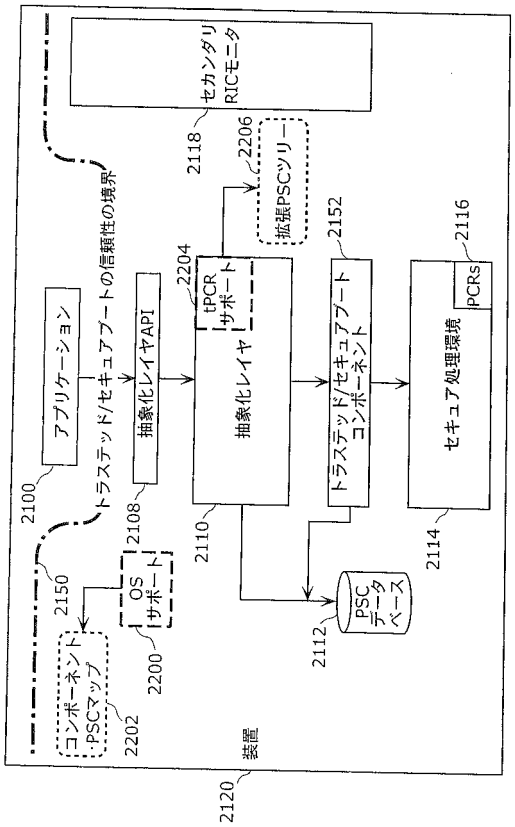

【図２０】



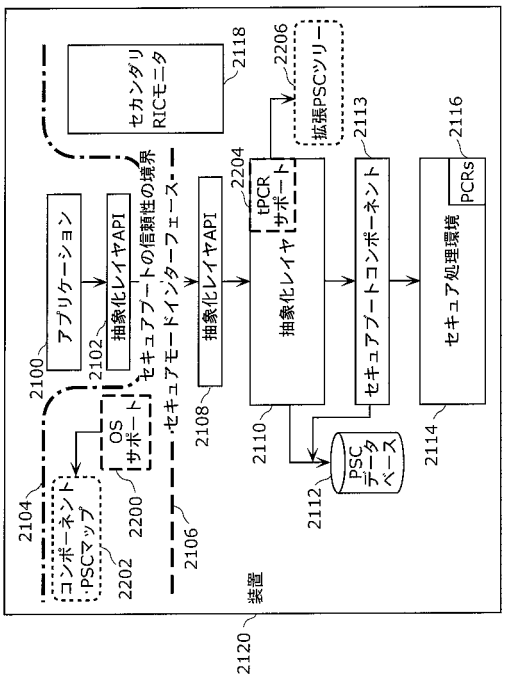
【図２１Ａ】



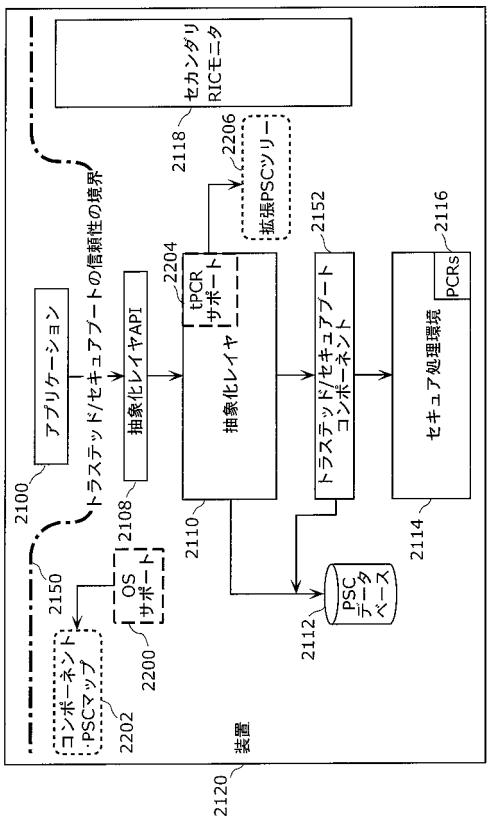

【図２１Ｂ】



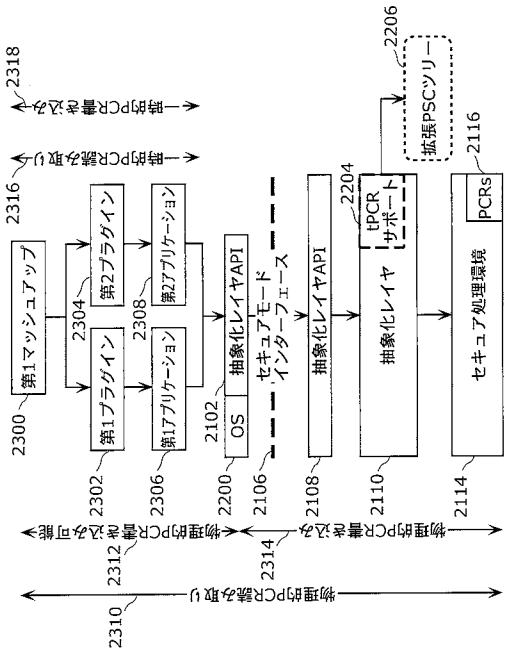

【図２２Ａ】



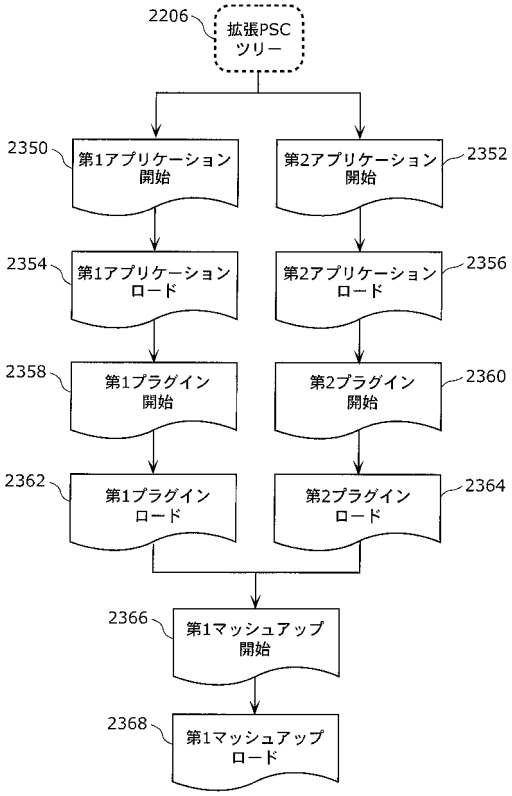

【図２２Ｂ】



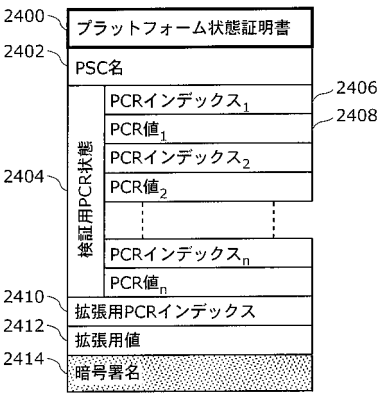
【図２３Ａ】



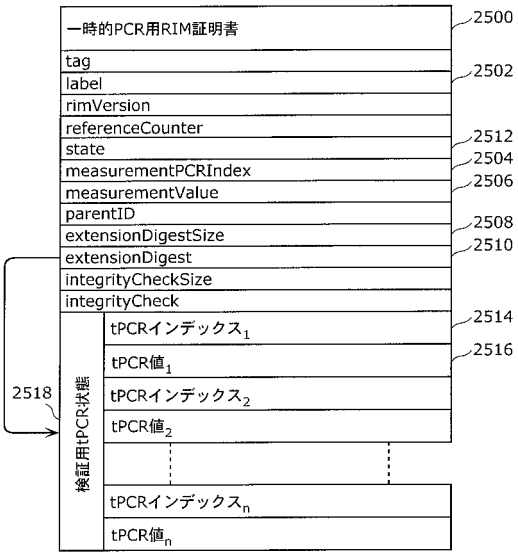

【図２３Ｂ】



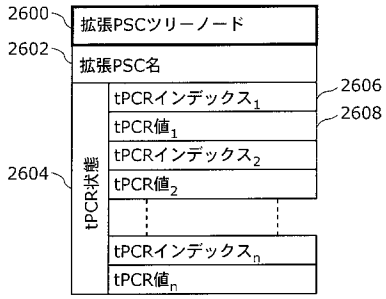

【図２４】



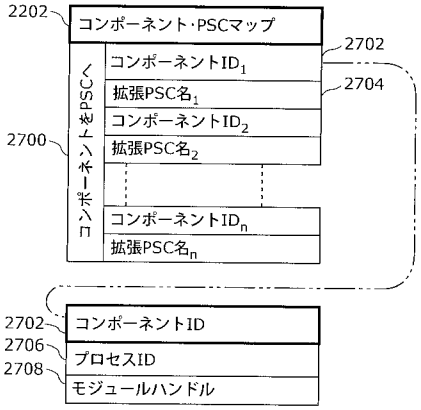

【図２５】



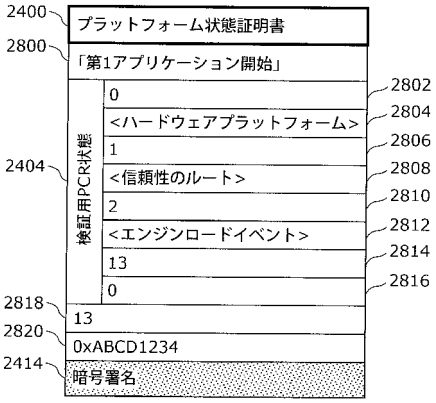
【図２６】



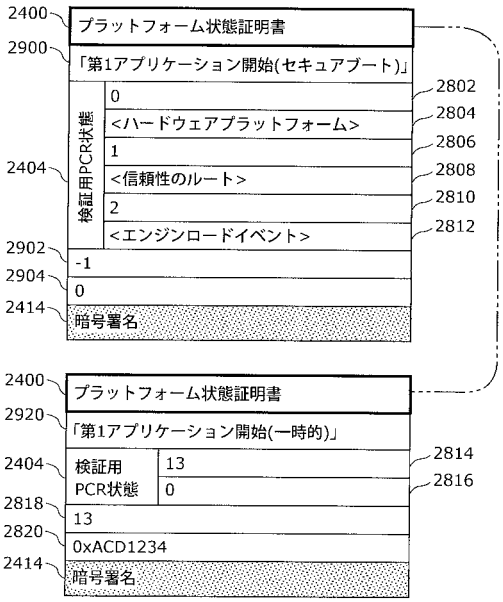

【図２７】



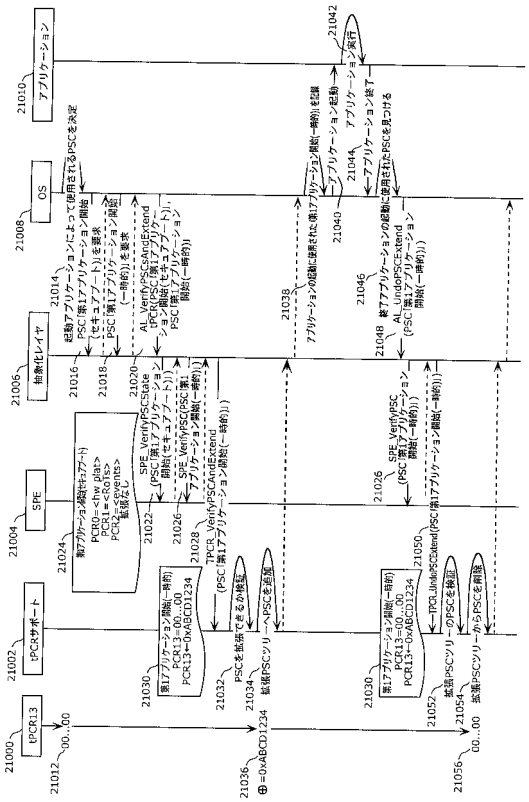

【図２８】



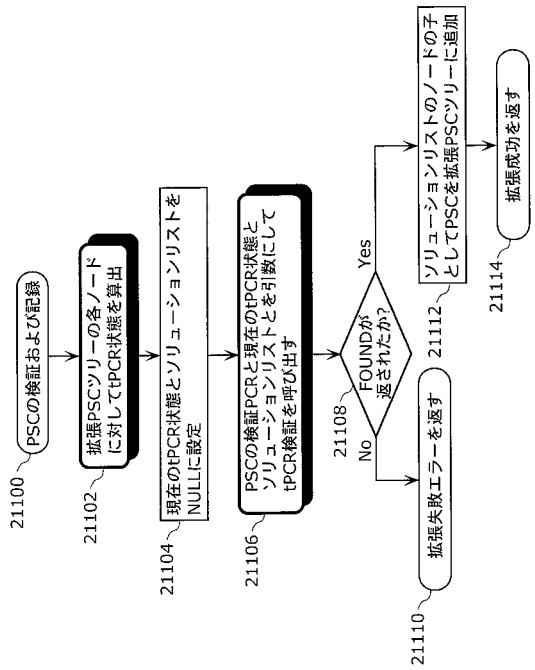

【図２９】



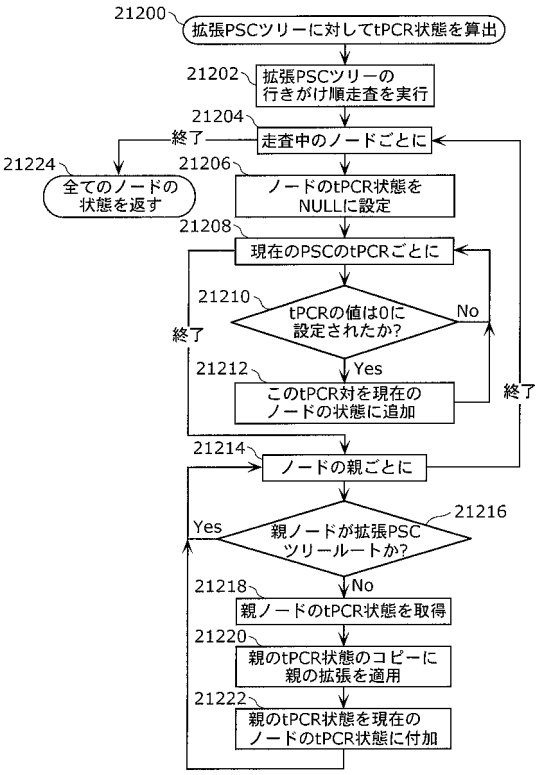

【図３０】



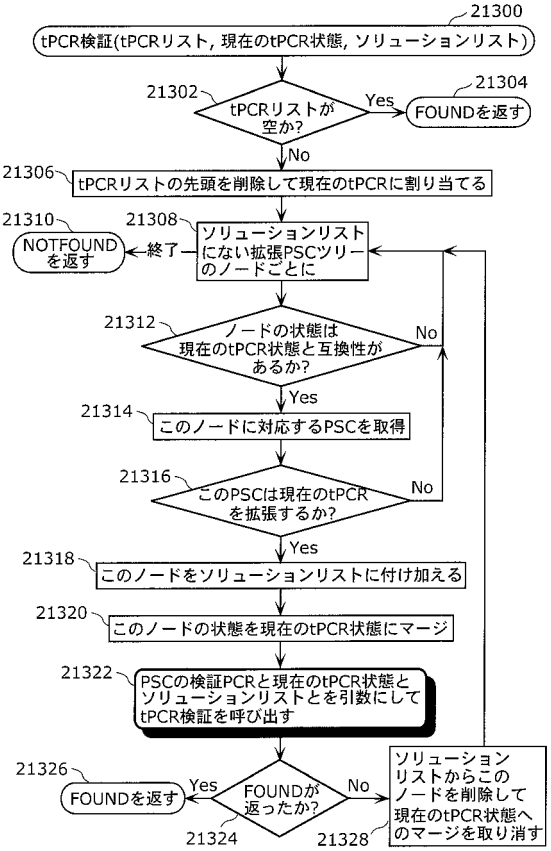
【図３１】



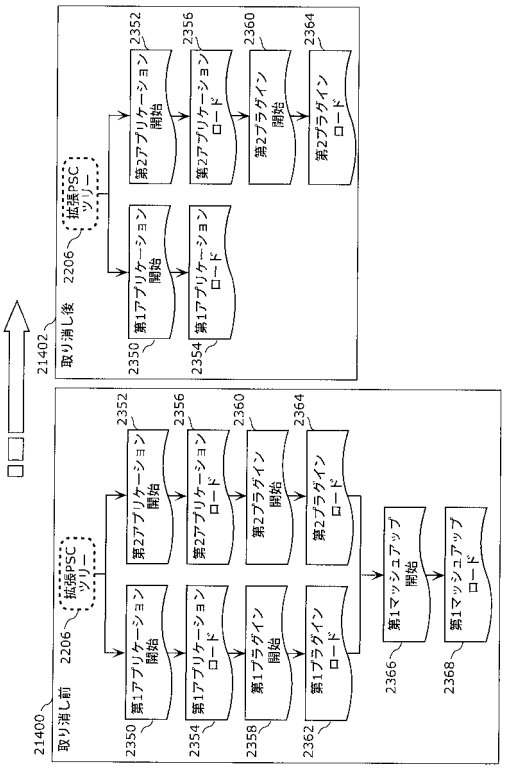

【図３２】



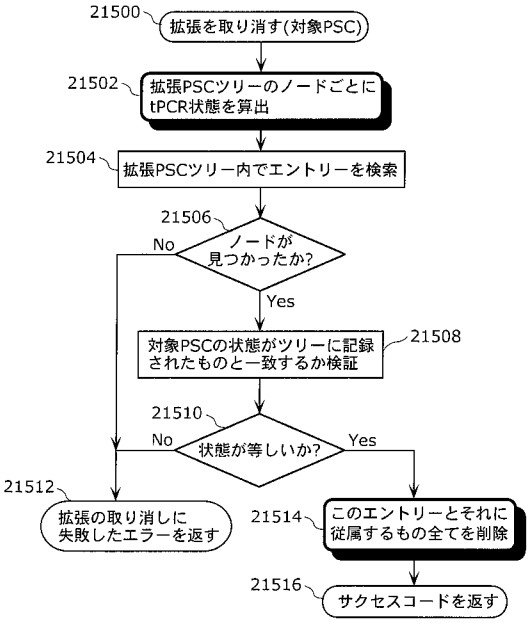

【図３３】



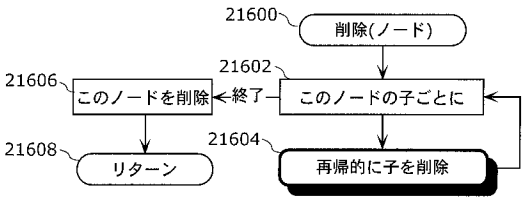

【図３４】



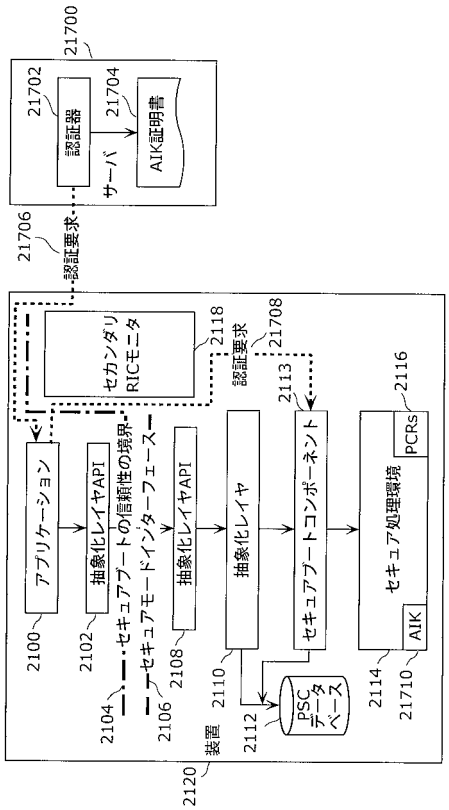
【図３５】



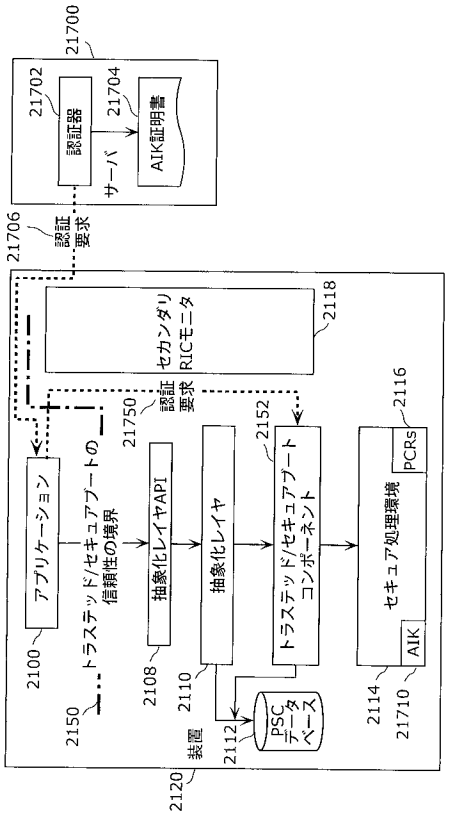

【図３６】



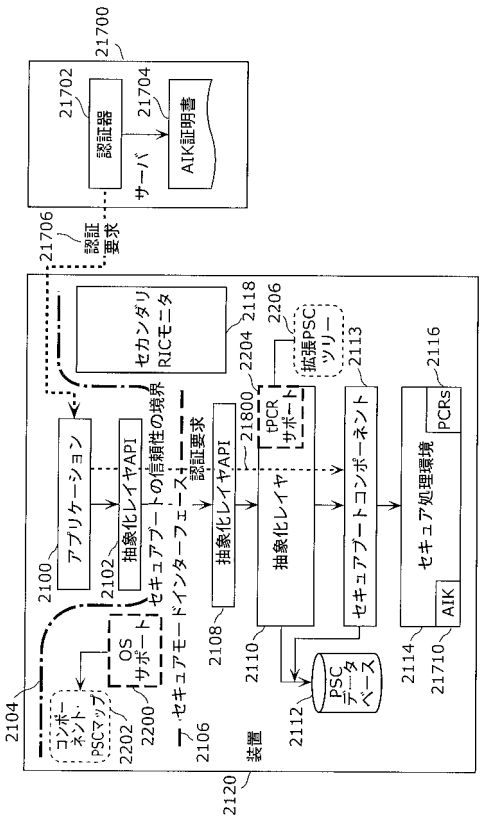

【図３７Ａ】



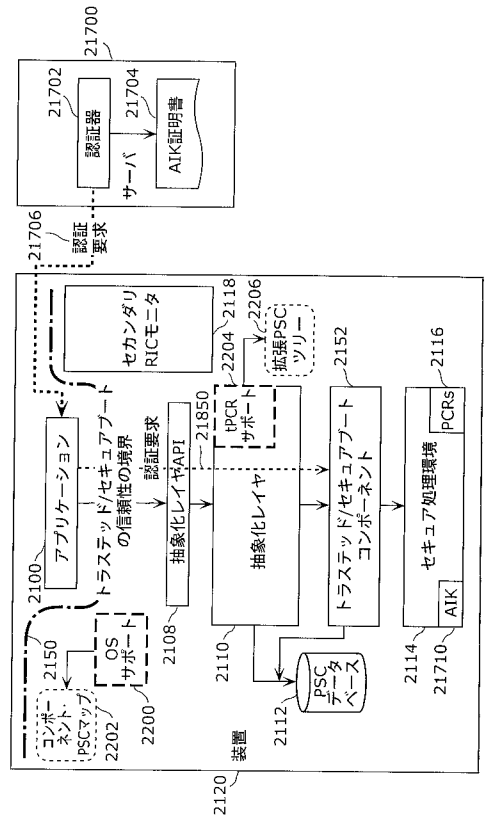

【図３７Ｂ】



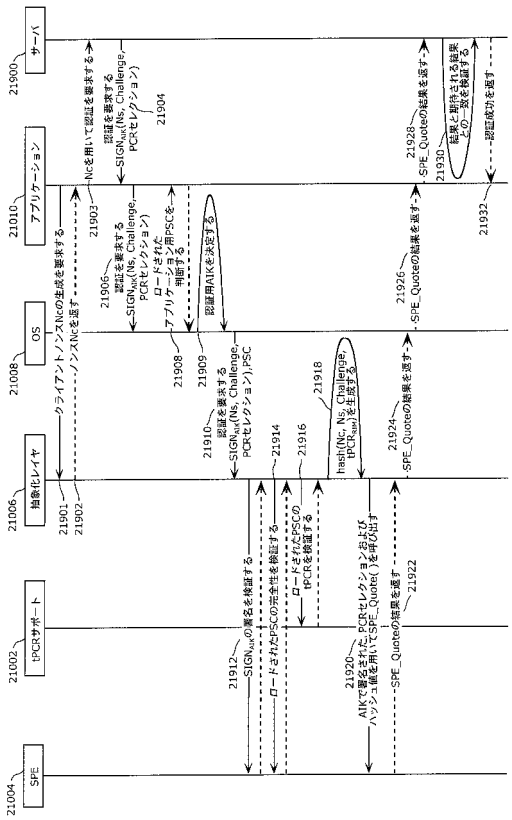

【図３８Ａ】



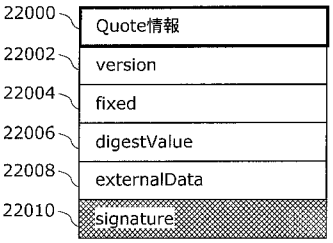
【図３８Ｂ】



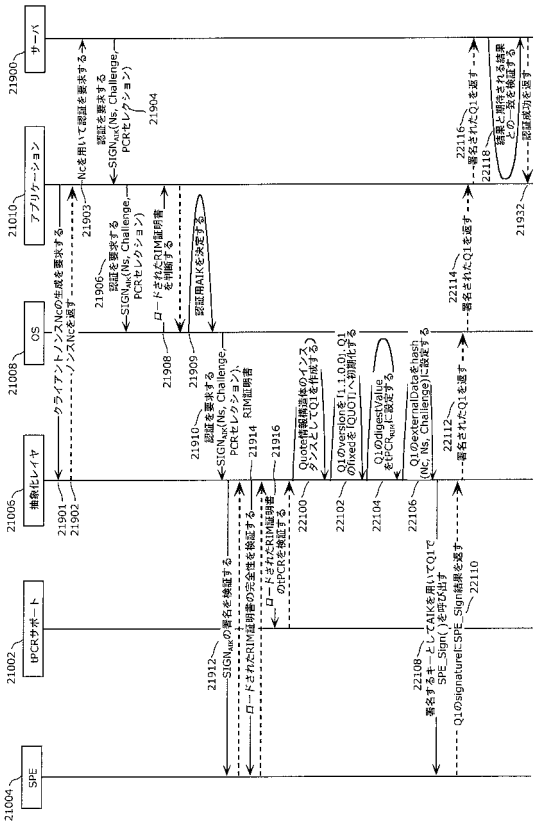

【図３９】




【図４０】




【図４１】



フロントページの続き

(72)発明者　伊藤　孝幸
日本国大阪府門真市大字門真１００６番地　パナソニック株式会社内
(72)発明者　芳賀　智之
日本国大阪府門真市大字門真１００６番地　パナソニック株式会社内

審査官　久慈　渉

(56)参考文献　特開２００６－３２３８１４（ＪＰ，Ａ）
特開２００７－０７２９０９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／１２
Ｇ０６Ｆ　９／４４５