(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6014286号
(P6014286)
(45)発行日　平成28年10月25日(2016.10.25)　(24)登録日　平成28年9月30日(2016.9.30)

(51)Int.Cl.　　F I
G06F 21/57 (2013.01)　　G06F 21/57 350
H04W 12/00 (2009.01)　　H04W 12/00

請求項の数 10　(全 76 頁)

(21)出願番号　特願2016-24084(P2016-24084)
(22)出願日　平成28年2月10日(2016.2.10)
(62)分割の表示　特願2014-208128(P2014-208128)の分割
原出願日　平成23年3月4日(2011.3.4)
(65)公開番号　特開2016-146631(P2016-146631A)
(43)公開日　平成28年8月12日(2016.8.12)
審査請求日　平成28年3月11日(2016.3.11)
(31)優先権主張番号　61/314,395
(32)優先日　平成22年3月16日(2010.3.16)
(33)優先権主張国　米国(US)
(31)優先権主張番号　61/311,106
(32)優先日　平成22年3月5日(2010.3.5)
(33)優先権主張国　米国(US)

(73)特許権者　510030995
インターデイジタル　パテント　ホールディングス　インコーポレイテッド
アメリカ合衆国　１９８０９　デラウェア州　ウィルミントン　ベルビュー　パークウェイ　２００　スイート　３００
(74)代理人　110001243
特許業務法人　谷・阿部特許事務所
(72)発明者　アンドレアス　シュミット
ドイツ　６５９２９　フランクフルト　アム　マイン　トイトーネンウェグ　３７
(72)発明者　アンドレアス　レシェル
ドイツ　６０３８５　フランクフルト　ハイデシュトラッセ　１３１

最終頁に続く

(54)【発明の名称】機器にセキュリティを提供する方法および装置

(57)【特許請求の範囲】
【請求項１】
１または複数のコンポーネントを備え、かつセキュアな環境を有するゲートウェイデバイスにおいて、クライアントデバイスは、前記ゲートウェイデバイスに接続し、かつ前記ゲートウェイデバイスを使用して前記ゲートウェイデバイスを通してサービスに接続し、前記ゲートウェイデバイスおよび前記接続されたクライアントデバイスの妥当性検査のために使用されることができる検証データを生成する方法であって、前記方法は、
前記ゲートウェイデバイスの複数のコンポーネントの各々について、前記ゲートウェイデバイスの前記コンポーネントの測定を表す値を取得することと、
コンポーネント測定値の記録を含む測定ログ（ＭＬ）を生成することと、
各コンポーネントについての前記コンポーネント測定値から検証データを生成し、前記ゲートウェイデバイスの前記セキュアな環境内の１または複数のセキュアなレジスタに前記検証データを格納することと、
前記検証データおよび前記ＭＬをツリー構造に編成することであって、前記検証データを含む前記セキュアなレジスタは、前記ツリー構造の根を定義し、前記ＭＬは、前記ツリー構造の内部ノードを定義し、前記ＭＬに含まれる前記測定値は、ツリー構造の葉を定義する、ことと、
コンポーネント測定値および検証データを前記接続されたクライアントデバイスから集め、前記接続されたクライアントデバイスからの前記コンポーネント測定値および検証データを前記ゲートウェイデバイスの前記ツリー構造に、前記ツリー構造のサブツリーとし

て統合することと
を備える方法。
【請求項２】
前記ツリー構造は、前記セキュアな環境のセキュアな拡張動作を使用して形成される、請求項１に記載の方法。
【請求項３】
信頼されるサードパーティにコンタクトし、前記サブツリーのために前記信頼されるサードパーティによって発行された証明書を前記信頼されるサードパーティから受信することと、
前記受信された証明書を前記ツリー構造に組み込むために前記ゲートウェイデバイスの前記ツリー構造を更新することと
をさらに備える、請求項１に記載の方法。
【請求項４】
前記信頼されるサードパーティは、前記ゲートウェイデバイスの前記セキュアな環境の中で稼動するアプリケーションを備える、請求項３に記載の方法。
【請求項５】
前記サービスが前記ゲートウェイデバイスおよび前記接続されたクライアントデバイスの信頼性を妥当性検証することができるように、前記接続されたクライアントデバイスが前記ゲートウェイデバイスを介してアクセスしようとしている前記サービスに前記更新されたツリー構造を送信することをさらに備える、請求項３に記載の方法。
【請求項６】
１または複数のコンポーネントを備え、かつセキュアな環境を有するゲートウェイデバイスであって、クライアントデバイスは、前記ゲートウェイデバイスに接続し、かつ前記ゲートウェイデバイスを使用して前記ゲートウェイデバイスを通してサービスに接続し、前記ゲートウェイデバイスは、
前記ゲートウェイデバイスの複数のコンポーネントの各々について、前記ゲートウェイデバイスの前記コンポーネントの測定を表す値を取得し、
コンポーネント測定値の記録を含む測定ログ（ＭＬ）を生成し、
各コンポーネントについての前記コンポーネント測定値から検証データを生成し、前記ゲートウェイデバイスの前記セキュアな環境内の１または複数のセキュアなレジスタに前記検証データを格納し、
前記検証データおよび前記ＭＬをツリー構造に編成し、前記検証データを含む前記セキュアなレジスタは、前記ツリー構造の根を定義し、前記ＭＬは、前記ツリー構造の内部ノードを定義し、前記ＭＬに含まれる前記測定値は、ツリー構造の葉を定義し、
コンポーネント測定値および検証データを前記接続されたクライアントデバイスから集め、前記接続されたクライアントデバイスからの前記コンポーネント測定値および検証データを前記ゲートウェイデバイスの前記ツリー構造に、前記ツリー構造のサブツリーとして統合する
ように構成される、ゲートウェイデバイス。
【請求項７】
前記ツリー構造は、前記セキュアな環境のセキュアな拡張動作を使用して形成される、請求項６に記載のゲートウェイデバイス。
【請求項８】
前記ゲートウェイデバイスは、
信頼されるサードパーティにコンタクトし、前記サブツリーのために前記信頼されるサードパーティによって発行された証明書を前記信頼されるサードパーティから受信し、
前記受信された証明書を前記ツリー構造に組み込むために前記ゲートウェイデバイスの前記ツリー構造を更新する
ようにさらに構成される、請求項６に記載のゲートウェイデバイス。
【請求項９】

前記信頼されるサードパーティは、前記ゲートウェイデバイスの前記セキュアな環境の中で稼動するアプリケーションを備える、請求項８に記載のゲートウェイデバイス。
【請求項１０】
前記ゲートウェイデバイスは、
前記サービスが前記ゲートウェイデバイスおよび前記接続されたクライアントデバイスの信頼性を妥当性検証することができるように、前記接続されたクライアントデバイスが前記ゲートウェイデバイスを介してアクセスしようとしている前記サービスに前記更新されたツリー構造を送信するようにさらに構成される、請求項８に記載のゲートウェイデバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
本出願は、機器へのセキュリティの提供に関する。
【背景技術】
【０００２】
マシン間通信（Ｍ２Ｍ）の登場により、ｅ－Ｈｅａｌｔｈ、地理的位置の追跡（Geo-Tracking）、ホームオートメーションおよび消費者向け機器への利用が可能になっている。そのような用途の多くでは、ネットワーク運営者の機器を顧客の構内に配置することが必要となる。そのような機器および装置は、悪意ある攻撃の対象となる。そのような悪意ある攻撃に対処するために、そのような顧客構内に置かれる機器には、ファイアウォールやウィルス監視等の他の形態の機器保護に加えて、機器の完全性の検証が必要となる。
【０００３】
機器の完全性の保護についてはいくつかの方法が論じられている。それらの方法の１つにセキュアブートがあり、これは、信頼される実行環境で、完全性が検証されたソフトウェアコンポーネントだけをロードおよび実行するものである。しかし、この方法は体系化されていない測定結果のセットを必要とし、そのような測定結果の数が非常に多い場合には管理が煩瑣になる可能性がある。従って、必要とされるのは、測定結果を収集、分類および体系化することを助けて、完全性を満たさないコンポーネントの効率的な探索の利益となる方法およびそれに関連する装置である。
【発明の概要】
【０００４】
無線送受信ユニット（ＷＴＲＵ）の妥当性検査に使用することができる検証データを生成する方法を含む、検証データを生成する各種技術が本明細書に開示される。ＷＴＲＵは、１つまたは複数のコンポーネントと、いくつかのセキュアなレジスタを備えたセキュアな環境とを有することができる。一実施形態によれば、ＷＴＲＵの複数のコンポーネント各々について、ＷＴＲＵのそのコンポーネントの測定結果を表す値を取得することができる。コンポーネント測定値の記録を含む測定ログ（ＭＬ）を生成することができ、他のコンポーネント固有のデータをＷＴＲＵに記憶することができる。コンポーネントごとにコンポーネント測定値から検証データを生成し、その検証データを、信頼されるプラットフォームモジュール内のセキュアレジスタの１つまたは複数に記憶することができる。検証データおよびＭＬは、ツリー構造に編成することができる。検証データを含むセキュアレジスタが、ツリー構造の根を定義することができる。ＭＬがツリー構造の内部ノードを定義し、ＭＬに含まれる測定値がツリー構造の葉を定義することができる。ツリー構造は、セキュアな環境のセキュアな拡張動作を使用して形成することができる。
【０００５】
別の実施形態によれば、ＷＴＲＵのコンポーネントの測定結果を表す値を取得することができる。測定値から検証データを生成し、その検証データを、ＷＴＲＵのセキュアな環境内のレジスタに記憶することができる。測定値は、ツリー構造の葉ノードに記憶することができる。セキュアな環境内で１回または複数回の拡張動作を行って、葉ノードに記憶された値をツリー構造の根ノードまで拡張することができる。根ノードは、生成された検

証データが記憶されるセキュアレジスタ内のデータを含むことができる。

【０００６】

別の実施形態によると、無線送受信ユニット（ＷＴＲＵ）によって生成されたツリー型検証データを妥当性検査する方法が記載される。ツリー型検証データは、ツリー構造に編成された、検証データ要素、測定ログ（ＭＬ）およびコンポーネント測定値を含むことができる。検証データ要素は、ツリー構造の根ノードを定義することができる。ＭＬはツリー構造の内部ノードを定義することができる。コンポーネント測定値は、ツリー構造の葉ノードを定義することができる。ツリー型検証データは、編成されたツリー構造で受け取ることができる。受け取ったツリー型検証データの根にある検証データ要素から開始して、ツリー構造をトラバースすることができる。ツリー構造のトラバースの一部として、受け取ったツリー構造の分岐ノードおよび分岐ノードの子ノードの値を、参照ツリーの同じノード位置にある値と比較することができる。そして、ノード値の比較に基づいて、ＷＴＲＵまたはＷＴＲＵの個々のコンポーネントの妥当性を認めるかどうかを決定することができる。

【０００７】

別の実施形態によれば、無線送受信ユニット（ＷＴＲＵ）によって生成された測定ログ（ＭＬ）のノード値を証明する方法が記載される。ＭＬの値は、根ノード、内部ノード、および葉ノードを備えるツリー構造のノードとして記憶することができる。部分木証明書機関（ＳＣＡ）で証明すべきノード値を知らせるアテステーション（attestation）パッケージを受け取ることができる。ノード値を、ＳＣＡで証明することができるノード値として認識することができる。ノード値に関連付けられた妥当性検査情報を含む、ノード値に関連付けられたマニフェストを作成することができる。妥当性検査情報をＷＴＲＵのセキュアな環境にバインドするように構成された、ノード値の証明書を作成することができる。証明書は、マニフェストと共に発行され、証明書を自身のＭＬに記憶するＷＴＲＵのセキュアな環境に提供されることができる。

【０００８】

本明細書に記載されるシステム、方法、および装置の他の特徴および態様は、以下の詳細な説明および関連する図面から明らかになろう。

【図面の簡単な説明】

【０００９】

上記概要並びに以下の詳細な説明は、添付図面と併せて読むと、より良く理解される。本明細書に記載されるシステム、方法および装置を説明する目的で、例示的実施形態を図面に示すが、本発明は、ここに開示される特定の方法および手段に限定されない。

【図１】例示的なＬＴＥ無線通信システム／アクセスネットワークの図である。

【図２】ＬＴＥ無線通信システムの例示的ブロック図である。

【図３】ツリー型の記憶測定ログ（ＳＭＬ）および検証データの概略的な構造を示す図である。

【図４】ツリーの形成を示すアルゴリズム（アルゴリズム１）の例の図である。

【図５】右エッジが正しい構成を示す図である。

【図６】不完全なツリーのクリーンアップを示すアルゴリズム（アルゴリズム２）の図である。

【図７】深さ３の不完全なツリーの順序形成／ツリー分岐を示す図である。

【図８】葉の測定値をｍと表したツリー検証データの最大容量構成を示す図である。

【図９】ツリー型のＳＭＬにおけるノード構成の分類を示す図である。

【図１０】ｄ＝１６の場合に２ｄｆ個のｂａｄな葉のランダム分布にあるｂａｄな内部ノードの予想含有比の図である。

【図１１】ＳＭＬツリー内の全ての基本三角形における正しい値の設定を示す図である。

【図１２】線形のハッシュチェーンで最初の破綻箇所を見つけるアルゴリズム１を示す図である。

【図１３】ハフマン符号化ツリーの例を示す図である。

【図１４】ツリーの間引きの例を示す図である。
【図１５】最適なツリー型の検証システム図および関連する通信を説明する図である。
【図１６】モジュールによって指示されたソフトウェアコンポーネントまたは機能が別のモジュールを使用する代理子リンクを伴うツリーを説明する図である。
【図１７】二分木への判断基準の投入を決定するアルゴリズム２を示す図である。
【図１８】ＴＰＭコマンドを使用したツリーへの投入を決定するアルゴリズム３を示す図である。
【図１９】ｎ分木への判断基準の投入を決定するアルゴリズム４を示す図である。
【図２０】二分木への代理子リンクの投入を決定するアルゴリズム５を示す図である。
【図２１】完全性検査に合格しないノードおよび葉を判定する、比較および間引きのアルゴリズム６を示す図である。
【図２２】TPM_Reduced_Tree_Verify_Loadを説明するアルゴリズム１の図である。
【図２３】TPM_reduced_Tree_Verifyを説明するアルゴリズム２の図である。
【図２４ａ】TPM_Tree_Node_Verifyを説明するアルゴリズム３の図である。
【図２４ｂ】TPM_Reduced_Tree_Updateを説明するアルゴリズム４の図である。
【図２５】ＰＶＭのデータカテゴリを示す図である。
【図２６】根を有する部分木の部分木証明プロトコルを示す図である。
【図２７】証明書と部分木のバインドを示す図である。
【図２８】左が非平衡の多ツリー構造の図である。
【図２９】本明細書に記載されるツリー構造の例示的実施形態の図である。
【図３０】コンポーネントの下位ツリー構造の図である。
【図３１】分割妥当性検査のステップ１、測定結果の収集を示す図である。
【図３２】分割妥当性検査のステップ２、部分木の証明を示す図である。
【図３３】分割妥当性検査のステップ３、サービスの接続を示す図である。
【図３４】分割妥当性検査にＨ（ｅ）ＮＢを使用する事例を示す図である。
【図３５】Ｈ（ｅ）ＮＢがローグ（rogue）機器へのアクセスを阻止する図である。
【図３６】Ｍ２Ｍ　ＧＷが、種類、機器クラス、機器プロパティ、または接続プロファイル（プロトコル、接続ネットワーク、帯域幅）に基づいて機器をグループに分類し、機器妥当性検査ツリーについてのグループ証明書を提供する図である。
【図３７】Ｍ２Ｍ　ＧＷによるＰ２Ｐの分割妥当性検査を示す図である。
【図３８】１つまたは複数の開示実施形態を実施することができる例示的な通信システムのシステム図である。
【図３９】図３８に示す通信システムで使用することが可能な例示的な無線送受信ユニット（ＷＴＲＵ）のシステム図である。
【図４０】図３８に示す通信システムで使用することが可能な例示的無線アクセスネットワークおよび例示的コアネットワークのシステム図である。
【発明を実施するための形態】
【００１０】
本明細書で言及する場合、用語「無線送受信ユニット（ＷＴＲＵ）」は、これらに限定されないが、ユーザ機器（ＵＥ）、移動局、固定または移動型の加入者ユニット、ページャ、携帯電話、携帯情報端末（ＰＤＡ）、コンピュータ、または無線環境で動作可能な他の種類のユーザ装置を含む。また、用語「基地局」はこれらに限定されないが、ノードＢ、サイトコントローラ、アクセスポイント（ＡＰ）、または無線環境で動作可能な他の種類のインタフェース装置を含む。
【００１１】
検証データを生成する各種技術が本明細書に開示され、それらには、ＷＴＲＵの妥当性検査に使用することができる検証データを生成する方法が含まれる。ＷＴＲＵは、１つまたは複数のコンポーネントと、いくつかのセキュアなレジスタを備えたセキュアな環境とを有することができる。セキュアな環境は、セキュアな実行環境を提供するセキュアなハードウェアおよび／またはソフトウェア環境を含むことができる。例えば、セキュアな環

境は、信頼されるプラットフォームモジュール（ＴＰＭ）、スマートカード、ユニバーサル集積回路カード（ＵＩＣＣ）またはそれらの組合せ等である。このセキュアな環境を使用して、例えば暗号機能等のセキュリティ機能、例えば演算レジスタ、メモリ、乱数生成器、タイマ、および／またはクロック等のセキュアなリソースを保護することができる。

【００１２】

一実施形態によると、検証データは、ＷＴＲＵの複数のコンポーネント各々について、ＷＴＲＵのそのコンポーネントの測定結果を表す値を取得することによって生成することができる。コンポーネントの測定値の記録を含む測定をＷＴＲＵに記憶することができる。各コンポーネントのコンポーネント測定値から検証データを生成し、その検証データを信頼されるプラットフォームモジュール内部のセキュアなレジスタの１つまたは複数に記憶することができる。検証データおよびＭＬは、ツリー構造に編成することができる。検証データを保持するセキュアレジスタは、ツリー構造の根を定義することができる。ＭＬは、ツリー構造の内部ノードを定義し、ＭＬに保持される測定値は、ツリー構造の葉を定義することができる。ツリー構造は、セキュアな環境のセキュアな拡張動作を使用して形成することができる。

【００１３】

別の実施形態によると、ＷＴＲＵのコンポーネントの測定結果を表す値を得ることができる。その測定値から検証データを生成し、検証データを、ＷＴＲＵのセキュアな環境内部のレジスタに記憶することができる。測定値は、ツリー構造の葉ノードに記憶することができる。セキュアな環境内で１回または複数回の拡張動作を行って、葉ノードに記憶された値を当該ツリー構造の根ノードまで拡張することができる。根ノードは、生成された検証データが記憶されているセキュアレジスタ内のデータを含むことができる。

【００１４】

別の実施形態によれば、ＷＴＲＵによって生成されたツリー型の検証データの妥当性を検査する方法が記載される。ツリー型の検証データは、ツリー構造に編成された、検証データ要素、測定ログ（ＭＬ）およびコンポーネントの測定値を含むことができる。検証データ要素は、ツリー構造の根ノードを定義することができる。ＭＬは、ツリー構造の内部ノードを定義することができる。コンポーネントの測定値は、ツリー構造の葉ノードを定義することができる。ツリー型の検証データは、編成されたツリー構造で受け取ることができる。受け取ったツリー型の検証データの根にある検証データ要素から、ツリー構造をトラバースすることができる。ツリー構造のトラバースの一部として、受け取ったツリー構造の分岐ノードおよび分岐ノードの子ノードにある値を、参照ツリーの同じノード位置にある値と比較することができる。そして、ノード値の比較に基づいて、ＷＴＲＵまたはＷＴＲＵの個々のコンポーネントの妥当性を認めるどうかを判定することができる。

【００１５】

別の実施形態によれば、ＷＴＲＵによって生成された測定ログ（ＭＬ）のノード値を証明する方法が記載される。ＭＬの値は、根ノード、内部ノードおよび葉ノードからなるツリー構造のノードに記憶することができる。部分木認証機関（ＳＣＡ）が、証明すべきノード値を示すアテステーションパッケージを受け取ることができる。そのノード値は、ＳＣＡで証明することができるノード値として認識されることができる。ノード値に関連付けられた妥当性検査情報を含む、ノード値に関連付けられたマニフェストを作成することができる。妥当性検査情報をＷＴＲＵのセキュアな環境にバインドするように構成された、ノード値の証明書を作成することができる。証明書は、マニフェストと共に発行されてＷＴＲＵのセキュアな環境に提供されることができ、ＷＴＲＵは証明書を自身のＭＬに記憶する。

【００１６】

構造化された妥当性検査は、データおよび妥当性検査動作の諸面が構造化される妥当性検査の方法論である。構造化妥当性検査の別個であるが互いに関連した概念および方法が本明細書に記載される。例えば、ツリー型妥当性検査（ＴＦＶ：Tree-Formed Validation）が本明細書に記載され、部分木の証明、拡張およびＴＦＶの変形、並びに分割妥当性検

査を使用する方法に焦点を当てる。分割妥当性検査では、妥当性検査作業を２つ以上のネットワークエンティティに分散し、それにより、ネットワークエンティティは、（接続された）機器に対して分散方式で機器の完全性検査を行うことができ、検査を行う各エンティティは必ずしも機器全体を妥当性検査する必要はなく、その一部分を妥当性検査することができる。

【００１７】

図１は、Ｅ－ＵＴＲＡＮ（Evolved-Universal Terrestrial Radio Access Network）６０５を含むＬＴＥ（Long Term Evolution）の無線通信システム／アクセスネットワーク４００を示す図である。Ｅ－ＵＴＲＡＮ６０５は、ＷＴＲＵ６１０および数個のｅｖｏｌｖｅｄノードＢ（ｅＮＢ）６２０を含む。ＷＴＲＵ６１０は、ｅＮＢ６２０と通信状態にある。ｅＮＢ６２０は、Ｘ２インタフェースを使用して相互とインタフェースをとる。各ＮＢ６２０は、Ｓ１インタフェースを通じて移動性管理エンティティ（ＭＭＥ）／サービングゲートウェイ（Ｓ－ＧＷ）６３０とインタフェースをとる。図１には１つのＷＴＲＵ６１０および３つのｅＮＢ６２０を示すが、無線通信システムアクセスネットワーク６００には無線および有線の機器の任意の組合せが含まれてよいことが明らかである。

【００１８】

図２は、ＷＴＲＵ６１０、ｅＮＢ６２０、およびＭＭＥ／Ｓ－ＧＷ６３０を含むＬＴＥ無線通信システム５００の例示的ブロック図である。図２に示すように、ＷＴＲＵ６１０、ｅＮＢ６２０、およびＭＭＥ／Ｓ－ＧＷ６３０は、機器にセキュリティを提供する方法を行うように構成される。

【００１９】

典型的なＷＴＲＵに見られる構成要素に加えて、ＷＴＲＵ６１０は、オプションの結合されたメモリ７２２を備えるプロセッサ７１６、少なくとも１つのトランシーバ７１４、オプションのバッテリ７２０、およびアンテナ７１８を備える。プロセッサ７１６は、機器にセキュリティを提供する方法を行うように構成される。

【００２０】

トランシーバ７１４は、プロセッサ７１６およびアンテナ７１８と通信状態にあって、無線通信の送受信を容易にする。ＷＴＲＵ６１０でバッテリ７２０が使用される場合は、バッテリ７２０がトランシーバ７１４およびプロセッサ７１６に電力を供給する。

【００２１】

典型的なｅＮＢに見られる構成要素に加えて、ｅＮＢ６２０は、オプションの結合されたメモリ７１５を備えるプロセッサ７１７、トランシーバ７１９、およびアンテナ７２１を備える。プロセッサ７１７は、機器にセキュリティを提供する方法を行うように構成される。トランシーバ７１９は、プロセッサ７１７およびアンテナ７２１と通信状態にあって、無線通信の送受信を容易にする。ｅＮＢ６２０は、オプションの結合されたメモリ７３４を有するプロセッサ７３３を備えた移動性管理エンティティ／サービングゲートウェイ（ＭＭＥ／Ｓ－ＧＷ）６３０に接続される。

【００２２】

一般に、機器の完全性を保証するには、ソフトウェア／ファームウェアまたはハードウェアの測定（例えば暗号ハッシュ値等）を行い、その結果を信頼される参照値と比較する。この信頼される参照値との測定結果（検証データと呼ばれる）の比較、またはそのような測定結果の関数若しくはグループ化は、機器内部で行われる場合も（自律妥当性検査）、外部で妥当性検査エンティティによって行われる場合（準自律または遠隔妥当性検査）もある。準自律または遠隔妥当性検査の場合は、測定結果を体系化されていないセットとしてペイロードで送信することができ、その際、暗号化、完全性保護、および暗号による証明を行うことができる。

【００２３】

完全性の検証に合格しなかったコンポーネントを見つけるために、測定結果のセットと参照値のセットとの比較を行って、完全性の測定に合格しなかったインデックスのセットを得ることができる。しかし、そのような体系化されていない測定結果のセットは、測定

結果の数が非常に多い場合には管理が煩瑣になる可能性がある。

【００２４】

完全性の検査に合格しなかったモジュールの探索を最適化するために、検証データは、測定ログ、例えば記憶された測定ログ（ＳＭＬ）のハッシュツリーの形態で生成することができる。ＴＣＧ（Trusted Computing Group）アーキテクチャおよび仕様で使用される用語「ＳＭＬ」を、本明細書に記載される測定ログの各種実施形態の説明で使用することができるが、ＳＭＬは測定ログの例示的実施形態の１つである。体系化の一例はツリー型妥当性検査（ＴＦＶ）である。記憶された測定ログは平衡二分木に編成することができ、ＴＰＭコマンドを使用してツリーを生成するためのアルゴリズムを提供することができる。アルゴリズムは汎用的なものでよく、平衡二分木に合わせて拡張することができる。ツリー型検証で生成されるＳＭＬの内容はＴＣＧ　ＳＭＬに似るが、ＴＣＧ　ＳＭＬの処理および／または操作とは異なる形で、構築、形成、計算、使用、記憶、探索、更新、削除、通信による送信および／または受信、あるいはその他の方法で処理または操作することができる。

【００２５】

ＴＦＶは、例えば、後述するようなＴｒＥで実装することができる。ＴＦＶは、多くの他の使用事例の類の革新をもたらし、ＴｒＥが目的とする、複雑な種類のネットワーク側のプラットフォームの妥当性検査を可能にする基準となる。ＴＦＶの技術的主旨は本明細書に記載される。ＴＦＶポインティングの概要が本明細書に記載される。また、ＴＦＶをＴｒＥのコンテクストに置き、その応用例を本明細書に記載する。

【００２６】

ＴＦＶの要素の１つは検証データの階層構造であり、これは、遠隔のバリデータ（validator）によるプラットフォーム妥当性検査に種々の利点をもたらす。ツリー型の検証データを作成および管理するためのアルゴリズムが本明細書に記載される。さらに、ＴＦＶにおいて、プラットフォームの下部構造、すなわち検証データの部分木のアテステーションを使用してプラットフォームの妥当性を検査することができる。階層中の検証データの最上位レベルは根（の集合）と呼ばれ、特別の保護、例えばハードウェア保護がなされた検証データレジスタ等によりＴＦＶの際に保護することができる。ＴＦＶは、小規模の効率的なアルゴリズムでツリー型検証データに作用することができる。ＴＦＶの実装全体は、ＴＣＢ内部で実施するか、さらにはハードウェアで保護されたセキュアな実行環境で実施することができる。妥当性検査データの複雑な構造は、ツリー型検証データのツリー階層に関連付ける、かつ／またはツリー階層によって保護することができる。部分木は、構造と下部構造との関係により、検証データおよび妥当性検査データに固有のセマンティックを与えることができる。従って、部分木は、バリデータに対してプラットフォームの機能部分を識別することができる。

【００２７】

ＴＦＶの一実施形態では、変更を加えたＴＰＭを使用することができる。既存技術であるトラステッドコンピューティングは、ＴＦＶの実現および／または標準化への潜在的な入口点となる。ＴＦＶは、二分木等のツリーから、下部構造を有するより一般的な構造へと容易に一般化できる。

【００２８】

以下に記載するのはＴＦＶが提供することができる要素である。

【００２９】

セキュリティ：ＴＦＶでは、ツリー型妥当性検査データの根を保護することによりハードウェア保護が提供される。基準となる保護レベルは、ＴＰＭのＰＣＲの保護レベルであり、これをＴＦＶで維持できることが示される。プラットフォーム側のアルゴリズムを小規模にし、複雑度を低くすることで、小規模のＴＣＢまたはオンチップでの実装が可能となる。

【００３０】

管理：ＴＦＶは、プラットフォームの下部構造を確実かつセキュアに取り出し、当該下

部構造の集まりに基づいてプラットフォームを管理する方法および装置を含む。ＴＦＶの部分木で表されるモジュールは、柔軟に変更、更新、さらにはプラットフォーム間で移動することができ、任意のシナリオで必要とされるセキュリティプロパティを備える。

【００３１】

分散：ＴＦＶの階層は、妥当性検査をエンティティ間に階層的に分割することを可能にする。それにより、通信のシナリオと使用事例をより柔軟に設計することが可能になり、また効率にも利する。

【００３２】

効率：探索に効率的なデータ構造の１つである二分木をＴＦＶで実装することができる。例えば、ＴＦＶでは、不合格コンポーネント（完全性の測定値が望ましくない）の探索が、ＴＣＧのようなプラットフォームアテステーションよりも効率的になることが示される。ＴＦＶに導入される自然な階層は、妥当性検査プロセスにおける負荷分散の選択肢を与える。

【００３３】

ＴＦＶの例示的特徴の１つは、ＴＦＶは、予め組み込まれた階層的な順序により、多数の低容量プラットフォームを「より中央の」エンティティで妥当性検査するために設計できる点である。従って、ＴＦＶは、ゲートウェイに接続する機器および／またはゲートウェイを介してネットワークに接続する機器に適する可能性がある。例示的な使用事例の１つとして、Ｍ２M通信のシナリオが挙げられる。ＴＦＶのこの特質により、ＴＦＶは、多くの既存の概念に直交する概念となり、信頼されるプラットフォームによる状態のアテステーションに、より多くのセマンティックを与えることができる。プラットフォームによるアテステーションの他の手法は全く逆の理念を示す。それらの手法は、複雑な妥当性検査データを生成できるプラットフォームには適するが、データの階層的、モジュール的な構造についてはあまり想定していない。ＴＦＶは、ＰＢＡ、セマンティックアテステーション、仮想化および／またはＨＩＭ等の他の直交する手法と組み合わせることもできる。

【００３４】

説明で使用する用語と頭字語の一覧を以下に示す。

【００３５】

ＲＩＭ：参照となる完全性の判断基準（Reference Integrity Metric(s)）は、実際の測定データを比較することができる参照値を提供する。ＲＩＭは、妥当性検査のために、機器から提供される測定値に対応するものである。ＲＩＭは、所望の目標値を報告された測定値と比較する際の基準の役割を果たす。ＲＩＭは、例えばセキュアな検査機構で得られるコンポーネントコードの暗号ダイジェスト値として、一意にコンポーネントに関連付けられる点で完全性の証明を提供する。ＲＩＭは、測定値との直接の（決定論的な）比較を可能にする点で判断基準となる。

【００３６】

ＲＩＭ証明書：ＲＩＭ証明書は、ＴＴＰに署名された特定コンポーネントのＲＩＭを含んでいる。

【００３７】

ＡｕＶ：自律的妥当性検査

【００３８】

ＩＤＳ：侵入検出システム

【００３９】

ＤｏＳ：（Denial-of-Service）（攻撃）

【００４０】

ＰＶＭ：プラットフォーム妥当性検査および管理。ＰＶＥによるプラットフォームの妥当性検査と、（ｅ）ＨＭＳによるプラットフォームのＯＴＡ管理の組合せ。この組合せから導出することができる全ての可能な機能の組合せを備える。

【００４１】

ＤＭＳ：デバイス管理システム。ホーム（ｅ）ＮｏｄｅＢ管理システム（ＨＭＳ）（Ｔ

Ｓ３２．５８３，［４］）の一般化概念（３ＧＰＰ　ＬＴＥ）。一般機器に適用され、ＰＶＭ機能で強化される。

【００４２】

ＲＩＭマネジャ／マネジャＲＩＭｍａｎ：妥当性検査データベースＶ＿ＤＢを管理するエンティティ。管理を行う権限のある唯一のエンティティであり、暗号を用いてデジタル証明書に作用（検証、生成）する唯一のエンティティ。

【００４３】

ＳＡＶ：準自律的妥当性検査）。ＰＶＭが基礎とする妥当性検査の変形。

【００４４】

ＴＣＧ：トラステッドコンピューティンググループ

【００４５】

ＴＳＳ：信頼されるソフトウェアスタック

【００４６】

ＴＰＭ：信頼されるプラットフォームモジュール

【００４７】

ＴＴＰ：信頼されるサードパーティ

【００４８】

ＴＣＢ：トラステッドコンピューティングベース。実行時に信頼性を評定することができず、無条件で信頼しなければならないシステム部分。

【００４９】

ＣＮ：オペレータコアネットワーク

【００５０】

ＳＨＯ：選択されたホームオペレータ

【００５１】

ＰＫＩ：公開鍵インフラストラクチャ

【００５２】

ＰＤＰ：ポリシー決定点

【００５３】

ＰＥＰ：ポリシー実施点

【００５４】

Ｄ－Ｈ：ディフィー・ヘルマン（Diffie-Hellman）

【００５５】

ＴＦＶ：ツリー型妥当性検査

【００５６】

検証データ（verification data）：セキュアな起動プロセス時の内部検証の総合的な結果を一意に識別するデータ。主な例は、ＰＣＲ値、すなわち多数の測定値のハッシュ値がチェーン化された形態のダイジェストである。

【００５７】

妥当性検査データ（validation data）：検証データと区別して、妥当性検査データは、別のパーティ、すなわちバリデータに提出され、プラットフォーム状態の信頼性を評価するために使用される全てのデータである。

妥当性検査：例えばＴＣＧに従って遠隔アテステーションとして実現されたバリデータへの妥当性検査データの提出と、バリデータによるデータの評価を厳密に妥当性検査と呼ぶ。妥当性検査データはしばしば、クオート（quote）された検証データレジスタ（例えばＰＣＲ）の値等の検証データからなる。妥当性検査は、暗号による検証データの検証以外に、ポリシーの評価およびバリデータによるアクションのトリガを含むことができる。

【００５８】

検証データレジスタ：不正アクセスおよび変更から検証データを保護するハードウェア記憶域。

【００５９】

信頼されるプラットフォームとの信頼関係の確立は、妥当性検査のプロセスに依拠する。妥当性検査により、外部エンティティは、提供されたプラットフォーム構成の証拠に基づいて、予想されるプラットフォームの振る舞いへの信頼を構築することができる。遠隔アテステーション等の妥当性検査機構では、信頼されるプラットフォームが、起動プロセス中に作成された検証データを公開する。それらのデータは、ロードまたは起動された全てのコンポーネントの例えばハッシュ値等の入れ子構造の測定値を保持しているプラットフォーム構成レジスタのハードウェア保護された値とすることができる。値は、セキュリティが保護された拡張動作で、線形の順序で作成することができる。検証データとそれに関連する測定ログの線形順序に基づくバリデータによるコンポーネントの細粒度の診断は、非効率的である場合がある。ツリー型検証データを作成する方法が提供され、この方法では、コンポーネントの測定値が葉に相当し、保護されたレジスタが根に相当する。この方法の仕組みを、限定された数のハードウェア保護されたレジスタと標準的な拡張動作を使用して説明する。このようにして、検証データのセキュリティを維持することができ、記憶測定ログは常にツリーとして編成することができる。ツリー型測定ログおよび検証データを使用してプラットフォームの妥当性を検査する基本の機構について説明する。

【００６０】

コンピューティングプラットフォームへの信頼を構築するプロセスは、独自の共通パターンに従うことができる。プラットフォームの起動時に、プラットフォーム上の保護されたエンティティによりコンポーネントを測定することができる。例えば、ロードおよび／または実行の前にコンポーネントを測定することができる。信頼のチェーンの生成は、トラステッドコンピューティングシステムの重要な概念である。このチェーンは、システムの立ち上げから、実行された命令およびプログラムを含む現在のシステム状態まで、切れ目なく続くことができる。各コンポーネントは、次のコンポーネントを実行する前に次のコンポーネントを測定し、報告することが必要となる場合がある。直後のコンポーネントを測定することにより、測定と実際の実行との間にコードが監視されずに実行されることを防ぐことができる。測定プロセスは、測定のためのＲｏＴ（root of trust）で保護することができ、例えばコードおよび構成データのダイジェスト値を計算することによって実施することができる。

【００６１】

検証データは、保護された動作で測定値から集計するか、かつ／または保護された記憶域に記憶することができる。検証データは、セキュアな起動の完了後等に、プラットフォームの状態を一意に識別することができる。これらのプロセスの実施形態は、ＴＣＧ（トラステッドコンピューティンググループ）により認証し、セキュアブートを指定することができる。ＰＣクライアントには認証ブートを使用することができる。モバイルプラットフォームのセキュアブートも使用することができる。この両者の違いは、セキュアブートはローカルの検証および施行エンジンを追加的に含むことであり、このエンジンは、コンポーネントの測定結果が信頼される参照値と等しい場合にコンポーネントを起動させる。

【００６２】

ＴＣＧは、コンポーネントコードおよび／またはデータのハッシュ値である測定値から、それぞれ高信頼プラットフォームモジュール（ＴＰＭ）、モバイル高信頼モジュール（ＭＴＭ）の拡張動作を介して検証データを計算することを提案する。データは、プラットフォーム構成レジスタ（ＰＣＲ）に記憶することができる。例えば、バージョン１．１の仕様によれば最少で１６個のＰＣＲが存在することができ、またバージョン１．２では少なくとも２４個のＰＣＲをＴＰＭが識別することができる。ＰＣＲは、許可されたコマンドからアクセスすることができる。拡張動作では、下記のように線形順序の入れ子構造のハッシュ値のチェーンを構築し、これはMerkle-Damgard変換に似る。

【００６３】

【数１】

$$V_i \leftarrow V_i \diamond m \quad \underline{\mathrm{def}} \quad H(V_i \| \|) \quad \text{（式１）}$$

【００６４】
ここで、V_iは検証データレジスタを表し（ＰＣＲの場合ｉ＝０；：：：；２３）、Ｈは耐衝突性のハッシュ関数（ＴＰＭの場合はＳＨＡ－１）であり、ｍ＝Ｈ（データ）は測定値である。このようにして、ＴＰＭの保護された機能と隠蔽された機能により、ＴＣＧの高信頼プラットフォームの検証データを操作から守ることができる。
【００６５】
検証データには、より表現力の高い測定値の記録および／または記憶された測定ログ（ＳＭＬ）にあるコンポーネント固有のデータが伴ってよい。外部エンティティに向けた妥当性検査時に、検証データおよび／またはＳＭＬ等の他のデータにプラットフォームが署名し、バリデータに転送することができる。バリデータは、所望の粒度でプラットフォームの信頼性を評定できる場合があり、粒度は、妥当性検査時に伝達される全情報によって制限される可能性がある。妥当性検査の基準となる実施形態をＴＣＧによりアテステーションプロトコルに定義することができる。ＴＣＧのTrusted Network Connectワーキンググループが構想しているように、例えば最初のネットワークまたはサービスへのアクセス時に、最終的に妥当性検査を使用して、信頼されるプラットフォームに修復ステップを行うことがＴＣＧによって構想されている。
【００６６】
ＴＣＧ仕様で予見される線形順序と異なり、ツリー（例えばMerkleハッシュツリー）として検証データおよびＳＭＬを編成する方法が提供される。線形にチェーン化された検証データの効率面の問題を応用の観点から浮き彫りにする。検証データをツリーとして編成する場合の主要なセキュリティ上の問題は、ＴＣＧ仕様の測定拡張動作が提供されていることから、データの生成をセキュアにすることである。ハードウェア保護されたレジスタの限定されたセットの中に検証データを生成する方法およびアルゴリズムも提供され、レジスタは、ハッシュツリーの根ノードを正確に表す。ツリー様の検証データおよびＳＭＬをどのようにして効率的かつ効果的に妥当性検査に使用できるかも示される。ツリー型検証データの実装のオプションおよび実施された実験についても述べる。
【００６７】
検証データは、無条件のセキュリティでシステム状態についての情報を提供する。例えば、その情報はＳＭＬとは別個にセキュアにすることができ、ＴＣＧ標準によれば、ＳＭＬは、プラットフォームまたは妥当性検査を特に保護しない場合がある（署名されたアテステーションデータの一部でない場合がある）。署名されたＰＣＲ値、すなわち検証データ自体は、暗黙的な完全性の管理をＳＭＬに提供することができる。そのために、全ての拡張動作をさかのぼることにより、ＳＭＬの測定結果から検証データを再計算することができる。
【００６８】
認証ブートでＰＣＲ値を使用して測定ログをセキュアにするためのＴＣＧ標準方式は、信頼されないマシン上で監査ログをセキュアにするためにSchneierおよびKelseyによって導入された技術に基づくことができる。実際にはこれは簡略化である可能性がある。何故ならば、ハッシュチェーンの最後の要素はＰＣＲで保持されるのに対して、ＳＭＬは測定値を含み、ハッシュチェーンの中間項目は含まないためである。ＴＰＭを使用した完全性の測定は、完全性測定アーキテクチャ（ＩＭＡ）で、ＴＰＭを使用して完全性を測定し、線形のＳＭＬを生成するＬｉｎｕｘ（登録商標）カーネルモジュールとして実装することができる。
【００６９】
拡張動作を線形にチェーン化することで作成された検証データは、プラットフォームの遠隔診断および／またはコンポーネント単位の修復等の高度の管理には価値が限定される可能性がある。基本的に、ＳＭＬが操作された場所は、ＰＣＲに拡張される前の測定値の改ざんの場合でも、またはセキュアな起動後のＳＭＬ自体の改ざんの場合でも、確実に箇所を特定できない可能性がある。さらに、何百または何千の測定コンポーネントを含む実世界のＳＭＬの空間的複雑性により、妥当性検査に合格しないコンポーネント、すなわち

、測定値が「正しい」参照値と異なるコンポーネントをＳＭＬからふるい分けることが高費用になる場合がある。コードおよび／またはデータを検査するために、各種の暗号チェックサム機能を利用することができ、それらの機能は、「正しい」データのチェックサムの完全性が維持されることを必要とする場合がある。各種マシンにある有効なバージョンのソフトウェアの集中データベースを必要とすることは、重要な管理上の問題であり、効率的な解決を必要としている。例えばマシン間通信で必要とされるような将来のネットワーク化された機器の大規模な配備には、信頼される機器と、ネットワーク側のバランスが取れた効率的な信頼インフラストラクチャとが必要とされる可能性がある。セキュリティ要件は、ネットワークに緩く接続され、半自律的に動作する機器については高くなる可能性がある。当業界で検討されるシナリオは、接続する機器の遠隔からの完全性検査または妥当性検査については高レベルの要件を伴う可能性がある。本明細書に記載される方法および装置を使用して、妥当性検査に表現性を与え、効率的、かつ／またはセキュアにすることができる。

【００７０】

ＴＣＧインフラストラクチャワーキンググループの仕様は、この問題への取り組み、すなわち検証済みのコンポーネントと下位コンポーネントを階層的に区別することを含むことが可能である。プラットフォームの構造を表すＴｏＴ（Tree of Trust）の概念および表記を説明する。ＴｏＴのノードは、ＴＰＭからアプリケーションに至るプラットフォームコンポーネントを、信頼およびセキュリティについてのステートメントを注釈として付加して表すことができる。ＴｏＴを使用して、プラットフォームに有するべき信頼を評定することができ、さらには何らかの制約に従ってプラットフォームを再編成することもできる。

【００７１】

信頼を単に線形チェーン化することの弱点が予想される別の技術領域は仮想化である。仮想マシンは、可能性としては多くの層で動的に作成および／または破壊される可能性があり、結果として信頼の依存関係がツリー状の動的な構造となる。当業界は、プラットフォームの信頼性を正確に評定するためには構造化された妥当性検査データが必要となる可能性があることを認識しているが、そのようなツリー型のデータ階層と検証データ（ＰＣＲ値）との細粒度の関連付けがない可能性がある。

【００７２】

検証データおよびＳＭＬは、二分木構造に編成することができる。この構造では、検証データレジスタが根になり、ＳＭＬデータ構造が内部ノードを含むことができ、葉はコンポーネントの測定値となる。全体の構造は、例えばMerkleハッシュツリーのクラスの代表等のツリー型になる。この方法をｎ分木および任意のツリーに一般化することができる。図３のクリスマスツリーを使用して、ツリー型検証の一般的な概念を説明する。

【００７３】

図３は、ツリー型のＳＭＬおよび検証データの概略的な構造を示す。星は、検証データレジスタに記憶されたツリーの根を表す。コンポーネント（コードおよび／またはデータ）は、葉にある包みで示される。コンポーネントの測定結果のハッシュ値は、引き結び（slip knot）で示される。内部ノード（球）は、検証情報を上流方向に根まで伝達する。線は、妥当性検査のためのツリーのトラバースを示唆し、これについては下記でより詳細に説明する。

【００７４】

ハッシュツリーの根ノードに相当する検証データのセキュアな作成は、問題を呈する場合がある。通常の拡張動作では、コンポーネントに対するＲｏＴＭ（Root of Trust for Measurement）で取得される測定値と、現在の検証データレジスタ値V_iとが使用され、動作自体は、ハードウェア保護されたＴＰＭで行われる。

【００７５】

従って、特に、保護せずにＳＭＬに記憶されている測定結果は生成プロセスでは使用されない。これは、新しい葉を追加するとツリーのｄ－２個の内部ノードに影響する可能性

のあるハッシュツリーでは可能でない場合がある。ｄはツリーの深さである。１つの葉の測定値を入力として使用し、ＴＰＭの拡張動作および他のＴＰＭ機能を用いて、限定された数のハードウェア保護レジスタ（ＰＣＲ）の内部にツリー型検証データを生成することが課題となる。

【００７６】

ツリー型検証データを作成し保護するシステムに要求される最小限の要件から、以下で説明する方法および装置は、ＴＣＧ標準に準拠したプラットフォームおよびセキュアハードウェアの要素に限定されないことが明らかになろう。

【００７７】

プログラムの検証は、ロードの前およびブート中に行うことができる。アテステーションも本明細書に記載されるように使用することができる。コード認証は、トラステッドコンピューティングの目標の１つである。実行されるコードは、プラットフォームの起動をセキュアにすることにより保護することができる。例えば、ハードウェア機構を使用して、標準的なＰＣハードウェアでセキュアなコプロセッサを用いてホストへの信頼を実現することができる。信頼されるプラットフォームを適用することができる。セキュアなブートストラップ処理にはセキュアハードウェアが関与することができる。例えば、セキュアなコプロセッサは、異常を検出するとブート処理を停止することができる。これは、ブートストラップＲＯＭがセキュアであることを前提とする。これを保証するために、ブートベクトルおよびブートコードが直接セキュアコプロセッサから供給されるか、またはブートＲＯＭ自体がセキュアなハードウェアとなるように、システムのアドレス空間を構成することができる。

【００７８】

それでも、セキュアコプロセッサは、ソフトウェアの署名を既知の値に照らして検査することにより、システムソフトウェア（ＯＳカーネル、システム関連のユーザレベルソフトウェア）を検証する。コードの耐改ざん性をこのプロセスで実装することができる。この問題への対処法の１つは、例えばＸＯＭ（eXecute Only Memory）プロセッサアーキテクチャ等のハードウェア、およびその上に構築されるＸＯＭオペレーティングシステム等、プログラム実行のための信頼の基礎をハードウェアに置くものである。これは、プログラムのセキュアなロード、および／または外部エンティティへのアテステーションの問題を解決しない可能性がある。ＡＥＧＩＳでは、ＰＣでセキュアなブートを使用する。例えば、ＡＥＧＩＳは、Ｔｅｒｒａと同様に署名されたハッシュを使用してブート処理時に各レイヤを識別する。Ｔｅｒｒａでは、ロードされたコンポーネントを証明書の完全なチェーンで立証する（attest）ことができ、結果として仮想マシンを立証する。

【００７９】

ＴＣＧ仕様は、バイナリの実行可能ファイルを検証することによりプラットフォームの完全性を遠隔から検証する双方向の遠隔アテステーションを定義している。実行コードは、ロード時に測定することができる。測定結果はＰＣＲに検証データとして記憶することができ、ＴＰＭは、ＴＰＭ保護キーでそれらに署名することによりそのデータを立証することができる。ベリファイア（verifier）は、それらの判断基準を受け取ると、プラットフォームを信頼できると見なしてよいかどうかを判定する。構成を送信し、検証することができるため、ベリファイアがマシンの構成を知ることができる。さらに、バイナリのアテステーションは構成を開示し、プライバシー面のリスクを呈する。別の解決法では、「プロパティ」と「プロパティに基づくアテステーション」（ＰＢＡ）が論じられる。ＰＢＡは、詳細な構成データを明らかにすることなく、検証されるプラットフォームのセキュリティプロパティをベリファイアに保証することを可能にする。信頼されるサードパーティ（ＴＴＰ：trusted third party）を使用して、プラットフォーム構成をその構成で実現することができるプロパティ（特に望ましい／望ましくない機能）に対応付けた証明書を発行する。そして、ＴＰＭは、ゼロ知識の証明を使用して、完全な構成を開示することなく、ベリファイアにそれらのプロパティを立証することができる。

【００８０】

ＰＢＡは、ＴＣＧのプライバシー認証機関（ＣＡ）と同様に、しかしその役割を拡張して、プラットフォーム妥当性検査のインフラストラクチャ面の問題をＴＴＰに移す。別の代替法がＮｅｘｕｓ　ＯＳによって提示される。Ｎｅｘｕｓ　ＯＳは、最小限のトラステッドコンピューティングベース（ＴＣＢ）を基礎として、ユーザ空間と特権プログラムとの間に強固な分離を確立する。Ｎｅｘｕｓは、セキュアなメモリ領域並びにその領域を保護する監視および施行マシンを有する。応用の１つは、デバイスドライバをユーザ空間に移動するものである。Ｎｅｘｕｓによるアテステーションは、監視対象のプログラムに記述ラベルを付加し、それによりＰＢＡに似た表現力を可能にするが、システム内在型である。ＰＢＡの概念とＮｅｘｕｓの手法はいずれも、さらに動的に管理すべき多数のコンポーネントからなる複雑なシステムの妥当性を検査する手段を持たない。両者の手法は本発明の手法に直交し、本発明の手法と組み合わせることも可能である。

【００８１】

階層的完全性管理（ＨＩＭ）は、コンポーネント単位の完全性の測定と、ポリシーを利用したプラットフォームコンポーネント管理のための動的なフレームワークを提示する。コンポーネントおよび下位コンポーネントは、この目的に有用な最も一般的な構造である依存関係グラフを介してＨＩＭで関連付けられる。しかし、ＨＩＭは、遠隔からのプラットフォーム妥当性検査を目的としておらず、ＰＣＲ内の構造化されたプラットフォーム検証データを保護しない。そうではなく、ＨＩＭは、グローバルコンポーネント構成レジスタ（ソフトウェアレジスタ）テーブルに複数の測定結果を共に保持する。

【００８２】

大きなデータセットの完全性を保護するための、例えばMerkleによって導入されるハッシュツリー等のハッシュツリーの応用例の１つは、ＰＫＩにおける証明書管理である。これは、Merkleツリーや認証探索木等のツリー構造を使用して、長期間にわたるＣＡの記録責任を生じさせることができる。ハッシュツリーの使用は、一般的な長期間にわたるデジタルデータのセキュアな保管に合わせて拡張することができる。ハッシュツリーは、実行時のメモリ保護にも使用することができる。

【００８３】

システムは、記憶および／またはメモリ保護にハッシュツリーを使用することができ、システムは信頼されない記憶域とＴＣＢに分割することができる。ＴＣＢで実行されるプログラムはハッシュツリーを使用して、信頼されない記憶域に記憶されたデータの完全性を維持することができる。信頼されない記憶域は、例えば、プログラムがＴＣＢに収まらないデータを通常記憶およびロードする、容易にアクセス可能な大容量の記憶装置である。ツリー全体の根は、一定サイズの信頼されるオンチップレジスタに記憶することができるが、他のノードは、メインメモリまたはキャッシュに記憶される場合がある。ハッシュツリーの別の使用例には、無線センサネットワーク（ＷＳＮ）で配布コードの認証をどのように支援することができるかが示されることを含むことができる。またＷＳＮでは、複数のノードに関与するデータ集約の完全性をハッシュツリーを使用して保護することができる。

【００８４】

検証データを探索可能にする別の実施形態は認証済みのアペンド専用スキップリストを含むことができ、これは、「ショートカット」をたどることにより記憶されたデータ要素を高速に探索できるように設計された、ソートされ、互いにリンクされたリストである。

【００８５】

しかし、ツリーは、例えば妥当性検査に合格しない葉ノードのコンポーネントの部分集合を効率的に求める等、プラットフォームの状態の妥当性検査により適している可能性がある。耐改ざん性の検証データレジスタの限定されたセットを使用して、コンポーネントの測定値から、例えば二分木のMerkleツリー等のツリー構造を生成するシステム、方法および装置が本明細書に記載される。ツリー構造は、例えば標準的な拡張動作等のＴＰＭの機能を使用して生成することができる。アルゴリズムは小さくして、ＴＣＢ内部、特にオンチップで実行できるようにすることができる。方法のこの部分でハッシュツリーの根を

生成する際のセキュリティを増すことができ、それによりツリーノードにより高いセキュリティを提供することができる。本明細書には、一般的なＰＣＲ値およびＳＭＬに対して強化された診断能力で効率的な妥当性検査を行うためにツリー構造を活用するシステム、方法および装置も記載され、遠隔のプラットフォームの妥当性検査のセキュリティ機能を増大し、同時に、破綻箇所を探索する際のツリー状構造の効率から利益を得る。このようなツリー構造データの使用は、セキュアな診断、妥当性検査および／またはアテステーションに使用することができる。

【００８６】

本明細書に記載されるシステム、方法および装置は、限定された数の検証データレジスタを使用して、根となる１つの検証値をセキュアに生成することができる。例えばＴＰＭ動作、ＰＣＲおよび／またはＳＭＬ等のＴＣＧで規定されるトラステッドコンピューティングの具体的実施形態の各参照は、本明細書に記載されるシステム、方法および装置の実施で使用される例示的実施形態である場合がある。アルゴリズムおよび／または手順は、それらで使用される最小限の機能と共に、各セキュリティ技術に適用することができる。

【００８７】

ハードウェア保護されたレジスタ、

【００８８】

【数２】

$$v \stackrel{def}{=} \{V_1, \ldots, V_n\}$$

【００８９】

、例えばＰＣＲの１つは、最終的なツリーの根を含むことができる。ツリーは二分木として、アルゴリズムをコンパクトに保ち、例えば不合格コンポーネントを細粒度で検出できるようにする。葉は測定値を保持し、内部ノードは、変更を加えたＳＭＬに記憶することができる。ＳＭＬは、妥当性検査データのツリー構造を支援するように変更することができる。すなわち、測定値の線形リストではなく、そのデータ構造で標準的なツリー操作およびトラバースに対応することができる。プラットフォームの妥当性検査時に効率的に探索できるように、ＳＭＬは、新しい葉の追加をサポートし、エッジ間の関係を保持することができる。ツリーの深さｄの葉に新しい測定結果を追加するには、その葉の縮小ハッシュツリーのｄ－１個のノードおよび／または$V \in v$に記憶されたツリーの根を再計算することが必要になる可能性がある。Merkleツリーでは、２進の拡張動作（１）が非可換であるため、必然的にそれぞれ「左」のエッジと「右」のエッジへのエッジの色分けがある。葉はこの順序を継承し、葉は左から右へと追加される。葉ｎのｄ桁の２進表現、$0 \leqq n \leqq 2^d - 1$（$<n>$と表す）により、葉から根に至る一意の経路にある内部ノードおよびエッジの自然座標が得られる。すなわち、ｋ番目の桁（ＭＳＢから数える。$k = 1, \ldots, d$）$<n>_k = 1$が、その経路上の深さｋ－１のノードがそれぞれ、$<n>_k = 0$または$<n>_k = 1$により、左エッジで接続されるか、右エッジで接続されるかを決定する。

【００９０】

アルゴリズムの実行中に作成される各部分木の根は、$V \in v$にセキュアに記憶することができる。同じ深さｄ’の部分木（測定値は深さ０の部分木）が２つ存在する場合は、それらを併合して深さｄ’＋１の１つのツリーにすることができる。併合操作を使用する場合、部分木の根を保護している２つのＶの１つを併合操作の後に解放することができる。新たに到着する測定値のための更新アルゴリズムは、レジスタ$V_1, \ldots, V_{d-1}$が深さ$1, \ldots, d-1$の「アクティブな」部分木の現在の状態を保持し、V_dが現在のグローバルな根の値を保持できるように作成することができる。

【００９１】

ここで言う「アクティブ」とは、その根が同じ深さの部分木との併合による完成を待っている部分木と説明することができる。作成の際は、実際の測定値、保護されたレジスタ、および／または通常の拡張動作を使用し、非保護のメモリ箇所が関与しないように注意する。深さｄの完全な二分木の空ノードをニル（nil）で表す。ツリーの作成は、図４に

示すアルゴリズム１で行うことができる。

【００９２】

アルゴリズム１に含まれる各種動作には以下がある。

M　V_dに測定結果を追加する；$V_d \leftarrow m$

S_V　検証データレジスタをＳＭＬに記憶する；$V_k \rightarrow$ＳＭＬ

S_m　測定結果をＳＭＬに記憶する；m→ＳＭＬ

V　検証データレジスタをコピーする；$V_k \leftarrow V_{k+1}$

Ｅ１　測定結果でV_dを拡張する；$V_d \leftarrow V_d \diamond V_{k+1}$

Ｅ２　内部ノードレジスタを拡張する；$V_k \leftarrow V_k \diamond V_{k+1}$

【００９３】

上記の記号は、動作とその実行時を入れ替え可能に表している。上記にない１つの動作m←ＲｏＴＭは、S_mに包含することができる。

【００９４】

$n < 2^d$の場合、ツリーは右のエッジで不完全である可能性があり、図６に示すアルゴリズム２に示すクリーンアップ手順を実施することができる。

【００９５】

アルゴリズム２（図６）の結果、V_1が最終的に全ての部分木情報を含むように根が最終的に併合される。このクリーンアップ手順には、図４に示すアルゴリズム１の行１７～２１のテストのためにツリーがまだ完全でない場合に到達することができる。ツリーを完成させる規則は、図５に示す構成が右エッジで成立することである。

【００９６】

内部ノードは、左エッジに沿った移動（forwarding）の結果（軽微な冗長性を伴う）であってもＳＭＬに記述することができる。正式には、ツリーを完成させる上記規則は、本明細書に説明するように、ｘ◇ｎｉｌ＝ｘとなるように「◇」動作の概念を変えることと解釈することができる。

【００９７】

図４に示すアルゴリズム１で規定される順序で葉ノードおよび内部ノードをＳＭＬに追加していくと、結果得られるツリーの自然な順番付けを得ることができる。深さ３の不完全なツリーの場合のこの順序を図７に示す。

【００９８】

図７で、生成されたＳＭＬ内のマークした項目１０および１１は同一である。これは、１１は、クリーンアップアルゴリズム２の移動動作で作成されたためである。ＳＭＬの順序を使用して、二分探索でＳＭＬ中のツリーノードを扱うことができる。長さ$2^{d+1}-1$のＳＭＬ中の通し番号Ｋを与えられると、この探索は、最後の項目である根から進行する。残りの$2^{d+1}-2$個の項目は等しく大きさ2^{d-1}の部分に分割され、Ｋが左部分にあるか右部分にあるかが判定される。この手順は、Ｋが現在の部分の一番右の要素を指すまで繰り返される。行われる判定の連続で、根からＳＭＬ中のインデックスＫのノードに至る左と右のエッジのシーケンスが得られる。

【００９９】

図４のツリー形成アルゴリズム１は、ｂ等、任意の一様のアリティ（arity）のツリーに合わせて容易に適合することができる（関数または演算のアリティとは、その関数がとる引数またはオペランドの数である）。そのためには、２値座標＜ｎ＞を、ｂ値座標$<n>^{(b)}$と、図４に示すアルゴリズム１のそれぞれ行４および行１２で評価されるそのｄ番目の桁とｋ番目の桁に置き換えなければならない。その場合、評価される式を以下のものに変更しなければならない。

【０１００】

【数３】

$$\langle n \rangle \frac{(b)}{d} = b - 1.$$

【０１０１】
アルゴリズム２（図６）もそれに応じて適合することができる。任意のツリーに合わせてさらに一般化するには、関連するノード座標の設定、すなわち、マッピングｎ→ｎｏｄｅの設定が必要となる可能性がある。アリティが２より高い各ノードでは、そのノードに接続する区間のハッシュ拡張は線形になるので、上記の不利点が該当し、検出粒度の低下が発生する可能性がある。
【０１０２】
一般化の手順から、限定された数V_1，．．．，V_rの検証データレジスタで、ツリーの葉にある有限数のコンポーネントをカバーできることが明らかである。最大容量は以下のように計算することができる。第１のレジスタV_1の手順では、ｒ－１個の他のレジスタを長さｒ－１のパイプラインとして使用して、深さｒのツリーを構築することができる。V_1が占有されると、第２のレジスタで深さｒ－１のツリーに対応し、以下同様にして最後のレジスタV_rまで続き、最後のレジスタV_rではパイプラインの長さが０、ツリーの深さが１となる。従って、レジスタのツリーに保持される葉の合計数は以下によって得ることができる。
【０１０３】
【数４】

$$C_{trees}\sum_{k=1}^{r}2^{k}=2^{r+1}-2 \qquad (式2)$$

【０１０４】
ｖ１．２仕様に準拠したＴＰＭのＰＣＲ数であるｒ＝２４の場合は、ｒ個のツリーの葉に３３，５５４，４３０個のコンポーネント測定結果の場所が得られる。例えばＴＣＧのＰＣクライアント仕様ではＰＣＲ０～７が予備なので、最後の１６個のＰＣＲに制限した場合でも、仕様では１３１，０７０個の測定結果を数える。この容量は大きいが、標準は線形に拡張可能であるため、無制限ではない。従って、起動時または実行時に行われる測定数は事前には分からないため、最後のレジスタは、予備として、容量制限に達した時に線形に拡張することができる。図８にこの構成を示し、ツリーの検証データの最大容量の構成を示す。図８では、葉の測定値をｍと示す。
【０１０５】
本ツリー形成アルゴリズムの空間的複雑性は非常に低い。内部データは正確に３つのｄ∈｛１，．．．ｒ｝、ｎ∈｛０，．．．$2^{d}-1$｝、およびｋ∈｛１，．．．ｄ｝を必要とするため、そのデータサイズは最大で
【０１０６】
【数５】

$$d+2\lceil\log_2 d\rceil \le r+2\lceil\log_2 r\rceil d$$

【０１０７】
ビットである。
【０１０８】
また、実装によっては、１つのレジスタが現在の測定値を受け取って保持する必要がある場合があり、かつ／または、検証データレジスタに対する動作用の中間レジスタとして現在の測定値を受け取って保持する必要がある場合がある。ＳＭＬの大きさは緩やかに増大する。深さｄの完全に埋められた二分木の場合は、葉の測定結果を含む$2^{d+1}-2$個のノード値がＳＭＬに記憶される（根ノードはV_iに保持される）。すなわち、ツリー型ＳＭＬは、測定値だけを含む線形に形成されたＳＭＬの大きさの２倍未満である。
【０１０９】
時間的複雑性を推定するために、深さｄの完全なツリー、すなわち2^{d}個の葉の測定値を考える。ツリーの構造から、各動作の発生回数を数えることができる。S_mは、各葉ノードで、すなわち2^{d}回発生する。Ｅ１およびＭは、深さｄ－１の各内部ノードで発生し

、すなわち2^{d-1}回発生する。ＶおよびＥ２は、深さｄ－２から上の各内部ノードで発生し、すなわち2^{d-1}－１回発生する。最後に、Ｓｖは、V_1に残っている根を除くツリーの各内部ノードで発生する。すなわち、S_vは2^d－２回発生する。これらすべてから、フロー制御を無視すると、2^{d-1}（E_1＋Ｍ）＋（2^{d-1}－１）（Ｖ＋E_2）＋$2^d S_m$＋（2^d－２）S_vの推定がアルゴリズムの実行時間として得られる。類似する動作｛E_1，E_2｝、｛Ｍ，S_v，S_m｝をまとめると、2^{d-1}（E_1＋E_2）－E_2＋2^{d-1}（Ｍ＋２S_v＋２S_m）－２s_V＋（2^{d-1}－１）Ｖとなる。

【０１１０】

各種メモリ動作が等しく時間を要し、共通の定数

【０１１１】

【数６】

$$M \approx S_v \approx \frac{1}{2} S_m \approx \frac{1}{2} V \leq S,$$

【０１１２】

で制約されると想定し、ここで単純な読出し／記憶の実装、上記の省略した動作についてのＳｍ、そして同様に拡張動作

【０１１３】

【数７】

$$E_1 \approx E_2 \leq E$$

【０１１４】

に係数２が含まれるとすると、ｄ＞１の場合のツリー形成の時間的複雑性の大まかな推定は、

【０１１５】

【数８】

$$\leq 2^d (E + 4\frac{1}{2} S) \text{---} (E + 4S)$$

【０１１６】

によって与えられる。拡張動作が支配的な要因である場合は、ツリー形成で必要とされる拡張動作は、認証ブートの線形チェーンより１回少ない可能性がある。

【０１１７】

上記手順で生成されたツリー型検証データを妥当性検査するために、各ツリーノードで入手可能な情報を利用する妥当性検査方式を説明する。平均の計算費用を、不合格の測定結果の数およびその相対的な比率との関係で計算することができる。

【０１１８】

通常の認証ブートで生成および記憶され、順次ＰＣＲに拡張された測定結果の線形チェーンを参照事例として取り上げると、ツリートラバースによる妥当性検査は大きく異なることが分かる。線形チェーンの場合、ＳＭＬの操作は基本的には箇所を特定することができず、一方、ツリー型ＳＭＬのトラバースでは操作が行われた部分木を特定することができる。同様の事柄が診断用の妥当性検査、すなわち、妥当性検査対象のプラットフォームの要求基準構成に一致しないコンポーネント（本明細書では不合格コンポーネントと呼ぶ）の探索にも当てはまる。線形にチェーン化されたＳＭＬの場合、この探索には、各測定結果を参照値と比較すること、および／または、ＰＣＲに至るまでの拡張動作のチェーンを再計算してＳＭＬの完全性を検証することが必要となる可能性がある。線形ＳＭＬの操作は箇所を特定することができないため、ＰＣＲ値を再現できないことは、診断用の妥当性検査が可能でない可能性があり、不合格コンポーネントを適正コンポーネントから区別できない可能性があることを意味する。

【０１１９】

ツリー型ＳＭＬの場合は、状況ははるかによい。ＳＭＬの操作が疑われる部分木が特定

された場合でも、ＳＭＬツリー内のその補完を妥当性検査することができる。また、診断用の妥当性検査の場合は、不合格コンポーネントの集合を判定し、同時に根の検証データレジスタの内容を検証する際に大幅な高速化を期待することができる。

【０１２０】

ツリー型ＳＭＬの妥当性検査を使用して、可能な場合は、妥当性検査に合格しない葉の部分集合を見つける、かつ／またはＳＭＬの操作を検出することができる。ローカルに利用できる比較用の参照ツリーがバリデータにあることを想定することができる。妥当性検査は、ツリーの根、すなわち検証データ要素Ｖから開始し、ＳＭＬデータのツリーをトラバースしていく。それにより、測定結果が参照値と異なる、不合格コンポーネントと呼ばれるコンポーネントの葉の集合を得ることができる。ツリーのトラバースでは間引きを行う深さ優先探索を適用することができ、判定は全ての分岐ノードで行うことができる。ツリーは二分木とすることができる。分岐ノードとその２つの子のＳＭＬツリー値を同じノード位置の参照ツリー値と比較し、一致する場合は結果をｇ（ｇｏｏｄ）、かつ／または一致しない場合はｂ（ｂａｄ）と表すことができる。この表記法では、図９に示すように以下のような状況が発生しうる。図９は、ツリー型ＳＭＬのノード構成の分類を示す。

【０１２１】

図９の事例（ａ）では、この親ノードより下の部分木全体の妥当性検査が肯定の結果となり、トラバースはこのノードで終わることができる。図９の事例（ｂ）では、バリデータが子ノード値に拡張動作を適用することにより、親ノードを再計算することができる。再計算された値が親ノードの値と一致しない場合は、根がｂａｄとマークされた部分木の１つでＳＭＬの操作が行われたことを意味する可能性がある。これは例外として処理することができる。

【０１２２】

そうでない場合、妥当性検査は次のツリーレベルに進み、ｂａｄな値が見つかる部分木、すなわち（ｂ）では左、右、または両方の部分木をトラバースすることができる。事例（ｃ）では、ツリー操作の例外を検出することができる。この検出は、拡張動作を再計算することなく行うことができる。最後の事例（ｄ）は、二分木が不完全な場合、かつ／または右の枝がヌルの場合に発生する可能性がある。そして、値ｘが値ｙと等しい場合があり、その場合トラバースは左に進み、それ以外の場合はツリー操作の例外が発生する可能性がある。

【０１２３】

ツリー型ＳＭＬを妥当性検査する利点の１つは、正しい根を持つ部分木を、不合格コンポーネントの以降の探索から除くことができることである。次いで、ツリー妥当性検査の性能を定量的に評定するための単純な確率論モデルについて説明する。例えば、ＳＭＬが深さｄの完全木であるとする。バリデータは、既知の所望のプラットフォーム構成を表す参照用の完全木を有する。ハッシュ演算の再計算が妥当性検査の複雑性を推定する際の主要な費用要因となる可能性があり、対して比較は低費用である可能性がある。不合格の葉ノードの無作為の集合を仮定する。

【０１２４】

診断妥当性検査と呼ばれる楽観的妥当性検査方式を使用することができ、これは、根から、不合格コンポーネント、すなわち、参照ツリーの葉に対してｂａｄな測定値を持つコンポーネントまでの経路をトラバースする。この方式の特性の１つは、真正な測定値を持つ不合格コンポーネントを見つけることである。診断妥当性検査は次のように進行することができる。参照ツリーの対応するノードと異なる内部親ノード、すなわちｂａｄな親ノードに到達する場合は、図９の状況の１つである、事例（ｂ）か、または事例（ｃ）の一番右の構成に遭遇する可能性がある。後者の場合は、明らかなＳＭＬの完全性の破綻であるため、親ノードの再計算は行わなくてよい。この根構成を有する部分木は、不合格コンポーネントについて信頼できる情報をもたらさない可能性があるため、以降のトラバースから除くことができる。この場合、それ以降のステップは、バリデータのポリシーに依存することができる。事例（ｂ）のノード構成は、１回の拡張動作０で根ハッシュから親ハ

ッシュを再計算して、バリデータの参照木からは分からない可能性のある構成が真正であることを確かめる必要のある構成である。根が、調査対象のｂａｄな親ノードのｇｏｏｄな子である部分木は、以降のトラバースから除いてよい。この診断妥当性検査の手順では、構成／事例（ａ）および図９（ｃ）の左から３つの構成を診断妥当性検査から暗黙的に除外する。それらの構成は、意味をなす場合には、ＳＭＬツリーのさらなる鑑定評価で検討することができる。

【０１２５】

診断妥当性検査では、不合格（ｂａｄ）の葉ノードから根までの経路の和集合にあるｂａｄな内部ノードを訪れ、そのノードにハッシュ演算を行うことが必要となる可能性がある。その他の点では改ざんされていないツリーでは、それによりｂａｄな親ノードを含む（ｃ）の右の構成／事例を暗黙的に除外することができる。独立同分布（ｉ．ｉ．ｄ．）のｂａｄな葉の部分集合は、葉の含有比ｆ∈［０，１］を構成する。ｂａｄな葉の数は$2^{d}f$である。ｂａｄな内部ノードの予想数E^{inner}（ｆ）は下記で説明するように計算することができる。

【０１２６】

本発明で対処される問題の１つは、無作為の独立同分布による葉の選択で生成される二分木の二色分け（ｂａｄな内部ノードとｇｏｏｄな内部ノード）と、その部分木を根までつなぐ経路の色分けの問題である。そのような葉と経路の無作為の選択は、長さｄの独立同分布のビット列を無作為に選択することと等価である可能性がある。Ｎ＝2^{d}の葉の集合からｋ回選択した後の色分けされた葉の予想数

【０１２７】

【数９】

$$E_k^N$$

【０１２８】

を計算する。反復的に、

【０１２９】

【数１０】

$$E_0^N=0,$$

【０１３０】

かつ

【０１３１】

【数１１】

$$E_{k+1}^N=E_k^N\frac{E_k^N}{N}+(E_k^N+1)\frac{1-E_k^N}{N}=1+E_k^N\frac{E_k^N}{N}$$

【０１３２】

であり、これを解くと

【０１３３】

【数１２】

$$E_k^N=N(1-(1-N^{-1})^k$$

【０１３４】

が得られる。

【０１３５】

選択されたビット列の全ての部分列は統計的に独立しているので、同じ引数がレベルｄ－１，．．．０の内部ノードに適用される。従って、色分けされた内部ノードの予想数は、ｄ－１の合計

【０１３６】

【数１３】

$$E_k^{inner} = \sum_{l=0}^{d-1} E_k^{2^l}$$

【０１３７】

により得られる。まだ求められていないのはある選択の予想数

【０１３８】

【数１４】

$$E_k^N = fN$$

【０１３９】

に対応する選択の予想回数ｋであり、０≦ｆ≦１は葉の目標含有比である。ｋについてこの式を解くと、

【０１４０】

【数１５】

$$k = \frac{\ln(1-f)}{\ln(1-2^{-d})}$$

【０１４１】

が得られ、Ｎ＝２^{d}が挿入されている。これより、ｆに依存関係にあるｂａｄな内部ノードの予想数Ｅ$_{inner}$（ｆ）を計算することができる。

【０１４２】

図１０に、ｄ＝１６の場合の２^{d}－１個の内部ノードの含有比を示し、ｄ＝１６の場合は上記説明に従ってハッシュ演算を行うことができる。これは、ｂａｄなコンポーネントを確実に判定するために必要とされる可能性のあるハッシュ演算の回数に相当する。線形ＳＭＬの参照事例では、最終的なＰＣＲ値を再計算するために２^{d}＋１回のハッシュ演算が必要となる可能性がある。この事例を図１０の上の縦軸で概略的に表す。

【０１４３】

参照値との比較に関しては、状況が多少異なる場合がある。診断妥当性検査のためのツリーのトラバースは、参照ツリーの対応する内部ノードとの比較に合格しないｂａｄな内部ノードに沿って下方に進行していくことができる。そのために、ｂａｄな内部ノードの両子ノードを各事例で比較する場合があり、よって比較に関する複雑性が数Ｅ^{inner}（ｆ）の２倍になる可能性がある。線形のＳＭＬでは、２^{d}個の測定結果全てを参照値と比較する必要がある場合がある。

【０１４４】

バリデータにおけるハッシュ演算の費用をｈとし、２つのハッシュ値（ＳＨＡ－１の場合は１６０ビット）の比較の費用をｃとすると、線形の場合の総妥当性検査費用は、（２^{d}＋１）ｈ＋２^{d}ｃ＝２^{d}（ｈ＋ｃ）＋ｈ＞２^{d}（ｈ＋ｃ）になる。これは、ツリー型ＳＭＬの診断妥当性検査と同じ情報を線形ＳＭＬから得るための最小の手間である。一方、ツリー型ＳＭＬの場合（根を数に入れる）は、費用は（Ｅ^{inner}（ｆ）＋１）（２ｃ＋ｈ）となる。

【０１４５】

【数１６】

$$\frac{E^{inner}(f)+1}{2^d} \le \frac{h+c}{h+2c} = \frac{\lambda+1}{2\lambda+1}$$

【０１４６】

の場合ツリー型妥当性検査はより効率的になり、λ＝ｃ／ｈ＜＜１である。右側に０．９９の限界を与える非常に大きな余裕λ＜０．０１をとった場合でも。そして、ｄ＝１６の場合、ツリー型妥当性検査は、ｂａｄな葉ノードの含有比ｆが８５％と高い場合に、よ

り効率的である可能性がある。

【０１４７】

ツリー型ＳＭＬの診断妥当性検査は、線形ＳＭＬよりもハッシュ演算に関してより性能が高い可能性があり、ｂａｄなコンポーネントの含有比が大きい場合でも線形ＳＭＬに勝る可能性がある。ツリー型ＳＭＬの診断妥当性検査は、不合格コンポーネントの含有比が小さい場合に非常に有利である可能性がある。ツリーの妥当性検査は、ｂａｄな葉ノードが非一様に分布している場合、例えばクラスタ化を示す場合により効率的である可能性がある。

【０１４８】

直接比較される線形のツリー妥当性検査と診断ツリーによる妥当性検査の両方を利用することが可能であるが、最終的なＰＣＲの再計算が失敗する場合には線形妥当性検査は不可能である場合がある。これは、その場合は、個々の測定結果の比較で信頼できる情報が得られないためである。つまり、各測定結果は、ハッシュチェーンを破綻させた測定結果を隠すためにＳＭＬ内で捏造される場合がある。ツリー型の利点の１つは、バリデータの計算量を減らしても妥当性検査データが得られる点である。

【０１４９】

ツリー形成アルゴリズム自体に関しては、ＴＣＧ標準に準拠する信頼されるブートプロセスと同レベルのセキュリティを達成するために、検証データレジスタに対する動作をハードウェアで保護されたＴＰＭ環境内部で実行することができる。上記で列挙したツリー形成アルゴリズムの大半の動作の一部は、標準に準拠したＰＣＲで実行可能な非標準ＴＰＭ機能であるが、実際には、通常の拡張動作E_1は内部の標準機能であり、S_vおよびS_mはＰＣＲの読出し動作で実現することができる。

【０１５０】

ＴＰＭを拡張してＰＣＲをツリー型検証データレジスタに変えるために必要になる可能性のある最小限の変更を説明し、一方、ツリー形成アルゴリズムはＴＰＭの外側で実行することができる。その後、ツリー形成のためのＴＰＭ内部コマンドについて説明する。ツリー型検証データをソフトウェアベースで実装する別の実装を説明し、その場合、根のレジスタは、信頼されるアプリケーションで管理されるソフトレジスタとすることができ、そのレジスタの現在の状態は「本物の」レジスタ、例えばＰＣＲにより保護される。最後に、ＴＰＭのソフトウェアエミュレーション環境「ｅｔｈｅｍｂａ」に組み込まれたＴＰＭエミュレータによるツリー形成の実験的実装について説明する。

【０１５１】

最小限の手法を採用してツリーの形成を実装し、アルゴリズム１および２と共に使用するＰＣＲを可能にすることができる標準的ＴＰＭへの変更を得る。この手法は、上記の基礎的な動作をＴＰＭコマンドまたはそれに変更を加えたコマンドで実装するものである。内部ノードの現在の状態を表すレジスタに対する記録（bookkeeping）作業を含むアルゴリズムのコアは、認証ブートやセキュアブート等のシステムの完全性測定プロセスでツリー形成を行うソフトウェアのＲｏＴ（root of trust）として実現することができる。

【０１５２】

動作S_vおよびS_mは問題を生じず、TPM_PCRReadコマンドで実現するか、それぞれツリー形成ソフトウェアに直接実現することができる。Ｅｉは、ツリーの最下位レベルの全右エッジで発生する可能性があり、Ｖに拡張された測定結果の左側のシブリング（sibling）から得られるすでに測定された値を保持しているＶを拡張することができる。従って、E_1は、式（１）で定義される標準のTPM_Extend動作に含めることができる。E_2は、ツリー内部の右エッジで発生し、TPM_PCRRead、次いでTPM_Extendでモデル化することができる。

【０１５３】

動作ＭおよびＶは、ツリー内部の最下位レベルの左エッジで発生する可能性がある。これらは、２つの理由から問題を呈する可能性がある。まず、ＰＣＲは直接書き込みされない場合があり、ＭまたはＶにおける最初のステップとしてTPM_PCR_Resetを介してＰＣＲ

をリセットする自然な手法が問題となる場合がある。これは、１６より上の標準ＴＰＭのＰＣＲだけしかリセットすることができず、それも正しいローカリティからしかリセットできないためである。従って、十分なＰＣＲがリセット可能であることと、そのＰＣＲが、ツリー形成ソフトウェアが信頼されるコードとして実行されるローカリティに対応することが必要となる。

【０１５４】

第２に、リセット後も、ＰＣＲを変更できる動作であるTPM_Extendは値を直接レジスタにコピーするのではなく、１６０ビットの２進の０ｘ００であるリセット後のＰＣＲの既存値と、入力値とで（１）を正確に実行することができる。それにより、入力値と異なる結果が得られる。ＰＣＲに直接書き込む、またはＰＣＲ間で値を移動する新しいコマンドを露出することを回避する選択肢の１つは、ＰＣＲがリセット後の初期状態にあることを示すリセットフラグをＰＣＲに備えるものである。そして、そのフラグが真である時にＰＣＲに直接書き込みを行い、その後フラグを偽に設定するようにTPM_Extendを変更することができる。

【０１５５】

ＭおよびＶが常にツリーの左エッジで発生し、右側のシブリングが空（ニル：nil）の場合は関係する２つのシブリングに応じて決定論的に結果を生成することを認識すると、第３の選択肢は、Merkleハッシュツリーの定義からわずかに逸脱するものとなろう。その場合、ＳＭＬツリーの各基本三角形の正しい値設定は図１１に示すようなものとなる。

【０１５６】

すなわち、最初のステップでＶまたはＭをTPM_PCR_Reset、次いでTPM_Extendでモデル化して、０◇ｘ＝Ｈ（０｜｜ｘ）を得ることができる。次いで、右のシブリングをそのレジスタ内で通常通りに拡張し、その結果をＳＭＬに書き込む。それに合わせた中間段階におけるニルのノード値の扱いとツリーの完成についても下記で説明する。多くの場合、ＳＭＬに記憶されるハッシュツリーは不完全、すなわち空の葉ノードおよび内部ノードを含んでいる可能性がある。連続した測定プロセスでは、値がニルと表されたノードは動作ＭおよびＶで手続き的に処理することができ、これは右のニルのシブリングを無視してよいことを意味する。これは、ツリー形成の中間段階であるアルゴリズム１の行８および１８と、最後の測定後にツリーが完成するアルゴリズム２の行２９で発生する。

【０１５７】

一般に、すなわち標準ＴＰＭの制約に背くと、ニルは、演算◇の両側単位元とすることができる。すなわち、ｘ◇ｎｉｌ＝ｎｉｌ◇ｘ＝ｘ、かつｎｉｌ◇ｎｉｌ＝ｎｉｌ（式３）。

【０１５８】

この規約は、上記の規則／事例（ｄ）を証明する。これは通常の拡張動作の再解釈であり、アルゴリズムの構築において演算ＭおよびＶを省くためにも使用することができる。すなわち、ＭおよびＶを、レジスタＶをニルにリセットし、その後それぞれ演算Ｖ←Ｖ◇ｍ、Ｖ←Ｖ◇Ｖ’を行うことに置き換えることができる。

【０１５９】

この規約を実装する場合、ニルは、ＰＣＲレジスタの追加的フラグ並びに◇の入力および出力として表すことができる。ＰＣＲの場合、ニルフラグは特定のリセットコマンドで設定することができる。ＰＣＲの拡張動作の入力としてニルが発生する場合は、ＴＳＳのロジックまたはＴＰＭの変更でハッシュ演算（１）の実行を阻止し、直接ＰＣＲに書き込むことができる。

【０１６０】

ツリー形成をＴＰＭ／ソフトウェアに分割する実装は、結果として得られる根の検証データレジスタ値のセキュリティレベルについて妥協する。ツリー型検証データは、ここで提案するアルゴリズムをＴＰＭ内部で実装することによって生成することができる。そのためには、下記のようにＴＰＭの変更が有効である場合がある。変更されたＴＰＭは、通常のTPM_Extendコマンドと同じ入力パラメータでコマンドTPM_Tree_Extendを公開するこ

とができる。ＴＰＭは、ＰＣＲのうち現在ツリーの根に指定され、占有され、ロックされているＰＣＲ、およびアルゴリズムにより中間Ｖとして使用することができるＰＣＲを示すＰＣＲのフラグを維持することができる。さらに、ＴＰＭは、上記の追加的データを維持する。最も単純な事例では、内部ロジックで、ツリー形成のために２つ以上のＰＣＲを同時に使用することを防止することができる。TPM_Extendは目標ＰＣＲ値の更新を出力することができ、一方、TPM_Tree_Extendは、検証レジスタのデータ値が上記の自然な順序を生成するように、更新後の検証レジスタのデータ値の変数１，．．．．ｄを順次返すことができる。この戻り値は、アルゴリズム１および２のＳＭＬ書き込み動作の出力とすることができる。ｄ個の値が返されると、受け取り側はそのツリーが全て埋まり、対応する根がロックされていることを知ることができる。別の選択肢は、呼び出しのたびに全ての中間Ｖを返すTPM_Tree_Extendを含むことができる。

【０１６１】

Merkleハッシュツリーを使用して、従来ＰＣＲで行われるのと同様にして、信頼されるプラットフォームのセキュアな起動プロセスの完全性を保護する方法が記載される。プラットフォーム妥当性検査においてツリー型検証データを使用すると効率性および柔軟性が増大することが実証されている。これは、特に、ネットワークを介した遠隔妥当性検査およびプラットフォーム管理で有効である可能性がある。ツリー形成アルゴリズムの小ささと複雑性を考えると、仕様がそれに応じて変更されれば、直接ＴＰＭの内部に動作を実装することができる。これは、将来のＴＰＭ生成に実現可能な手法である可能性がある。

【０１６２】

一般化に関しては、ツリーは、暗号ダイジェストを使用した完全性保護を適用できるごく一般的な構造でないことは確かである。例えば、有向グラフの識別を提供するようにハッシュを拡張した研究者がある。他の研究者は、可変の一方向関数、例えばマルチセットハッシュを適用して、ＲＤＦグラフ等の複雑なデータ構造を一意に識別している。これらの方向に沿って、ツリー型検証データを例えば有向非周期グラフおよび依存グラフに一般化することを構想することができる。そのような一般化は、複雑なプラットフォーム管理および保護の作業には潜在的に興味深いが、いずれも複雑性を増大させ、妥当性検査の二分木の効率を損なうと考えられる。従って、そのような一般化のための適用事例はこの段階では検討しない。

【０１６３】

しかし、上記で提案した、ＴＰＭ完全性測定機能の唯一のコマンド拡張であるTPM_Tree_Extendは、柔軟性のある、ＴＰＭ方式のツリー検証データ管理アーキテクチャの開始点とすることができる。例えば動的なプラットフォーム管理のために、そして最終的には、TPM_QuoteがＰＣＲ値について遠隔のバリデータに提供するのと同等のセキュリティ表明でツリー型ＳＭＬの内部ノードをクオートするために、部分木の根のセキュアな更新を可能にすることができる。そのようなさらなる拡張については下記で説明する。

【０１６４】

測定結果を構造化すると、完全性検査に合格しなかったコンポーネントを特定する助けとなる。そのような構造化の一実施形態では、測定結果を線形チェーンに編成し、検証データを$V_i = f(V_{i-1}, m_i)$，$_{i=0,1,\ldots,n-1}$として導出することができ、ｎは、線形チェーンの長さであり、m_iはｉ番目の測定結果であり、V_iは、線形チェーン中のｉ番目の反復から導出された検証データを表す。コンポーネントのｎ個の測定ハッシュ値を処理することによって導出される最終的な値V_{n-1}を検証データとして使用することができる。各V_i：ｉ＝０，１，．．，ｎ－１に、信頼される参照値R_iがあってよい。検証プロセス中に最後の値V_{n-1}がその参照値R_{n-1}と異なることが示される場合は、中間値V_kの少なくとも１つがその参照値R_kと一致しないことを意味する。そのような不一致の原因は、いずれかのコンポーネントの完全性の破綻、または単に正確に計算された（かつ／または完全性が損なわれていない）検証データＶｊと、破損または誤った参照データＲｊとの比較における不一致である可能性がある。測定ログをチェーン（V_0～V_{n-1}）とすると、検証データとその参照値との不一致の最初の発生は、図１２に示すアルゴリズム１を使用

することによって見つけることができる。ｎ個の要素があり、インデックスは０～ｎ－１の範囲であり、インデックスの操作には整数の除算が行われる。

【０１６５】

参照値｛Ｒｉ：ｉ＝０，１，．．．ｎ－１｝は信頼することができ、危険に曝されることはない。この場合、不一致がある場合は、ｉ番目のコンポーネントの完全性の破綻か、または、ｉ番目のコンポーネント自体は完全性が損なわれていないが、その測定結果ｍ自体が損なわれている可能性があることを意味する。いずれの場合も、実用化された検証システムでは、ｉ番目のコンポーネントが信頼できず、ロードする前に修復できることを宣言することができる。アルゴリズム１では破綻の最初の発生を見つけるが、それ以降の破綻は見つけることができない。

【０１６６】

完全性検査不合格のモジュールの探索を最適化するために、上記のツリー型検証手順で説明したように、検証データは、記憶測定ログ（ＳＭＬ）のハッシュツリーの形態で生成することができる。記憶測定ログは平衡二分木として編成され、ＴＰＭコマンドを使用してツリーを生成するアルゴリズムを提示する。このアルゴリズムは汎用的であり、ｎ分平衡木（ｎ＞２）に拡張することができる。

【０１６７】

上記のツリー型検証の概念を拡張して、ツリーが非平衡である場合のツリー型検証を検討する。さらに、拡張してソフトウェアコンポーネント間の相互依存関係を考慮することができる。用語「機器」は、ソフトウェア／ファームウェア／ハードウェアの完全性が検証されるエンティティの意で使用する。エンティティは、ホームゲートウェイ、Ｍ２Ｍゲートウェイ、アクセスポイント、Ｈ（ｅ）ＮＢ、ＦｉＮＢ、中継ノード、モバイル機器、タブレット機器、または任意の他のゲートウェイまたは機器であることに留意されたい。

【０１６８】

一般性を失うことなく、特定のソフトウェアコンポーネントは、他のコンポーネントと比べてより多くの攻撃に曝される可能性がある。例えば、機器内では、装飾機能を可能にするコンポーネントより、通信プロトコルスタックの方が頻繁にハッキングされる可能性がある。この論点は機能にも当てはまる。遠隔の妥当性検査および準自律的な妥当性検査では、完全性の測定結果は、ネットワーク内のプラットフォームの妥当性検査を行うエンティティに送られる。そのような妥当性検査エンティティは、不合格のコンポーネントまたは機能の履歴を観察し、保持することができる。

【０１６９】

また、機器（および／またはそのソフトウェアコンポーネントおよび／または機能）が攻撃を受ける、かつ／または完全性が損なわれた場合、機器の利害関係者に対する影響または費用は、コンポーネント／機能に応じて異なりうることを一般的に想定することができる。

【０１７０】

上記のような側面から、攻撃の頻度および／または費用若しくは影響の重大度を考慮し、検証データ構造と検証アルゴリズムの設計に反映することができるコンポーネント／機能の検証方式が必要となる。

【０１７１】

ただし、まず攻撃の頻度（または発生の確率）が異なる場合を考える。ソフトウェアコンポーネントまたは機能の完全性が破綻する頻度に基づいてソフトウェアコンポーネントまたは機能に対する完全性破綻の確率分布関数（ＰＤＦ）を推定し、検証データ構造の構築で使用することができる。確率分布は離散分布関数であり、ｎ個のコンポーネント／機能を有する機器の場合、ベクトル（P_0～P_{n-1}）と表すことができる。そのような確率分布関数は、全ての観察結果の平均をとるか、最後の十分な数のサンプルに平均化が行われる窓平均で生成することができる。また、平均化は、非加重平均であっても、重み関数Ｗ［ｎ］による加重平均であってもよい。重み関数の例には、これらに限定されないが、指数関数、ガウス関数等がある。重複のある、または重複のないスライド窓を使用して、時

間領域で変動するＰＤＦを得ることもできる。この（１つまたは複数の）確率分布ベクトルは、メッセージングまたは設定により機器に伝えることができる。一般性を失うことなく、ＰＤＦの時間的に静的な集合｛P_0，．．．，P_{n-1}｝を仮定する。

【０１７２】

機器において、ソフトウェアコンポーネントの分布（ＰＤＦ）およびハッシュ完全性測定値を使用して、ハフマンアルゴリズム、算術符号化、または類似の最適符号化ツリー構築アルゴリズムを使用して、最適なハッシュツリーを構築することができる。図１３に、ハフマン符号を使用したツリーの構築例を示す。そのようなツリーを構築する利点は、最も頻繁に攻撃されるコンポーネントの符号長が短くなり、探索時間の予想値が低減することである。従って、不合格のコンポーネントまたは機能を探索する際のオーバーヘッドが低減する。また、最も頻繁に攻撃を受けることが予想されるコンポーネントに最も早く探索が行われる。

【０１７３】

ハフマンアルゴリズムまたは算術符号化アルゴリズムを機能させるために、確率を非ゼロにすることができる。従って、攻撃の確率がゼロのコンポーネントまたは機能は、ツリー構築の際に除くことができる。その結果ツリーのサイズが低減し、通信の過負荷および探索時間を低減することができる。または、全てのノードを含める場合は、攻撃の確率がゼロのコンポーネントまたは機能には、非常に小さい非ゼロのδの値を割り当てることができる。一般に、攻撃の確率分布は非一様なので、生成されるツリーは非平衡木になる。

【０１７４】

ネットワーク内で、そのようなツリーを予め構築し、機器の製造時設定に基づいて、または認証許可され、完全性を検証することが可能なソフトウェア／ファームウェア更新の後に、資格のある機関で導出されたハッシュ値をツリーに入れ、参照のためにセキュアに記憶することができる。機器内では、機器がセキュアにブートされるたびに、ツリーが事前に構築されていない場合は最適な符号木アルゴリズム（ハフマン符号、算術符号等）を適用することにより、攻撃頻度のＰＤＦを使用してそのツリーを再構築して記憶し、ノードおよび根には、評価されたハッシュ値（または拡張したＰＣＲ値）が入れられる。

【０１７５】

ソフトウェアコンポーネントの測定結果を使用して、機器でツリーのノードおよびツリーの根に値を入れるアルゴリズムを開発することができる。そして、そのような検証ツリーを機器からネットワーク内のプラットフォーム検証／妥当性検査エンティティに送信し、参照ツリーと比較して機器の信頼状態を確立し、不合格ノードを一意に特定することができる。このアルゴリズムは本明細書に記載される。

【０１７６】

通信の過負荷を低減するために、深さｄ，ｄ－１，．．．，０に根およびノードを含む、特定の深さｄの間引きしたツリーを送信することができる。深さｄは閾値に基づいて評価することができる。別の間引き方法は、攻撃発生の累積確率が所定の閾値未満であるノードを除くものである。より適当である可能性のあるさらに別の方法は、ツリーに含まれているノードの累積分布値が指定の閾値を超えた場合に、ツリーから全ての残りのノードを間引くものである。図１４に、深さｄ＝２へのツリーの間引きを示す。

【０１７７】

図１５に、システム図および通信の交換を示す。図１５では、符号１で、接続の履歴に基づいてコンポーネント／機能に対する攻撃の離散確率分布を生成する。符号２で、ハフマン符号化アルゴリズムを使用して符号木を生成する。符号３で攻撃の確率分布を機器に通信する。符号４で、ハフマン符号化アルゴリズムを使用してＷＴＲＵで符号化木を生成する。符号５で、セキュアなブート時にツリーに信頼値を入れる。符号６で、信頼ツリーがＷＴＲＵから通信される。符号７で、参照によりツリーを検証し、ツリー内の不合格ノードを探索する。符号８で補正アクションが行われる。

【０１７８】

上記説明では、コンポーネントの非一様な攻撃確率モデルを検討することによって、結

果として生じた非平衡木の使用を考察した。攻撃の確率に加えて、攻撃の費用または影響も考慮することができる。機器のｉ番目のコンポーネント（ｉ＝０，１，．．．，ｎ－１）が｛P_i，C_i｝に関連付けられるとする。P_iは発生の確率を表し、C_iは利害関係者への費用または影響を表す。すると、「攻撃／セキュリティ侵害による予想費用（影響）」に従ってコンポーネントを順序付けることができ、順序付けは、$E(cost)_i = P_i \times C_i$を比較することによって行われる。そして、正規化された分数の予想費用を

【０１７９】

【数１７】

$$Ef(\mathrm{cost})_i \frac{E(\mathrm{cost})_i}{\sum_{k=0}^{n-1} E(\mathrm{cost})_k}.$$

【０１８０】

で求めることができる。

【０１８１】

コンポーネントが（正規化された予想費用が高い順に）順序付けられると、この場合もハフマン符号化または算術符号化を使用してツリーを形成することができる。そのようにして形成された検証ツリーは、破綻が利害関係者にとって最も高い費用／影響を与えうるコンポーネントに最も高い重みを（探索用に）与えることができる。

【０１８２】

ツリー構造は、それ単独ではソフトウェア構造の相互の依存関係を完全に捉えることはできない。例えば、種々のコンポーネントに使用される共有ライブラリがある場合がある。従って、そのような相互依存関係を示すために、最初の最適な符号木を構築した後に、ツリーのノード間およびノードと葉の間にリンクを追加する。そのようにしてツリーを拡張して機器のソフトウェアアーキテクチャまたはコンポーネント間の相互依存関係を捉える。ツリーの葉はソフトウェアコンポーネントに対応付け、ノードをライブラリおよびアーカイブに対応付けることができる。ソフトウェアモジュールと共有ライブラリ間の相互依存関係は、代理の子リンクで対応付けることができる。このようにしてツリーは一般化されたツリーになる。ノードに測定値を入れるアルゴリズムを、追加的に代理子リンクをトラバースするように変更することができる。図１６にそのような代理子リンクを有するツリーを示す。モジュールＡと示すソフトウェアコンポーネントまたは機能は、モジュールＤで表す共有ライブラリに依存または利用し、同様に、モジュールＢで示すソフトウェアコンポーネントまたは機能は、葉のコンポーネント／機能Ｃを利用する。

【０１８３】

機器が、アルゴリズムの信頼される動作、並びに／または、導出されたハッシュ値をノードおよび／若しくは根に信頼される形で入れることを保証する例えばＴｒＥ等の信頼される実行環境を備える場合、アルゴリズムは、ＴｒＥによって提供される信頼される動作および命令を使用することができる。

【０１８４】

信頼されるプラットフォームモジュール（ＴＰＭ）およびその上位層の信頼されるソフトウェアスタック（ＴＳＳ）を使用してそのようなＴｒＥを構築する場合は、ＴＰＭ並びにＴＳＳに適切なコマンド拡張を検討することができる。

【０１８５】

上記のように、任意の最適な符号構築アルゴリズムを使用してツリーを構築することができる。上記ではハフマンアルゴリズムを使用してツリーを構築する例を提示した。

【０１８６】

以下の説明では、葉がソフトウェアコンポーネントまたは機能に相当する。説明を簡潔にするためにコンポーネントを参照するが、議論は機能にも該当する。耐衝突性のハッシュ関数Ｈを使用してコンポーネントの測定結果を計算することができる。対象ノードの子の判断基準を使用してノードの判断基準を計算するために関数Ｆを使用するとする。アルゴリズムPopulate_metricは、ノードを最適な符号木の根に設定して呼び出すことができ

る。ツリーの葉の判断基準を初期化して、対応するコンポーネントの測定結果にする。

【０１８７】

図１７に示すアルゴリズム２は、ツリーの順序付け後にトラバースを行い、内部ノードの判断基準を計算し、計算された判断基準値をノードに入れる。ＴＰＭに合わせてアルゴリズムを適合するために、関数ＦはPCR_Extendの動作とする。以下のアルゴリズムでは、Reg_Numberは、ＰＣＲレジスタ番号を示す添え字索引である。

【０１８８】

図１８に示すアルゴリズム３は、アルゴリズム２（図１７）を変更して、ＰＣＲレジスタを用いて機能し、PCR_Extend動作を使用するようにしたものである。それぞれ図１７および図１８に示すように、アルゴリズム２および３はｎ分木で機能するように変更することができる。ただし、それに応じてハフマンアルゴリズムを変更してツリーを生成してもよい。最も低い２つの確率を左および右の子に追加する代わりに、ｎ個の最も低い確率が追加される。従って、枝のラベリングはｌｏｇ２（ｎ）ビットを含む。例えば、４分木の場合は、４つの確率が追加され、４つの枝を００、０１、１０、１１とラベリングする。図１９に示すアルゴリズム４は、ｎ分木用にアルゴリズム２を更新したものである。そして、図２０に示すアルゴリズム５も同様に更新することができる。

【０１８９】

上記のように代理子リンクを扱うように一般化されたツリーを構築するために、図１８に示すアルゴリズム３を更新することができる。アルゴリズム５は、代理子リンクを考慮したアルゴリズム２の更新である。最初に、ハフマンアルゴリズムを使用して二分木を構築する。後に代理子リンクとして相互依存関係を追加する。左および右の子に付加されるラベル０および１は、この場合も葉の符号を見つけるために使用される。代理子リンクを使用してノードに判断基準を入れることができる。代理子リンク自体にはラベルは付加しなくてよい。

【０１９０】

ノードに判断基準を入れると、検証ツリーをネットワーク内の妥当性検査エンティティに送信することができる。完全性ツリーと参照ツリーとの比較が行われる。これは、図２１に示すアルゴリズム６に従って行うことができる。

【０１９１】

図２１に示すアルゴリズム６は、内部ノードの依存関係が存在するために代理子リンクを持たない（一般には非平衡の）ツリーに有効である可能性がある。

【０１９２】

代理子リンクを持つツリーを処理する場合も同じアルゴリズムを適用することができる。その場合、アルゴリズムは、代理子リンクの関係には何も特別なことは行わずに実行される。そのようにして実行されても、アルゴリズムは、不合格コンポーネントからなる同じ間引きツリーを得る。しかし、代理子リンクは、破綻に追加的な可視性を与える。これは、このアルゴリズムに従って完全性検査に合格しないコンポーネントが特定された時、代理子リンクの関係が、その特定された不合格コンポーネントの影響を受ける可能性のある他のノード（不合格コンポーネントのすぐ上に位置するとは限らない）についての示唆を与えるためである。

【０１９３】

上記アルゴリズムの別の変形例では、「双方向」方式で上記および同様の欠陥検出アルゴリズムを実装することができる。一実施形態では、例えば再度図１４を参照すると、送信側は、特定の内部深さｄ≦Ｄまでのみのノードについて測定結果を送信することができ、ここでＤは最も長い枝の深さである。ベリファイアは、それらの内部ノードを検証することができる。

【０１９４】

破綻が検出された場合、ベリファイアは、その不合格の内部ノード以下の部分木に対応するさらなるデータを送信側が送信することを要求することができる。送信側は要求に応え、ベリファイアはより深いレベルの測定結果の完全性を調べることができる。この方法

は、２回のみの対話ではなく、そのような対話をより多く含んでよい。

【０１９５】

上記のような「双方向」プロトコルは、例えば通信の帯域幅やペイロードの制限の理由で、ツリーの全ノードについてのツリー型ＳＭＬを一度に全て送信することが難しい場合に使用すると理にかなう可能性がある。完全に形成された二分木が含むデータは、ＳＭＬの線形チェーン構造のデータの高々２倍である。送信する必要があるデータの追加量はツリー形成の深さと完全度に依存するが、事例によっては、完全に形成された二分木の場合でも、送信する必要のあるデータの追加量は多くなく、例えばもう数キロバイト分の測定値である。しかし、双方向型のプロトコルは、詳細な証明書や、ＲＩＭ証明書、ＯＭＡＤＭ情報等、より表現力の高いセマンティックな妥当性検査データでツリーに注釈を付け、ツリーのデータサイズが大きい場合に、検討することがより妥当である可能性がある。

【０１９６】

二分木形態の妥当性検査データ構造は、妥当性検査システムの最適化基準の特定の選択の結果生じる可能性があり、最適化基準は、１）内部ＳＭＬに対する侵害を検証するための確実な手段、および、２）侵害されたコンポーネントの高速な探索である。しかし、二分木形態の妥当性検査データ構造は以下のような費用を招く。１）機器とベリファイアの両方でツリーを完全に構築するための計算費用の増大、および、２）内部ＳＭＬを記憶する必要性による記憶費用の増大（最高で線形ＳＭＬの量の２倍）。機器の各種コンポーネント間に相互依存関係が多くある場合は、ツリーは最適な構造とは限らない。

【０１９７】

他の種のシステム最適化基準では、結果としてツリーとは異なる可能性のある妥当性検査構造が生じる場合がある。例えば、高速で細粒度の破綻検出能力を犠牲にして内部の記憶媒体を絶対的に最小限にすることに関心がある場合には、線形のＳＭＬ構造が最良の選択肢である可能性がある。

【０１９８】

別の可能な最適化基準には、隣接するコンポーネントが構造化されていない高い相互依存関係を有することが分かっている場合に、危険に曝されている既知の１つまたは少数のコンポーネントの近傍（小さな長方形や周囲の円等）の完全性を迅速に判定する際の容易さ／速度がある。そのような場合は、例えば相互につながった輪やメッシュのように見える非ツリー型の妥当性検査データ構造が生じる可能性がある。この話題をより一般的に扱うために、グラフ理論による検査を応用することができる。

【０１９９】

ツリー型の保護された検証データレジスタによるセキュアな動作について以下に説明する。信頼されるプラットフォームモジュール（ＴＰＭ）に機能性を概念的に追加して、ツリー型記憶測定ログ（ＳＭＬ）の完全性を保護するハッシュツリーの根に相当するプラットフォーム構成レジスタ（ＰＣＲ）を扱う。これによりＳＭＬの内部ノードの検証と更新が可能になり、さらには、通常のＰＣＲと同じセキュリティレベルでノードの値を立証することも可能になる。重要な応用例として、ＳＭＬ部分木の証明がどのようにしてプラットフォームプロパティのアテステーションを可能にするかを示す。

【０２００】

より多くの信頼される機能を追加して、ツリー型検証データに作用することができる。根が検証データレジスタで保護されているツリー型ＳＭＬの内部ノードは、全体的なセキュリティレベルを維持しながら、制御された方式で、その完全性を検証し、新しい値で更新できることが示される。内部ツリーノードのためにTPM_Quoteコマンドの変形を導入し、これは、TPM_Quoteが通常のＰＣＲの値に対して行うのと同程度に正確に、ノードの完全性を立証することができる。定義されたコマンドのセットと共に、ＴＰＭの完全性測定機能を、ツリー型ＰＣＲおよびＳＭＬに動作する包括的なコマンドのセットで補うことができる。それらのコマンドを使用すると、線形にチェーン化された通常のＴＰＭ　ＰＣＲおよびＳＭＬに比べてはるかに高い柔軟性および表現力でツリー型検証データおよび／または妥当性検査データを使用することができる。

【０２０１】
基本のシステムモデルおよび表記体系も説明する。また、上記のＴＰＭコマンドの拡張も提供して定義し、また、それらの動作に関連するいくつかの構造上の拡張および基本の使用カテゴリも提供される。導入されるコマンドの主要な使用事例として、信頼される第３者によるツリー型ＳＭＬの部分木の根ノードを証明するためのプロトコルの実証と、ツリー型のプラットフォーム妥当性検査手法の簡単な評価と、今後の研究の見通しについても述べる。
【０２０２】
後に必要となる可能性のあるプラットフォームの最小限の要素および能力について説明する。ＴＣＧの用語およびいくつかの概念を本明細書で使用し、記載するが、本明細書に記載される実施形態は、ＴＰＭやＳＭＬ等のＴＣＧ標準に準拠したプラットフォームおよびセキュアハードウェア要素、並びに例えばＰＣクライアント仕様に準じて設計されたシステムに限定されない。
【０２０３】
上記で定義した拡張動作のツリー形成の変形はＴＰＭの内部で動作し、１つの測定値を入力として受け取るが、その他の点ではＴＰＭ外部のシステムに関しては能動的に作用しない。これは、本明細書に記載される更新機能には当てはまらない。更新機能は、ＴＰＭの内部で保護される特定数ｒの検証データレジスタＶ，
【０２０４】
【数１８】

$$V \stackrel{def}{=} \{V_1, \ldots, V_r\}$$

【０２０５】
、およびＴＰＭ外部の低保護の記憶域に記憶されたハッシュツリーデータに作用する。
【０２０６】
すなわち、記憶測定ログ（ＳＭＬ）に保持されたハッシュツリーは、更新動作に必要なＴＰＭ機能にアクセスする権限のある信頼されるソフトウェアスタック（ＴＳＳ）で管理することができる。ＴＳＳは、許可され完全性が保護されたコマンドインタフェースを介してＴＰＭを呼び出す。実際的な理由でＴＣＧを一般的な語として使用するが、ここに提示する概念は、ＴＣＧ、ＴＰＭ、およびプラットフォームに制限されない。ハードウェア保護された検証データレジスタのセットと拡張動作を使用することができる。後者は、通常のＴＰＭ拡張動作、
【０２０７】
【数１９】

$$V \leftarrow V \diamond m \stackrel{def}{=} H(V \| m) \quad (式4)$$

【０２０８】
で定義することができる。Ｖは、検証データレジスタを表し、Ｈは、耐衝突性のハッシュ関数（ＴＰＭの場合はＳＨＡ－１）であり、ｍ＝Ｈ（データ）は測定値である。以下では、混乱が生じないように、任意のレジスタＶに関して引数として◇を広く使用する。
【０２０９】
ＳＭＬは、上記のTPM_Tree_Extendコマンドで生成された、例えばMerkleハッシュツリー等の２項の一方向演算から得られた深さｄの二分木を含むことができる。内部ノードおよび葉の自然座標は、長さ１，．．．ｄの２進列であり、列の長さ１が、ノードが存在するツリー中のレベルである。内部ノードまたは葉をｎとし、ｎの座標の２値表現をｎ～＜ｎ＞＝（n_1，．．．，n_1）∈｛０，１｝^{x1}と表記する。＜ｎ＞のｋ番目の桁を＜ｎ＞＝n_k，ｋ＝１，．．．１とする。混乱が生じないように、ノードは、ＳＭＬにおける自身の値（例えば１６０ビットのハッシュ値）で識別することができ、その座標と区別する。その他の場合は、ノードの値と座標との対についてｎ＝（ｎ，＜ｎ＞）となる。
【０２１０】

ｎのトレースＴは、ｎから根に至る経路にあるｎを含む全ての内部ノードの順序付けリストになり、すなわちＴ（ｎ）＝（t_1，．．．．．t_1）であり、t_k～（n_1，．．．．n_k）である（式５）。

【０２１１】

ノードの自然半順序をｍ≦ｎと表記し、これはｎ∈Ｔ（ｍ）と等しい。半順序は、∀ｍ∈Ｍ：∃ｎ∈：ｍ≦ｎの時かつその時に限りＭ≦Ｎを設定することにより、ノードの集合Ｍ、Ｎにも成立する。

【０２１２】

ｎの縮小ツリーＲは、そのトレースにある全てのシブリングのリストになる。これは容易に自然座標Ｒ（ｎ）＝（ｒ１，．．．．ｒ１）、ただし

【０２１３】

【数２０】

$r_k \sim (n_1,\ldots \neg n_k)$

【０２１４】

（式５）で表され、

【０２１５】

【数２１】

¬

【０２１６】

は２項の否定を表す。

【０２１７】

引数の順序を２値パラメータに依存させる変形例では、ハッシュチェーン演算

【０２１８】

【数２２】

$x \diamond y \quad \underline{def} \quad H(x||y)$

【０２１９】

を固定長の入力ハッシュ値に使用する。ｃ∈｛０，１｝について、以下を設定することができる。

＜ｘ｜｜ｙ＞＝　｛ｃ＝１の場合ｘ◇ｙ；

＜ｘ｜｜ｙ＞＝　｛ｃ＝１の場合ｙ◇ｘ；

【０２２０】

このカイラル（chiral）のMerkle-Damgard演算は、ツリー内の左のシブリングと右のシブリングを区別し、それらの親ノードを正しい順序で計算できるようにする拡張動作の１バージョンである。実装上の問題を無視すると、（拡張された）ＴＰＭは＜．｜．｜．＞＝を内部で行うことができる。

【０２２１】

事例によっては、ＳＭＬに記憶されたハッシュツリーが不完全、すなわちニルで表される空の葉と内部ノードを含んでいる場合がある。Merkleハッシュツリーのニルノードを一貫性をもって扱うために、ニルが、演算◇の両側単位元である、すなわちｘ◇ｎｉｌ＝ｎｉｌ◇ｘ＝ｘ、かつｎｉｌ◇ｎｉｌ＝ｎｉｌ（式６）と仮定すると有用である。

【０２２２】

これは通常のＴＰＭ拡張動作の再解釈であり、まずＶをニルにリセットし、次いで何らかの値ｘについてＶ◇ｘを行うことにより、Ｖ∈ｖへの直接の書き込みをモデル化するためにも使用することができる。この規定を実装するために、ニルを検証データレジスタ並びに＜．｜．｜．＞の入力および出力のフラグとして表すことができる。Ｖに対して、ニルフラグは特定のリセットコマンドで設定することができる。Ｖの拡張動作の入力としてニルが発生した場合は、ＴＳＳのロジックまたはＴＰＭの変更で、拡張およびＰＣＲへの書き込みを直接阻止することができる。

【０２２３】

ツリー型ＳＭＬに対してセキュアに動作するようにするための標準ＴＰＭの動作上の拡張について説明する。保護の目標は、ＰＣＲ内で保護される従来の検証データレジスタ値と同じ保証レベルを、そのようなＳＭＬの内部ノードおよび葉に実現することである。新しいノード値による根ノードの更新について初めに説明し、その後、ツリー型検証データに使用するための構造およびコマンドの拡張についてさらに説明する。

【０２２４】

ＳＭＬツリーの内部ノードまたは葉のセキュアな更新の方式は以下の通りである。まず、そのノードの現在の値の真正性を検証することができる。これは、当該ノードに関連付けられた縮小ハッシュツリーに保持されているデータを使用して、レジスタＶ（説明を簡略にするためにこの項の残りの部分では固定する）で保護されるツリーの根を再計算することによって行われる。この検証は、TPM_Reduced_Tree_Verify_Loadと呼ばれる、ＴＰＭ内部の保護された動作でなければならない。この動作ではまた、その次の更新動作TPM_Reduced_Tree_Updateで使用するために、検証データレジスタのセットに、検証済みの縮小ツリーデータをロードする。この機能は、更新するノードの新しい値を取得し、縮小ツリーデータを使用して根Ｖに至るまでの親ノードを更新する。両コマンドは、各種目的、例えば独立したノードの完全性検証に単独で使用することができる。適宜、両コマンドを組み合わせてノードおよび根を更新する１つのコマンドとしてもよい。

【０２２５】

根が検証データレジスタＶ∈ｖで保護されている、深さｄのＳＭＬツリーのレベル１≦ｄにあるノードをｎとする。最初のステップでは、ｎの新しい値でＶを更新し、縮小ツリーＲ（ｎ）がＳＭＬ中で改ざんされていないことを検証する。

【０２２６】

Ｖのセキュリティレベルを維持するために、この検証もＴＰＭで保護された動作によって行うことができる。そのために、ＴＳＳは、引数（（＜ｎ＞，ｎ，Ｒ（ｎ））でTPM_Reduced_Tree_Verify_Loadを呼び出す。Ｖから１＋１個の利用可能なレジスタを選択し、それらをB_1、．．．．．．B_1、およびV^*と呼ぶ。図２２に示すアルゴリズム１は、どのようにＳＭＬノードが検証され、その縮小ツリーが検証データレジスタのセットにロードされるかを示す。

【０２２７】

このアルゴリズムで中心的に使用されるカイラルな拡張は、ｎのトレース上にある親要素の計算時に子ノードの順序が正しくなることを保証する。ＴＳＳは、計算されたトレースＴ（ｎ）および検証ステータスを戻り値として得る。図２２に示すアルゴリズム１では１＋ｌ個の追加的な検証データレジスタがある。

【０２２８】

図２２のアルゴリズム１の単純な変形は、縮小ツリーを順次処理することにより、縮小ツリーをＴＰＭ内部に記憶することなく、単一の検証データレジスタを使用して動作することができる。この補助コマンドTPM_Reduced_Tree_Verifyは、ＴＳＳまたは他のパーティによるＳＭＬの簡易な検証に有用である可能性がある。これを図２３に示すアルゴリズム２に示す。この変形例で必要とされるＲ（ｎ）の順番付けは、ＴＳＳ以下のソフトウェア層で実現される入力バッファ、例えばＴＰＭデバイスドライバ、または対応するＴＰＭ内部ロジックを使用して行うことができる。

【０２２９】

上記の元のツリー形成アルゴリズム（図４および図６）と同様に、アルゴリズム１および２（それぞれ図２２および図２３）は、非標準的な動作、詳細にはカイラルな拡張を使用する。カイラルな拡張の出力先は常に検証データレジスタなので、この動作は、カイラルな拡張の中間の引数に応じて、他方の引数を別の検証データレジスタ（図２２のアルゴリズム１のように、すでに検証レジスタがあるのでない場合）にロードすることにより、レジスタスワップを行ってＴＰＭ内部動作◇より先に行うことによって実装することができる。これにより、全ての引数に同じ保護レベルを保証することができる。

【０２３０】

それぞれ図２２および図２３のアルゴリズム１および２で行われる検証は、入力された縮小ツリーに対する入力ノード値の完全性を保証するので、効果が限定される。ＳＭＬツリーの完全性が侵害された場合は、より詳細な情報が望まれる。上記の下方へのツリートラバースを介した妥当性検査方式を行うことにより、少なくとも完全性の侵害が発生したツリーレベルを取得することができる。

【０２３１】

アルゴリズム３（図２４ａ）に示すコマンドTPM_Tree_Node_Verifyは、正しくない縮小ツリーおよび／またはトレースの要素が根からｎに至る完全性チェーンを最初に断絶させたレベルを返す。このコマンドでは、チェーンを断絶したシブリングは判定することができない。上記の参照ツリーを利用できる場合は、さらなる診断が可能である場合がある。

【０２３２】

TPM_Reduced_Tree_Verify_Loadは、新しい値ｎ’で更新することが可能なノードｎに行うことができる。アルゴリズム４（図２４ｂ）に示すコマンドTPM_Reduced_Tree_Updateは引数ｎ’で呼び出され、求められた先行するTPM_Reduced_Tree_Verify_Loadの結果に作用することができる。TPM_Reduced_Tree_Verify_Loadは、更新するノード座標（ｎ）およびレジスタＶを固定する。このバインドをコマンドシーケンス内で実現するには各種方法を用いることができる。

【０２３３】

初めに、ＴＰＭは、下記で説明するように、ツリー操作のための状態および追加的データを記憶し、管理することができる。さらに、コマンドのシーケンス、TPM_Reduced_Tree_Verify_LoadおよびTPM_Reduced_Tree_Updateを、例えばＴＰＭで保護されるＯＩＡＰ／ＯＳＡＰの正式なコマンドセッションで実装されるようにローリングノンス（rolling nonce）で暗号的にバインドすることができる。最後に２つのコマンドを結合して１つの更新コマンドTPM_Tree_Node_Verified_Updateとし、引数は（＜ｎ＞，ｎ，ｎ’Ｒ（ｎ））である。このノード更新コマンドは、ｎ’’の新しいトレースおよびＶの新しい値を返し、ＴＳＳはそれらでＳＭＬを更新する。

【０２３４】

検証データレジスタをハッシュツリーの特定のノードまたは根に関連付け、それに関連するコマンドTPM_Tree_Extend（上記で定義）、TPM_Reduced_Tree_Verify_Load、TPM_Reduced_Tree_Updateを用いて、それらのレジスタＶはステートフルネス（statefulness）を取得する。特定の重要度である状態は以下である。

【０２３５】

Active Root（ＡＲ）。TPM_Tree_Extend動作で現在構築中のＳＭＬツリーの根を表す。

【０２３６】

Complete Root（ＣＲ）。いくつかのコンポーネントの測定、すなわちTPM_Tree_Extend動作の完了結果としてのツリーの根を表す。ＡＲは、ツリーが完全である時、すなわち２ｄ個の葉ノード測定結果を含む時、またはある段階でツリーを閉じることが求められる場合にＴＳＳからトリガでＣＲに遷移することができる。ＣＲ状態にあるＶは、TPM_Tree_Extendによるさらなる更新から保護することができるが、ポリシーおよび権限付与によっては、TPM_Reduced_Tree_Updateから、さらには通常のTPM_Extend動作からもアクセスされる場合がある。

【０２３７】

Tree Build（ＴＢ）。TPM_Tree_Extend動作により、別のＡＲレジスタにアクティブツリーを構築するために使用されるレジスタを表す。

【０２３８】

Reduced Treeノード（ＲＴ）。TPM_Reduced_Tree_Verify_Loadの結果、すなわちレジスタB_kの１つを表す。ＲＴ　Ｖは、対応するTPM_Reduced_Tree_Update、または別の許可されたコマンドが発生するまで保護されることができる。

【０２３９】

２つ以上のツリーを管理する場合は、例えば一意の識別子（ＵＩＤ）を使用してＶの状

態を個々のツリーに関連付ける必要がある。さらに、各レジスタまたはいくつかのレジスタについて、ノード座標を記憶する必要がある場合がある。これらのデータは、ＴＰＭ内部の検証データ割り当てテーブル（ＶＤＡＴ）に保持され、ツリーデータ管理ユニット（ＴＤＭＵ）により管理することができる。

【０２４０】

ＴＰＭで保護されたノード値検証は、アテステーションによるプラットフォーム妥当性検査の新たなコアセマンティックを可能にすることができる。詳細には、あるノード値を立証するTPM_Quoteの変形を定義することができる。そのようなコマンドであるTPM_Tree_Node_Quoteは、TPM_Quoteと同じ引数に加えてTPM_Reduced_Tree_Verifyの引数で呼び出すことができる。するとこのコマンドは図２３に示すアルゴリズム２を実行するが、別のＰＣＲ　Ｖ’にｎのコピーを追加的に保持する。成功すると、Ｖ’にTPM_Quoteを実行する。そのようなクオートの受け取り側は、得られた署名がＳＭＬツリーの内部ノードに対するものであることを認識することができる。可能性の１つは、TPM_Quoteコマンドの署名済みブロブ（blob）に含まれる固定列を変更するものであり、通常は「QUOT」であるのを、例えば「TREEQUOT」に変更する。

【０２４１】

このコマンドによるノード値のアテステーションは、TPM_Quoteで検証データレジスタ（ＰＣＲ）値をクオートする場合と同じセキュリティ表明をノードに提供する。ただし、このアテステーションは、その値がＳＭＬツリーのいくつかの内部ノードに対応するという追加的なセマンティックを有することができ、すなわち、ｎが根である特定の部分木の状態を有効に立証する。このセマンティックをバリデータに明示的に伝えるために、ＡＩＫ（Attestation Identity Key）で署名されたアテステーションパッケージに追加的なデータ、例えば文字列「Tree Node」を含めることができる。そのような属性の効果は、根のレジスタが、上述のようにTPM_Tree_Extendコマンドの結果得られる管理されたＳＭＬの根のレジスタである場合、すなわちＣＲ状態にある場合にTPM_Tree_Node_Quoteによって割り当てられた場合は、著しく強化することができる。この管理は、クオート生成の一部とすることができる。

【０２４２】

アテステーションメッセージを妥当性検査するために、バリデータは、クオートされたノードｎ＝（ｎ，＜ｎ＞）の値ｎを必要とする場合がある。バリデータにさらに情報を転送することは必要でない可能性があり、TPM_Tree_Node_Quoteの上記説明は最小限の暴露の原理に従う。このコマンドの変形では、ＳＭＬツリー内のノードの位置が妥当性検査に関係する場合は、ノード座標（ｎ）に署名することもできる。バリデータに送られる拡張された妥当性検査データは、ｎの縮小ツリーおよび根の検証データレジスタを含むことができる。

【０２４３】

代替実施形態として、バリデータに縮小ツリーの検証を課すことも可能である。この手法は、ウェブサーバから配信されるウェブページの完全性を立証し、通常のTPM_Quoteコマンドを使用してその完全性をサーバの状態のアテステーションにバインドするために、他の事例で使用される。これにより、下記のようにTPM_Tree_Node_Quoteの変形例が実現される。

【０２４４】

このコマンドは、ノード値ｎ、Ｒ（ｎ）のノード値、および根Ｖの選択子を引数として受け取る。ＴＰＭは、ＶのＣＲ状態を制御した後に、「REDTREEQUOT」の固定列属性でこの（連結された）データに署名する。ウェブページの完全性を立証するために使用されるこの手法には、追加的なデータのハッシュをTPM_Quoteコマンドのノンスの入力に挿入すること（通常はクオートの新しさを保証するために使用される）により、縮小ツリーをＴＰＭからのクオートにバインドする回避策が使用されることに注目すべきである。

【０２４５】

内部ノードをクオートする第１および第２の実現の変形例は、第１の変形例はプラット

フォームに検証の負荷を課すのに対し、第２の変形例はバリデータに負荷を課す点で、互いに正反対の可能性を示す。従って、両変形は、実際的な効率の領域が異なる可能性がある。例えば、第１の変形例の場合は、Ｍ２Ｍ通信のように多数の分散した妥当性検査プラットフォームがあり、第２の変形例の場合は、分散した多数のバリデータ（上記のウェブクライアント等）がある。ただし、第２の変形例は、バリデータにＶで表される完全な状態が公開されるため、原理上プラットフォームの情報暴露に関して欠点がある。これはプライバシーを害する可能性がある。

【０２４６】

本発明で導入されるＴＰＭ完全性測定機能の各種拡張は、以下のカテゴリに分類することができる。TPM_Tree_Extendは、ＰＣＲで根が保護された特定のＳＭＬツリーを構築する連続的な測定プロセスで使用される。TPM_Reduced_Tree_Verify_Load、TPM_Reduced_Tree_Verify、そして最後にTPM_Tree_Node_Verifyは、プラットフォームの内部診断のためのコマンドである。TPM_Reduced_Tree_Verify_LoadをTPM_Reduced_Tree_Updateの準備ステップとして使用することを別にすれば、これらのコマンドは、他の事象が発生する前にＳＭＬ部分木で表されるプラットフォームの特定のプロパティを検証するために使用することもできる。

【０２４７】

TPM_Reduced_Tree_UpdateおよびTPM_Tree_Node_Verified_Updateは、部分木の制御された更新に使用することができる。内部ノード更新動作の特定の使用例について以下のように説明する。

【０２４８】

単一のシステムコンポーネントの更新。この場合は、新しい値で葉を更新する。

【０２４９】

部分木で表されるシステムモジュールの更新。この場合は、新しい部分木の根で、元のツリーの内部ノードを更新する。

【０２５０】

最後に、TPM_Tree_Node_Quoteは、ツリー型ＳＭＬを、遠隔パーティによるプラットフォームの妥当性検査に使用できるようにするコマンドである。このコマンドは、ツリー型データを使用する妥当性検査の新しい主要要素を示し、すなわち、システム状態の定義部分のみを表す部分木を立証する可能性を示す。具体的な使用例は下記で説明する。

【０２５１】

コマンドの拡張は、ツリー型妥当性検査において単独で使用することができる。以下に、コマンドの拡張を組み合わせてモジュール方式で部分木を妥当性検査する方法を説明する。

【０２５２】

ツリー型ＳＭＬおよび検証データレジスタを使用した妥当性検査は、遠隔アテステーションにセマンティックを追加する。ＳＭＬの部分木を立証できることにより、従来の遠隔アテステーションをはるかに上回る表現力が可能になる。ツリー型検証データは、プラットフォームアテステーションにセマンティック（例えば、ＰＢＡのようにプロパティと妥当性検査データの関連付け）を追加する他の提案を具現化する１つの方式である。トラステッドコンピューティングの問題の１つは、プラットフォームアテステーションへのセマンティックの関連付けである。ＴＣＧ仕様は、ロードされる際に実行コードが測定される双方向の遠隔アテステーションを定義する。測定結果は検証データとしてＰＣＲに記憶され、ＴＰＭは、ＴＰＭで保護されたＡＩＫ（Attestation Identity Key）でそのデータに署名することにより、データを立証する。完全な構成のダイジェストが送信されるため、ベリファイアはマシンの構成を知ることができる（システムの動的要因を考慮すれば常時）。従って、送信される妥当性検査用のデータは、多用途で効率的な遠隔プラットフォーム妥当性検査を可能にする表現力を欠く。セマンティックアテステーションの必要性は早期から認識されており、１回のアテステーションの範囲を複雑度が制限された仮想化サブシステムに制約して、複雑で動的な高レベルプログラムプロパティを立証できるようにす

ることが提案された。

【０２５３】

プロパティおよびプロパティベースアテステーション（ＰＢＡ）が本明細書に記載される。ＰＢＡは、部分木証明書機関（ＳＣＡ）と呼ばれる信頼されるサードパーティ（ＴＴＰ）を介して、検証の済んだプラットフォームのセキュリティプロパティをベリファイアに保証することを可能にする。ＳＣＡは、プラットフォームの構成を、その構成で実現することが可能なプロパティ（特に望ましい／望ましくない機能）に対応付けた証明書を発行する。ＰＢＡは、ＴＣＧのプライバシーＣＡと同様に、しかしプライバシーＣＡの役割を拡張して、プラットフォーム妥当性検査のインフラストラクチャ上の問題をＳＣＡに移す。部分木の証明は、上記問題を克服する手段の１つである。これに関連する一実施形態は、更新証明書に基づいて別のプラットフォーム構成についてのデータを再シールするＴＰＭコマンドで行われる、ハードウェアで支援される更新である。

【０２５４】

検証データと異なり、妥当性検査データは、別のパーティ、すなわちバリデータに提出され、プラットフォームの状態の信頼性を評定するために使用される全データである。バリデータに妥当性検査データを提出するプロセスは、例えばＴＣＧに準じて遠隔アテステーションとして実現され、バリデータによるデータの評価は正式に妥当性検査と呼ばれる。妥当性検査データは、クオートされた検証データレジスタ（例えばＰＣＲ）の値等の検証データを含むことができる。妥当性検査は、暗号による検証データの検証以外に、ポリシーの評価およびバリデータによるアクションのトリガを含むことができる。

【０２５５】

ＳＭＬツリーに関連付けられたツリー型検証データおよび妥当性検査データは、プラットフォームアテステーションのセマンティックを強化するために上記の手法に加えて使用することができる構造化データを提供する。ツリー型検証データを使用する別の方法を提供して、ＰＢＡに関連する概念を実現する。

【０２５６】

ＳＣＡが、ＳＭＬツリーのノードを、有意で信頼されるステートメントに置き換える仕組みを示す。ステートメントは、そのノードが根である測定対象コンポーネントの部分木についての部分木証明書である。このプロセスは、ＳＣＡによるプラットフォームの部分的な妥当性検査を実現し、結果として信頼される表明が得られ、それがプラットフォームの利用可能な妥当性検査データに取り込まれる。それを後に別のバリデータに対して使用して、プラットフォームをより完全に妥当性検査することができる。次に、変形例を実現するために実装することが可能な部分木証明の汎用的な基本プロトコルとその使用事例を述べる。変形例も記載する。

【０２５７】

部分木の証明は、信頼されるプラットフォーム（これに限定されないが、簡略化されたシステムモデルではＴＰＭとＴＳＳの組合せ）がＳＣＡからツリー型ＳＭＬの内部ノードの値についての証明書を得るプロセスである。そのために、プラットフォームは、署名済みのステートメントをＳＣＡに提出して、そのノード値が、根の値が検証データレジスタで保護されたＳＭＬツリーに含まれていることを知らせる。ＴＰＭは可能な実装の１つであり、任意の検証データレジスタを使用することができる。その証拠に基づき、ＳＣＡは、そのノード値についての追加的な属性を含む証明書をプラットフォームに発行し、証明書がＳＭＬツリーに取り込まれる。このプロセスでは保護されたツリー操作を使用する。

【０２５８】

プラットフォームは、信頼されるプライバシーＣＡ（ＰＣＡ）から発行された証明書C_aと併せて、アクティブなＡＩＫａを所有することができる。プラットフォームとＳＣＡ間の通信は、暗号化され、完全性が保護されて、マン・イン・ザ・ミドル攻撃を緩和する。証明対象の内部ノードが選択される。このプロトコルでは、破綻条件と否定応答は言及されない。部分木証明プロトコルは、図２６に示すように少なくとも５段階で進行する。

【０２５９】

段階１で、ｓに対するクオートを作成する。そのために、ＴＳＳは、引数（＜ｓ＞，ｓ，Ｒ（ｓ），ａ）でTPM_Tree_Node_Quoteを呼び出し（図を簡潔にするために、図２６には呼び出し可能な全ての引数は示さない）、Ｐ＝Ｓｉｇ$_{a}$（ｓ）を受け取る。

【０２６０】

ツリーの根、すなわちレジスタＶが証明対象として選択された場合は、代わりにTPM_QuoteをＶに対して使用することになる。段階２で、ＴＳＳは、アテステーションパッケージＱを作成する。パッケージは、ＳＣＡを検証するための情報、少なくともＱ⊆｛Ｐ，ｓ，Ｃ$_{a}$，ａ$_{pub}$｝を含む。ａのパブリック部ａ$_{pub}$は、Ｃ$_{a}$から明らかでない場合にＱに含めることができる。また、ｓの値はＳＣＡに既知である場合があり、その場合はＱから省略してよい。

【０２６１】

より多くの情報、例えばクオートの一部である場合にはノード座標＜ｓ＞を含めることができる。ＱがＳＣＡに送信される（ＳＣＡの公開暗号鍵で暗号化され、完全性が保護された状態で）。この段階は、ＴＣＧに規定される遠隔アテステーションと同様である。

【０２６２】

段階３は、ＳＣＡの行為からなる。初めに、ＳＣＡは、Ｐの署名を検証することにより、かつ／または証明書チェーンをＰＣＡのルート証明書までたどることにより、Ｑを検証することができる。ｓが、証明可能なノード値であると認識すると、ＳＣＡはｓのマニフェストＭ$_{s}$を作成する。このマニフェストは、根ｓがＳＭＬにある部分木の存在に関連付けられたプラットフォーム状態についての追加的な情報を含むことができ、例えば、タイムスタンプ、プラットフォームの機能、部分木の葉ノードの測定結果で表されるロード対象のコンポーネントから組み合わせられたモジュールの識別、および／または他のプラットフォームプロパティ等を含むことができる。マニフェストは、部分木証明により追加される妥当性検査データであり、ノード値ｓのセマンティック意味をバリデータに提供する。ＳＣＡはｓの証明書を作成することができる。この証明書Ｃｓは、ＭをＡＩＫ　ａにバインドすることにより、Ｍで表されるプロパティをプラットフォームにバインドする。これは、次の２つの方式で行うことができる。

【０２６３】

【数２３】

$$C_s = \begin{cases} \mathrm{Sig}_{SCA}(M_s \| P) & \text{ｓが公開される場合} \\ \mathrm{Sig}_{SCA}(M_s \| \mathrm{bind}(a)) & \text{ｓが隠蔽される場合} \end{cases}$$

【０２６４】

最初の場合、ＳＣＡは、マニフェストおよびＡＩＫ書名済みのノード値に署名し、ａとの間接的なバインドを確立する。そして、プラットフォームがノード値ｓを公開すると、Ｃｓとａのバインドを検証することができる。２番目の選択肢では、ＳＣＡに、ａのパブリック部、Ｃ$_{a}$、連続番号、Ｃ$_{a}$のフィンガープリント等、ａを一意に識別する何らかのデータｂｉｎｄ（ａ）に署名させることによって間接的にバインドが実現される。公開鍵インフラストラクチャのセマンティックではバインドにより、Ｃ$_{s}$は、Ｃ$_{a}$に関連付けられた属性証明書となる。最後に、ＳＣＡは、２番目の事例では少なくともＭ$_{s}$およびＣ$_{s}$、並びにｂｉｎｄ（ａ）を含むパッケージＲを作成し、プラットフォームに返す。

【０２６５】

段階４で、Ｒから導出されるあるデータでＳＭＬの更新を作成する。ＳＭＬの更新は、部分木証明書と、ツリー中の証明されたノードの位置＜ｓ＞とのバインド関連付けを生成することができる。

【０２６６】

これにより、プラットフォームは、Ｃ$_{s}$およびＭ$_{s}$によって立証されたプロパティがプラットフォームの構成に存在することをバリデータに表明することができる。表される部分木にＣ$_{s}$をバインドするＳＭＬ更新には各種方式が考えられ、それぞれ特定の使用事例に

対して異なる形で適する。

【０２６７】

新しいノードの集合Ｕ＝｛u_1，．．．，u_k｝（例えば値およびＳＭＬツリー中の位置）が下記のプロパティで作成される。初めに、集合は、ｓ以下の部分木が更新の対象となるように、Ｕ≦ｓを保持することができる。これを行うことができるのは、厳密にＵより下の古いＳＭＬツリーノードｎ≦Ｕ、すなわちｎ／∈Ｕは、更新により無効化され、更新後の根の検証データレジスタを基準としてはそれ以上検証できないためである。２番目に、Ｕは依存関係を持たない。すなわちｕ，ｕ’∈Ｕ：ｕ≦ｕ’である。

【０２６８】

無依存性は、Merkleハッシュツリーで実施される一方向（上方）への情報の流れで、Ｕによりツリーを更新する際の一貫性を保証するプロパティである。詳細には、この性質により、更新の結果が、Ｕの要素が処理される順序に依存しないようになる。

【０２６９】

段階５は、ＳＭＬツリー更新そのものである。ｕ∈Ｕに対して反復的にTPM_Tree_Node_Verified_Updateが引数（ｕ，ｎ，ｕ，Ｒ（ｎ））で呼び出される。ｎは、位置＜ｕ＞にある古いＳＭＬノード値である。この結果新しいトレースＴ（ｕ）が返され、ＴＳＳはそのトレースでＳＭＬを更新する。上記要領でツリーの更新を実行すると、ＳＭＬおよび根の検証データレジスタに一貫したセキュリティレベルが維持される。すなわち、動作◇はＴＰＭの内部で実行される。Ｕが多くの要素を含んでいる場合は、段階５に関して説明したように更新を行うと効率的でない可能性がある。これは、そのような場合、TPM_Tree_Node_Verified_Updateは多くの重複する縮小ツリーを検証することになり、そのため（複雑な）ハッシュ計算に冗長性をもたらすためである。「効率的でセキュアなノードセットの更新（Efficient Secure Node Set Update）」と呼ばれるより効率的な更新アルゴリズムを下記で説明する。

【０２７０】

上述のように、内部ノードの大きな集合Ｕを更新することは、TPM_Tree_Node_Verified_Updateを使用すると非効率的である場合がある。これは、Ｕの要素からなる縮小ツリーの重複によっては、検証のための冗長なハッシュ演算が多数行われる可能性があるためである。上記のTPM_Tree_Extendコマンドを使用して、大量更新の方式を適用して部分木証明プロトコルの段階５の単純なアルゴリズムを改良することができる。この方式は、更新の集合Ｕの部分集合の範囲は、古いＳＭＬ値に依存しない部分木である、すなわち、それら部分木の根がＵにあるノードに依存するという観察に基づく。従って、そのような集合の範囲となるツリーの根は、高費用の検証を行わずに事前に計算することができる。

【０２７１】

いくつかの定義を提供する。Ｕ＝｛u_1，．．．．．，u_k｝が、依存関係のない更新の集合であるとする。ＳＭＬツリー中のノードは、ａ）それがＵの要素であるか、ｂ）そのシブリングがＵにあるか、またはｃ）そのシブリングがＵに固有である場合に、Ｕに固有と呼ぶ。この再帰的な定義では、更新後の値がＵに依存し、Ｕの補集合にあるＳＭＬノードに依存しないノードを取り出す。部分集合Ｖ⊆Ｕの範囲となる根は、Ｖの全要素のトレースの一意の交差点となる。部分集合Ｖ⊆Ｕの範囲となる部分木は、Ｖの要素のトレースと、省略された範囲となる根より厳密に上にあるノードとの和集合になる。部分集合Ｖは、それが範囲とする部分木の全要素がＵに固有である時かつその時に限りＵに固有と呼ばれる。

【０２７２】

これらの設定で、ＵによるＳＭＬのより効率的な更新を次のように行う。

１）（互いに素の）Ｕに固有の部分集合V_1，．．．．．V_k⊆Ｕを特定する

２）V_i，ｉ＝１，．．．ｋに以下を繰り返す。

ａ．V_iの要素の座標を以下により正規化する。

ｉ）V_iの範囲となる根の座標で与えられるプレフィクスを切り捨て、

ｉｉ）全ての座標が等しい長さ、すなわちV_iの範囲である部分木の深さになるま

でゼロを後置する。
ｂ）V_iの要素を各自の正規化された座標に従ってアルファベット順に並べて、順序付けされたリスト
【０２７３】
【数２４】
$\tilde{V}_i$

【０２７４】
を生成する。
ｃ）
【０２７５】
【数２５】
$\tilde{V}_i$

【０２７６】
にある全ての空白箇所（正規化後の座標中）をニル値で埋める。
ｄ）空いている検証データレジスタＶ’を選択する。
ｅ）対象Ｖ’を用いて
【０２７７】
【数２６】
$\tilde{V}_i$

【０２７８】
の要素にTPM_Tree_Extendを順次使用する。
ｆ）ＵからV_iを除去する。
ｇ）Ｕに（Ｖ’，＜v_i＞）を挿入する（v_iはV_iの範囲となる根）。
３）Ｕの残りの要素に対してTPM_Tree_Node_Verified_Updateを使用して上記のように通常の更新手順を適用する。
【０２７９】
部分木証明プロトコルの変形では、１回のプロトコルの実行時にＡＩＫの部分木証明のためのＰＣＡとＳＣＡの役割を組み合わせることができる。この利点は、ＡＩＫの生成、有効化、および使用が１回のセッションに結合されるため、ＡＩＫ証明書C_aの明示的な生成および検証が必要でなくなる可能性があることである。このプロトコルの組合せは単純である。
【０２８０】
受け取った部分木証明書をプラットフォーム状態にバインドすることは、ツリー型ＳＭＬの正しい構成、すなわち証明された部分木の根の位置に証明書をバインドすることを意味する。上記のように、これは、プラットフォーム構成全体のコンテクストにおける有意味の部分木証明書を利用する妥当性検査には必須である。C_sをＳＭＬツリーにバインドする具体的な目標の１つは完全性の保護である。これは、例えば後に別の証明書に置き換えられることを防ぐことができるためである。バインドは、部分木証明書を一意かつ検証可能に識別するデータでツリーの一部を更新することによって実現することができる。部分木証明書から幅広い範囲のデータ項目を生成し、ＳＭＬツリーの様々な位置に入れることができる。
【０２８１】
一例では、ＳＭＬの更新が些少で、Ｕが空である可能性がある。この可能性があるのは、ｓを公開する上記第１の選択肢でａが作成される場合である。そして、ｓをＳＭＬツリーに保持することができ、それ以下の部分木も保持されるかどうかは使用事例による。バインドによる関連付けは、Ｃで署名された実際のノード値ｓを介する。
【０２８２】

別の例として、証明書で立証されたプラットフォームプロパティに関するデータを、例えば鑑定用の使用のために更新後のツリーで保護できる事例を考える。すなわち、３つのデータ項目、ｓ、M_s、およびC_sを更新の集合に入れることができる。ノード値ｓはすでに正しいデータ形式であるが、他の２つは初めに処理してｍ（M_s））およびｍ（Ｃ（ｓ））とする。演算ｍは、プラットフォームのＲＴＭ（Root of Trust for Measurement）または他の適切な一方向演算によるハッシュ値の生成とすることができる。何らかのデータ項目が、適切なノード値形式の一意に識別する適切なデータをすでに含んでいる場合は、そのデータを直接抽出し、ノード更新値として使用することができる。具体例は、Ｃ（ｓ）に含まれる証明書フィンガープリントである。そして３つの更新ノードを更新集合として構成して、例えば図２７に示すようなＳＭＬツリーの更新後のノードの構成を生成することができる。

【０２８３】

更新された部分木の根はｓが元にあった位置に挿入され、値ｋ＝（ｍ（C_s）◇ｍ（M_s））◇ｓを有する。この構成は、部分木証明書およびマニフェストに独立した完全性保護を提供し、古いノード値を独立して保持する。C_sで表されるプラットフォームプロパティのアテステーションは、この構成でも、部分木の左内部ノード*だけをクオートすることにより、ｓについての情報を明かすことなく行うことができる。

【０２８４】

証明書と部分木のバインドの変形は各種ある。プラットフォームは、例えばセキュアなクロックによる内部タイムスタンプ等、自身で生成したデータ（の完全性保護値）を含めたい場合もある。この部分の実装は使用事例に固有であってよい。

【０２８５】

部分木証明書で表されるプロパティをバリデータに対して立証するために、プラットフォームは、TPM_Tree_Node_Quoteを使用して、少なくともマニフェストおよび証明書自体を含む対象妥当性検査データを保護する、更新後の部分木の中のノードまたはそれより上にあるノードをクオートすることができる。そして、プラットフォームは、必要に応じて妥当性検査データ、少なくとも表明されるプロパティの検証に必要な全てのデータ（この場合も少なくともM_sおよびC_sを含む）をバリデータに提出する。提出されたクオートですでに保護されている妥当性検査データは、基本的には、それ以上の完全性保護は必要としない。

【０２８６】

バリデータは、プラットフォームによる妥当性検査データのバインドを検証することができる。このプロパティを証明すること、すなわち妥当性検査を行うプラットフォームが、ＳＣＡに対して部分木証明を行ったプラットフォームと同じであることを証明することは、容易でない可能性がある。これを実現する方式の１つは、証明時と同じＡＩＫを部分木妥当性検査で使用するものである。その場合、プラットフォームはa_{pub}も提出し、必要な場合はC_aも妥当性検査データの一部として提出することができる。C_aが必要とされるかどうかは部分木証明書のセマンティックに依存する。すなわちＳＣＡはすでにＡＩＫ証明書を調べている可能性があり、C_sがその真実性を表明する可能性がある。対応する情報をマニフェストに入れることができる。同じＡＩＫを再使用すると部分的にプライバシーを損ない、この問題を解決する他の方法も検討することができる。

【０２８７】

部分木証明を使用するためのステップの１つは、プラットフォームが特定のＳＣＡから証明書を取得することが可能な部分木を見つけることである。プラットフォーム、ＳＣＡ、およびバリデータ間の対話の詳細には立ち入らずに、部分木発見方法の２つのカテゴリを以下で説明する。第１のカテゴリは、部分木発見の負担をプラットフォームに課すものであり、第２のカテゴリでは「ダム（dumb）」プラットフォームを想定し、ＳＣＡにより高い負荷を課す。

【０２８８】

一例では、ＳＣＡがいくらかの部分木発見データをプラットフォームに送信する。最も

単純な事例では、プラットフォームが直ちに証明できるノード値のリストを送信する。プラットフォームは、自身のＳＭＬツリーからそれらの値を探索し、特定された各ノードに部分木証明を行うことができる。この基礎となる手順は、各種の改良、特に発見用データを拡充することと発見手順をプラットフォームとＳＣＡ間のネゴシエーションプロトコルに拡張することを提案する。例えば、発見用データは、証明可能な根に対する条件として根ノードの位置を含むことができる。これは、認証ブートプロセスで生成されたＳＭＬの場合、後者の部分木の構築時にロードされるコンポーネントは、プラットフォーム起動の定義された段階でロードされることに相当する。

【０２８９】

ノードの絶対的な配置に対するそのような条件は、構成が動的に変化しうる複雑なプラットフォームの場合は実際には困難である可能性がある。従って、より精緻な条件では、いくつかの、例えば証明可能な根ノードの対の相対位置を表してもよい。ＳＣは、例えば「ｓは、ｒより後にある場合に証明可能である」（すなわちｒは、順序付けされたＳＭＬツリー中でｓの左に位置することができる）ことを知らせる表現等の表現を表明することができる。これは、別の機能が特定の機能前に動作可能になった場合は、該特定の機能がプラットフォーム上で動作可能になることを目的として解釈することができる。

【０２９０】

このモデルのより本質的に異なる変形は、発見データが部分木の根すなわち内部ノードから構成されるのではなく、葉、すなわち測定結果の値からなるものである。「ボトムアップ」式の発見手順では、信頼される値と認識していることをＳＣＡが表明している受け取った葉の測定値に基づいて、プラットフォームが、どの内部ノードが証明可能であるかについての「知識に基づく推測」を行うことが必要とされる場合がある。１つの方法は、全ての葉が発見データ中にある部分木の範囲となる根の集合を見つけるものである。そして、プラットフォームは、部分木の根をクオートし、ＳＭＬ部分木と共にＳＣＡに送ることができる。一般に、これはなお葉の順序に依存する可能性があるため、ＳＣＡはＳＭＬ部分木を検証し、その根を証明できるかどうかを判定しなければならない場合がある。場合によっては、プラットフォームは、ＳＣＡが一部の葉の値を知っている部分木の証明書の取得を求める場合があり、すなわち、発見用データを基準とした時に、対応する部分木の葉の集合に空白箇所がある場合がある。プラットフォームが他の信頼されるデータ、例えばＳＣＡが信頼するパーティから取得したＲＩＭ証明書を有する場合、プラットフォームは、それらのデータを部分木証明の段階２で提出して、ＳＣＡが部分木の根を証明するための判定を行うのを助けることができる。

【０２９１】

機器が部分木のローカルな発見を行うことができない場合は、計算をＳＣＡに移す実装を使用することができる。プラットフォームは、ｎより下の適切な部分木を探すためにＳＣＡから証明書を取得することを目的として、内部ノードｎを選択する。プラットフォームが証明可能な全てのノードを証明することを要求する場合は、ノードｎは、完全なツリーの根（Ｖ）に等しい可能性がある。続く２つのステップは上記で段階１および２で説明したのと同様であり、すなわちＶが選択された場合はプラットフォームがそれぞれTPM_Tree_Node_QuoteまたはTPM_Quoteを行う。アテステーションパッケージは、クオートされた根より下のＳＭＬ部分木と共にラップされる。ＳＣＡはこの情報を受け取り、ツリートラバース技法を使用してツリーの完全性を検証し、同時に、証明可能な根の集合Ｓを持つ１つまたは複数の（互いに素の）部分木S_iを見つけることができる。

【０２９２】

ＳＣＡは次いで上記のようにプロトコルの段階３を繰り返すことにより、全ての$s_i \in S$に証明書を作成する。このプロトコルでは、更新ノードリストＵに取り込まれた複数ノードを更新することができるので、見つかった全ての部分木の更新を１回のプロトコルの実行で行うことができる。変形例の１つは、プラットフォームが第１のステップでTPM_Tree_Node_QuoteまたはTPM_Quoteを送信せずに、選択されたノードｎから開始するＳＭＬをＳＣＡに提供するものである。そして、ＳＣＡは、証明する潜在的な候補部分木を探し、

特定された部分木の根ノードにTPM_Tree_Node_Quoteを提供するようにプラットフォームに要求する。これは、プラットフォームが事前に暗号操作を行わずにＳＭＬを送信できるようにする点でトレードオフである。それでも、プラットフォームは、ＳＣＡによる証明の前に適切なクオートを提供して、送信するＳＭＬの完全性保護を提供してよい。

【０２９３】

上記では、ＴＦＶを実装するシステム、方法および装置を説明した。以下では、ＴＦＶの変形および／または拡張を説明する。

【０２９４】

バリデータがプラットフォームから提出されたツリー型ＳＭＬをトラバースするための方法は、上記のようにレベル単位で下降していくことができる。以下で説明する変形手順では、バリデータが初めにＳＭＬツリー中の部分木の根の既知のｇｏｏｄな値を探して、妥当性検査済みのプラットフォームの既知のｇｏｏｄな部分を特定し、その後不合格の葉コンポーネントを求めてＳＭＬツリーをトラバースする。

【０２９５】

プラットフォーム全体のＳＭＬツリーの参照ツリーを保持することは難しい場合がある。そのようなツリーは大きく、プラットフォーム構成、例えばコンポーネントのロード順序に大きく依存する場合がある。ツリーのトラバースによる妥当性検査では、シーケンスに記述された診断妥当性検査が、妥当性検査されるプラットフォームの所望の基準構成に準拠しないコンポーネントを特定すると判断する際に、静的な測定結果シーケンスを想定する。実際、Merkle-Damgard変換は可換性でも結合性でもないので（マルチセットのハッシュ関数ではない）、ツリーの根の値は正確な測定結果シーケンスに依存する。これと同じことが線形に生成されたＰＣＲ値に当てはまるため、上記の性能の比較はなお理にかなう。ツリーの根が順序の影響を受けることの問題についてはさらに下記で取り上げる。下記に記載される方法は、ここで説明する方法と組み合わせることができる。

【０２９６】

単純な解決法では、１つのプラットフォームの（機能面およびセキュリティ面で）互いに実質的に等価な構成についての参照ツリーを維持することを状況が必要とする可能性がある。従って、完全な参照ツリーよりも細粒度の妥当性検査用データセットが望ましい場合がある。

【０２９７】

ＳＭＬツリーおよびそれに対応する参照ツリーの根の既知のｇｏｏｄな値のデータベースを保持する代わりに、バリデータは、様々な深さの部分木の既知のｇｏｏｄな部分木の根の値の拡張データベースを保持する。それらの値は対（ｒ，ｄ）として記憶され、ｒは部分木の根のノード値であり、ｈは部分木の深さである。深さｄの範囲は、完全なＳＭＬツリーの深さＤから０であり、ｄ＝０は、対応する既知のｇｏｏｄな参照値が実際に葉の測定結果であることを意味する。診断ツリートラバースにおいてあるレベルＬに達すると、暗号の検証後に、バリデータは、幅優先探索により、受け取ったＳＭＬツリーのそのレベルにあるノード値と、既知のｇｏｏｄな値｛（ｒ，ｄ）｜ｄ＝Ｄ－Ｌ｝とを比較する。一致する根を持つ部分木は「既知」とマークされ、その後の診断トラバースから除外され、診断トラバースはレベルＬ＋１に進む。

【０２９８】

既知の部分木の根のデータベースを保持することにより、バリデータにとっては、ツリーを根から下降していくのと反対の異なる妥当性検査方式が可能になる。バリデータは、効率的と思われる探索手順により、ＳＭＬツリー中の既知のｇｏｏｄな部分木の根の値（の部分集合）を探す。そして、バリデータは、それらの部分木の根の完全性を検証することができる。このために、バリデータは、部分木の根ノードの縮小ツリー、および他箇所で説明するその値のための検証手順を使用することができる。値の完全性が検証された場合は、対応する部分木を「ＯＫ」とマークし、以降の診断妥当性検査から省略することができる。

【０２９９】

既知の部分木をそのような既知として特定すると、異なるプラットフォーム構成、特に既知のコンポーネントと不合格コンポーネントの異なるロード順序の余地を残す。例えば、それぞれ８個の測定結果の葉を有する２つの部分木ＡおよびＢがあり、それらの間に不合格コンポーネントを含む別の部分木を有するとする。バリデータがＡおよびＢを既知と識別するが、通常であればＢ、Ａの順および直接の順序で受け入れ、または別の既知の部分木ＣがＡ、Ｃ、Ｂの順で真ん中にある場合、バリデータは、そのプラットフォーム構成を有効として受け入れるかどうかをポリシーに基づいて決定しなければならない。

【０３００】

上記変形の拡張例として、バリデータは、以前のプラットフォーム妥当性検査の結果に基づく対話型の学習により、参照ツリーデータベースに動的にデータを入れることができる。可能性の１つは、プラットフォームが初めて妥当性検査する時に診断妥当性検査を行い、本明細書に示すように、バリデータのコンポーネント（すなわち葉）、信頼される参照値およびポリシーに従って、ｇｏｏｄなプラットフォーム構成に相当する葉の測定結果の部分列から既知のｇｏｏｄな部分木を生成するものである。妥当性検査を行うプラットフォームが増えると、バリデータは、（構成に関連する）共通点やリスク等の基準に従って、既知のｇｏｏｄな部分木とプラットフォーム構成の統計的に重み付けしたデータベースを構築することができる。例えば、頻繁に出現する部分木を最初に探すことができる。

【０３０１】

上記の方法は、部分木証明の方法に関して説明した部分木発見方法に似る。これは、学習方式と共に、または別個に、ＳＣＡエンティティによって適用できる方法である。

【０３０２】

バリデータがプラットフォームから提出されたツリー型ＳＭＬをトラバースする方法は、レベル単位で下降していくものである。そのために、ツリー全体を１回の通信ステップでバリデータに提出し、参照ツリーと比較して評価する。ここで、様々なレベルのツリーノードを１つずつ順次送信する代替実施形態を説明する。

【０３０３】

ＳＭＬツリー全体をバリデータに提出することができる。これは一般には最適ではなく、特に不合格の葉の測定結果の比率が低い場合には最適でない可能性がある。その場合、ＳＭＬツリー全体の一部として多数の部分木が送信されることになるが、部分木の根が参照ツリーの対応するノードに関して正しいため、それら部分木に含まれるノードデータは妥当性検査に使用されない可能性がある。

【０３０４】

ＴＦＶで送信データを最小にするための基本手法は、他の箇所で述べたように、ツリーのトラバースによる妥当性検査の現在のステップで実際に必要なデータだけをバリデータに送信するものである。バリデータはまずＳＭＬツリーの根の検証データレジスタを受け取り、双方向型のツリートラバースによる妥当性検査のレベルＬ＝０としてそのレジスタを評価する。レジスタの値が参照ツリーの根と一致する場合、妥当性検査は終了する。その他の場合は、バリデータは、根の２つの子ノードをプラットフォームに要求する。子ノードを受け取ると、バリデータは、本明細書に記載される図９の完全性の破綻条件ｃ）を検出して終了するか、または子ノードから根を再計算する。子ノードを含むメッセージは、拡張動作の再計算によってその内容が親ノードに対して検証されるので、原則としては完全性についても新しさについても暗号による保護を必要としない。そして、根は、TPM_Quoteまたは同様の操作、例えばデジタル署名で保護して送信されるものと想定された。

【０３０５】

バリデータは、Ｌ＝１の２つの子を比較して、２つのうちどちらが参照ツリーに対して正しいかを判断することができる。２つの子は以降の探索から除かれる。そして、バリデータは、ツリーの深さ１にある全ての「ｂａｄ」なノードの子を要求する（すなわち、ツリー深さ２にある２つまたは４つの子ノード）。それぞれの子についての検証および参照ツリーとの比較は前述のように進行する。この手順は反復により継続する。このようにして、不合格の葉測定結果を見つけるのに必要とされるｂａｄな内部ノードが実際にバリデ

ータに転送される。上記のように、これは、短いメッセージサイズで暗号化の過負荷なしに非常に効率的に行うことができる。ツリー型ＳＭＬの大きさに対するデータ量は、不合格の葉の割合に応じて決まる。これを本明細書に説明される図１０に示す。

【０３０６】

ＴＦＶの適合性のためのツリー型検証データと他の方法との各種組合せがある。１番目は、頻繁に変化する、動的なプラットフォームの（部分的な）状態のために妥当性検査データを提供するものであり、２番目は検証データのための拡張可能な空間を提供するものである。

【０３０７】

ＰＣＲすなわち検証データレジスタは、わずか１つの（ハードウェアで）保護されたメモリ空間であり、よく知られるようにプラットフォームには検証データレジスタが不足する。本明細書に記載されるように、ハッシュツリーは基本的にこの制約を解消する。しかし、根が保護された検証データレジスタ、例えばＰＣＲを用いてツリー型ＳＭＬのセキュアな生成プロセスを行うと、固定された深さのツリーとなり、葉の測定結果のための容量が制限される可能性がある。プラットフォームが、自身のツリー型ＳＭＬに収まる以上の測定対象コンポーネントを有する場合には問題が生じる可能性がある。

【０３０８】

検証データが、メモリ領域等頻繁に変化する部分的なプラットフォーム状態の記録を保護している場合は、別の問題が生じる可能性がある。ＰＣＲ値を更新するハードウェア保護された動作であるTPM_Extend動作は時間がかかる。そのため、ＰＣＲは、頻繁に変化するデータの保護には適さない。従って、ＰＣＲは、直接的な実行時メモリ保護には適さない場合がある。各種システムではこの目的のためにMerkleハッシュツリーを用いるが、その根は通例はハードウェア保護されたレジスタには記憶されない。これは、Merkleハッシュツリーの更新は非常に時間がかかり、プラットフォームの動的な変化に追随していくことができないためである。

【０３０９】

（離散）時間ｔに依存する何らかのデータＷ（ｔ）で表される動的に変化するプラットフォーム状態を保護する際に用いられる基本的な考え方は、関連する検証データレジスタＲに対する「キャッシュしてから更新」の方式である。何らかの信頼されるソフトウェアＡＰＰ＿ＲがＲでこの状態シーケンスを保護する。ＡＰＰ＿Ｒは、長さＬの状態の区間Ｗ（ｋＬ＋１），．．．，Ｗ（（ｋ＋１Ｌ）を集め、それらをキャッシュしてからダイジェスト関数で測定し、ＰＣＲの拡張動作または例えばMerkle-Damgardを利用した同様のチェーン化動作を使用して、拡張してＲに入れる。

【０３１０】

ＡＰＰ＿Ｒには、様々な高いセキュリティ要件が該当する。特に、ＡＰＰ＿Ｒには変更および改ざんを防ぐ十分な保護が適用されなければならない（例えば仮想化やセキュリティカーネルとの統合等の方法が知られる）。さらに、ＡＰＰ＿Ｒの動作は、外部のバリデータによる評定に公開される場合がある。そのための最低要件は、ＡＰＰ＿Ｒの信頼性を例えばセキュアな起動時に検証し、ロード前にＡＰＰ＿Ｒを測定することによって記録し、バリデータにより検証できるように、Ｒで保護される実際のシーケンスＷにバインドすることである。

【０３１１】

望ましい機能を提供する方法は以下の通りである。シーケンスＷ（ｔ）を長さＬの区間に切り分けることができる。

【０３１２】

【数２７】

$$\underbrace{W(L(k-1)+1),\ldots,W(Lk)}_{\text{長さ}L},\ \underbrace{W(Lk+1),\ \ldots,W(L(k+1))}_{\text{長さ}L},\ldots$$

【０３１３】

そして、部分シーケンスにダイジェスト関数を適用して区間の測定値M_k＝ｍ（Ｗ（Ｌ（ｋ－１）＋１）｜｜．．．｜｜Ｗ（Ｌｋ）を生成する。変形の１つとして、例えば下記のように、区間の測定時にセキュアな提供元から入手されるタイムスタンプを取り入れると有用である場合がある。

M_k＝ｍ（Ｗ（Ｌ（ｋ－１）＋１）｜｜．．．｜｜Ｗ（Ｌｋ）｜｜ＴＰＭ＿Ｔｉｃｋ（ｎｏｗ）

【０３１４】

測定値の区間にタイムスタンプを付加すると、コンポーネントのロード順序に関する情報を失うことなく、（下位）構造の再順序付けが可能になる。さらに測定結果の「経時変化」も可能になり、バリデータは、所与の期間後に測定値の更新を要求することができる。区間の測定値は、例えば拡張動作Ｒ（ｋ＋１）＝Ｒ（ｋ）◇M_{k+1}で測定値を拡張して検証データレジスタに入れることによって保護され、その結果、通常の線形チェーンのＳＭＬと同様のシーケンスが得られる。

→Ｒ（ｋ）→Ｒ（ｋ＋１）
↑ ↑
M_k M_{k+1}

この場合、中間状態もＳＭＬで保護される。すなわち、Ｗを保護するＳＭＬは次のようになる。

$SML_W \subseteq [R_{init}, R(0), M_1, W(1), \ldots, W(L), R(L), \ldots]$

【０３１５】

この管理アプリケーションＡＰＰ＿Ｒの信頼される状態への望ましいバインドは次のように実現することができる。プラットフォームのセキュアブートまたは認証ブート機能により、ロード前にＡＰＰ＿Ｒを測定し、測定値ｍ（ＡＰＰ＿Ｒ）を拡張して検証データレジスタＱ（推移する信頼チェーン中の先行するＰＣＲ　Ｐの値で初期化してもよい）に入れる。そして、実際の記録レジスタＲのシーケンスを、Ｐ、および初期化値Ｒ＿ｉｎｉｔ、例えばセッションノンス若しくはタイムスタンプ、またはその両方で初期化する。その結果次の構造が得られる。

Ｐ → Ｑ → Ｒ＝Ｒ（０）
↑ ↑
ｍ（APP_R） R_{init}

【０３１６】

検証データレジスタの不足がある場合は、Ｐ、Ｑ、およびＲは１にしてよい。検証は、Ｗについて拡張されたＳＭＬに対して進行し、ＳＭＬは

［（Ｗ（Ｌｋ＋１），．．．，Ｗ（Ｌ（ｋ＋１））），Ｍｋ＋１，Ｒ（ｋ＋１），Ｒ（ｋ），．．．，／Ｒ（０），R_{init}，Ｑ，ｍ（APP_R），Ｐ］

およびＰ、Ｑ、Ｒのクオートを含むことができる。存在する場合は、ＳＭＬは、タイムスタンプの値も含むことができる。

【０３１７】

実際のハードウェア保護された検証データレジスタは、ツリー型ＳＭＬの根から切り離すことができ、ＳＭＬの根はその後、低保護のメモリ空間に記憶することができるが、検証データレジスタで保護される。このために、特殊なソフトウェアＴＦＶ＿ＡＰＰが根Ｗで保護されるハッシュツリーを管理し、根Ｗはなお通常のメモリ空間の深さｄ、葉Ｖ＿１，．．．，Ｖ＿（２∧ｄ－１）の内部ツリーにあってよい。そして、例えば１つ前の項で説明した区間方法を使用して、Ｗ（ｔ）の状態シーケンスがハードウェア保護された実際の検証レジスタＲで保護される。そして仮想レジスタＶをＳＭＬツリーの根として使用することができ、ＴＦＶ＿ＡＰＰで管理される。その仮想レジスタについて、ＴＦＶ＿ＡＰＰは、仮想レジスタが根の検証データレジスタであるかのように、他箇所で説明したコマンドを外部アプリケーションに公開する。Ｒの構造は先と同様に

→R_{N+1}（ｔ－１）→　R_{N+1}（ｔ）

↑ ↑
W（ｔ－１） W（ｔ）
となり、結合されたWは、図２９に示すようにツリーの根を表す。
【０３１８】
V_i（ｔ）の検証は、以下の検証データに拡張することができる。
V_i（ｔ），R（V_i（ｔ）），W（ｔ），R_{N+1}（ｔ），R_{N+1}（ｔ－１），R_N，R_{N-1}，m（ＴＦＶ＿ＡＰＰ），Q（｛R_{N-1}，R_N，R_{N+1}｝）。Qは、ＰＣＲ、R_{N-1}，R_N，R_{N+1}のクオートであり、以下に従って進行する。
ｉ．縮小ツリーR（V_i（ｔ））、および根W（ｔ）に関してV_i（ｔ）を検査する
ｉｉ．（２）によりR_{N+1}（ｔ）＝R_{N+1}（ｔ－ｌ）◇W（ｔ）を検査する
ｉｉｉ．追加的にR_{N+1}（０）、W（０）が送信された場合は、R_{N+1}（０）＝R_N◇W（０）を検査する
ｉｖ．R_N＝R_{N-1}◇m（ＴＦＶ＿ＡＰＰ）を検査する
ｖ．参照値と比較してm（ＴＦＶ＿ＡＰＰ）を検査する
ｖｉ．クオートQを検査する。
【０３１９】
追加的に、上記のようにタイムスタンプを適用してもよい。ＴＦＶ＿ＡＰＰのバインドは、
R_{N-1}→ R_N → R_{N+1}（ｔ＝０）
↑ ↑
m（ＴＦＶ＿ＡＰＰ） W（０）
と同様に根R_{N+1}に行われるものとし、W（０）は任意の初期化値である。
【０３２０】
ＳＭＬツリーのセキュアな形成には、当然の容量の制限がある。この制限を解消する一方法は、上記の部分木証明を簡略化したプロファイルで適用するものである。ある固定の深さのＳＭＬツリーが完全に埋められ、プラットフォームがツリーを（自己）証明し、得られた証明書を最初の測定として次の空のＳＭＬツリーに挿入する。その結果、図２８に示すような左が非平衡の多ツリー構造となる。
【０３２１】
レジスタＶは、ツリー型ＳＭＬであるＳＭＬ＿１の根を保護することができる。次のツリーにセキュアに続けるための独立した方式は以下の通りである。プラットフォームは、TPM_Quote(V,a_1)を呼び出して、ＡＩＫ　ａ＿１を使用して署名されたステートメントＱ＿１を得る。TPM_Quoteは、指定された検証データレジスタＶに対してＴＰのＴｒＥの内部で（例えばＴＰＭにより）実行することができる署名コマンドを表す。そのため、Ｑ＿１は、ａ１の署名およびＰＣＡ証明書が検証されると、値Ｖがかつて検証データレジスタＶに保持されていたことをベリファイアに表明する。プラットフォームは、増強したＳＭＬ＿１、SML_1^*：＝ＳＭＬ＿１｜｜Ｖ｜｜Ｑ＿１｜｜Ｃｅｒｔ＿ＰＣＡ（ａ＿１）すなわち、元のＳＭＬ＿１、その根の値、クオート、および対応するＡＩＫ証明書を後の検証のために保存する。プラットフォームはＶをリセットする。上記のプロパティにより、$SML1^*$は、ＳＭＬ１およびその根を保持しているレジスタＶと意味的に等価となる。この時点で、レジスタＶは、次のツリー構築のために解放されている。
【０３２２】
後続のツリーを先行するツリーにバインドすることで、統轄しているＲｏＴ（root of trust）による内部検証の継続、すなわち保持されている葉の測定結果に対する継続的な完全性の保証を保証することができる。このための例示的方法の１つは、１つ前のツリーにバインドした最初の測定値で次のツリーＳＭＬ＿２を開始するものである（例えばプラットフォームのＲＴＭを変更することによる）。これを行う各種例については下記で説明する。
【０３２３】
例えば、最後の根の値Ｖを最初の測定結果として直接使用することができる。すると次

のツリーは１つ前のツリーに適用された証明のセマンティックについての手がかりを一切含まないため、これは事象の順序を保存することができない。別の例によると、測定結果ｍ（Ｑ＿１）をＳＭＬ＿２の最初の葉として使用することができる。別の例によると、ｍ（Ｖ｜｜（Ｑ＿１）｜｜Ｃｅｒｔ＿ＰＣＡ（ａ＿１））をＳＭＬ＿２の最初の葉として使用することができる。別の例によると、ｍ（ＳＭＬ＿１*）をＳＭＬ２の最初の葉として使用することができる。

【０３２４】

最後の３つの選択肢は意味的に等価であるが、検証および／または生成の複雑度の点で実際的な重要度が異なる。証明されたツリー型ＳＭＬの連続は、制限なしに反復的に継続することができる。証明による内部での連続に代えて、信頼される第３者に要請してＳＭＬ＿１ツリーを証明させてもよい。

【０３２５】

ツリーの証明された連続は、要請されたＴＴＰにより、追加的なセマンティックを加えて拡張することができる。詳細には、根を証明する要求と共にＳＭＬ＿１が提出され、検証された場合は、すなわち根の再計算により整合性が検査された場合は、ＳＭＬ＿１のハッシュ値を、ＴＴＰから得られる証明書に直接埋め込むことができる。これは、後に、ＳＭＬ＿１がＴＴＰによる証明のために提出された時に改ざんされなかったという保証をバリデータに提供する。これにより、ＳＭＬ＿１の根の値は原則として古くなる。バリデータは、ＳＭＬ＿１のブロブに対するグローバルハッシュおよびそのハッシュに対するＴＴＰ証明書の署名を検証して、ＳＭＬ＿１の完全性を検査する必要がある。これは有利であるのは、ツリートラバースによる妥当性検査のためにＳＭＬに参照ツリーを使用することを望むバリデータにとって、内部ノードのハッシュ演算の再計算が必要でなく、参照ツリーノードとの直接の比較を使用できることを意味するためである。これにより、不合格コンポーネントの探索を迅速化することができる。さらなる変形として、ＴＰＭティックスタンプ（tick-stamp）またはＴＴＰタイムスタンプを証明に追加してもよい。

【０３２６】

より表現力のある妥当性検査データをツリー型ＳＭＬの検証データ中にバインドするいくつかの方法および選択肢を以下で説明する。

【０３２７】

任意構造のＳＭＬの表現性は、コンポーネントコードおよびデータの測定結果そのものに自ずと制限される。一般に、これは、妥当性検査を効率的に処理し、妥当性検査の結果に従ってプラットフォームを効果的に管理するのに十分な情報をバリデータに与えない。この理由から、ＴＣＧは、ＳＭＬの概念を拡大して、コンポーネント識別子、ロード時間、パラメータ、および関連する情報も記録するイベントログとし、ＴＣＧ　ＩＷＧは、認証ブートで収集される関連情報を取り込むための複雑で表現力のあるデータ構造である（実際にはツリー構造）ＩＭＬの概念を導入している。

【０３２８】

しかし、そのような拡充され、構造化された妥当性検査データは、ツリー型ＳＭＬ等の適切な関連構造を検証データにバインドすることを欠く可能性がある。特にツリー型ＳＭＬの場合には、コンポーネントおよびプラットフォームプロパティに関するコンテクスト情報、または他の有用な妥当性検査データをどのようにして実際の検証データからなるハッシュツリーにセキュアにバインドするかという問題が生じる。

【０３２９】

実際の測定プロセス、すなわちプラットフォームの起動段階では、妥当性検査データにメタデータを追加し、基本的には測定対象コンポーネントについて小さな部分木を確立することにより、メタデータをＳＭＬツリーにバインドする。例示的な実現（ただし多数のコンポーネントとその測定結果の葉に関する）は上記に記載した。ＴＦＶの文脈における測定プロセスを以下のように拡張して、ＳＭＬツリーにメタデータを含めることができる。

１．測定およびＳＭＬの構築を行うエージェント（ＲＴＭと同一、またはＲＴＭと協働す

る）がコンポーネント測定マニフェストをロードする。マニフェストは、対象コンポーネントについて収集し、ＳＭＬツリーに含めるべきデータについての指示を含んでいる。
２．ＳＭＬツリー内で次の葉が右のシブリング（その２進座標の最下位桁がＴ）である場合、エージェントは、ＳＭＬツリーに１つのニルを書き込み、TPM_Tree_Extendを使用してその値を保護する。これにより、コンポーネントが新しい独立した部分木を開始することが保証される。
３．エージェントがコンポーネント自体を測定し、その測定値をＳＭＬに書き込み、TPM_Tree_Extendを使用して値を拡張して根のＰＣＲに入れる。
４．エージェントは、マニフェストおよび各項目に対して以下を繰り返す。必要なメタデータを収集する。そのメタデータを妥当性検査データシーケンスに付加する。メタデータの測定結果、例えば暗号ダイジェストを取得する。そのメタデータの測定結果をＳＭＬツリーに挿入する。TPM_Tree_Extendを使用してメタデータの測定結果を拡張して根のＰＣＲに入れる。
５．エージェントは、そのコンポーネントの測定結果シーケンスが２の２乗に達するまでニル値を付加することにより、コンポーネントの部分木を埋める。

【０３３０】

この結果図３０に示す形態のコンポーネントの部分木構造が得られ、ｍ（．）は、適切な測定関数（各種形態のメタデータ間で異なってよい）を表す。

【０３３１】

メタデータの測定結果は、別の場所、例えば内部ノードに挿入してもよい。メタデータの挿入は手順上測定プロセスから切り離してもよく、これは例えばプラットフォームの起動を早めるために有用である可能性がある。そして、メタデータは別のプロセスで収集し、ＳＭＬツリーに挿入されるまでセキュアに記憶しておくことができる（少なくとも何らかの形態の完全性保護を適用して）。測定エージェントは、コンポーネント測定マニフェストにある要素の数に応じてニル値で部分木を埋めることにより、部分木中に十分な空き空間を作成する。起動時にコンポーネントの測定結果そのままであるツリー型ＳＭＬが完成すると、メタデータを測定し、TPM_Tree_Node_Verified_Updateコマンドを使用して該当する場所に挿入することができる。

【０３３２】

コンポーネント部分木に含めることができるいくつかの有用な種のメタデータとその潜在的な有用性を以下で説明する。

【０３３３】

コンポーネント識別子。例えば、コンポーネントの製造者、名称、バージョン、ビルド、発売日等からなるコンポーネント識別子は、含めることができる最も基本的なメタデータである。基本的にはコンポーネントの測定値自体がすでにコンポーネントを一意に識別するが、独立した識別子は、少なくないセマンティックを追加する。コンポーネント識別子は、測定の結果とは別個に、プラットフォームが当該コンポーネントのロードを試みたことを表明する。従って、特にコンポーネントコードまたはデータが侵害された場合は、どのコンポーネントが完全性の検証に合格しなかったかについての独立した情報がコンポーネント識別子からバリデータに与えられる。この情報は、対象コンポーネントのための正しいＴＲＶを見つけることを可能にするため非常に有用である。この情報は、上記方法のコンポーネントの順序の基準として有用である。ＴＣＧ標準によるイベントログまたはＩＭＬはすでにコンポーネント識別子を含んでいるが、通常は検証データによって保護されない。本発明では、コンポーネント識別子をＳＭＬツリーの検証データにバインドし、それにより、提供される完全性保護からより高い保証レベルを得ることができる。コンポーネント識別子は、含めるためのある形式（例えば１６０ビット列）にあるか、または測定してその形式を得てもよい。

【０３３４】

ＲＩＭ証明書。ＴＲＶの適切な識別は、コンポーネントのＳＭＬツリーにバインドする有用なメタデータとなりうる。ＴＲＶの識別は、例えば対応するＲＩＭ証明書のフィンガ

ープリントとして実現することができる。ＴＲＶの識別は、例えば、適正な証明書に従って自身のデータベースまたはＴＴＰを介して保持されているＲＩＭ（ＴＲＶ）を見つけるため、または証明書のステータス情報を得るために（バリデータによって）使用することができる。その場合、ＲＩＭ証明書自体はプラットフォームの妥当性検査データに組み込む必要はなく、バリデータに送信しなくてよい。プラットフォームの起動プロセスがセキュアである場合、例えばセキュアブートプロセスとして実現される場合は、対応するＲＩＭ証明書情報を含めることで追加的なセマンティックを得る。すなわち、ＲＩＭ証明書情報は、その特定のＲＩＭ証明書で証明されたＴＲＶがコンポーネントの測定値との比較で使用されたことを表明する。その場合、例えば鑑定用評価の要件によっては、コンポーネントの実際の測定値はツリー型ＳＭＬ内で古くなっている場合がある。

【０３３５】

コンポーネントのコンテクスト。コンポーネントのコンテクストは、コンポーネントがロードされる実行時環境の記述を含むことができる。

【０３３６】

タイムスタンプ。メタデータとして追加される保護されたタイムスタンプは、コンポーネントのロード時間をバリデータに提供することにより、測定結果にコンテクストを追加する。それにより、バリデータはコンポーネントのロード順序を検証することができる。ロードの順序は、コンポーネント間の依存関係を妥当性検査しなければならない場合に重要である場合がある。また、１つの測定結果の新しさを評定することも可能になり、データ構造が許す場合は、バリデータはコンポーネントの新たな測定結果を要求することができる。これは、測定結果の経時変化プロセスに基づいて行うことができる。測定結果の経時変化プロセスでは、測定結果が定義された寿命を有し、その寿命に達すると、バリデータはコンポーネントの新しい測定結果を要求することができる。さらに、測定結果に個々のタイムスタンプを適用すると構造化データの再順序付けが可能になり、バリデータは元のブートローダを導出することができる。

【０３３７】

機器または動作環境の現在の状態に応じて異なる形で設定することができる、実行時／調達期間設定等のコンポーネント開始パラメータ。

【０３３８】

コンポーネントのセキュリティポリシー。プラットフォームに施行エンティティが含まれている場合は、各コンポーネントに関連付けられ、そのエンティティによって施行されるセキュリティポリシーは、バリデータがプラットフォームの信頼性を評定するための重要なデータとなる。コンポーネントの測定結果およびメタデータを部分木に編成することにより、上記で部分木証明方法に関して説明したように、ＳＣＡ等のＴＴＰにそれらの妥当性検査を委託することができる。

【０３３９】

コンポーネント起動時の地理位置情報は、測定結果の表現力に別の次元（場所）を加える。このメタデータフィールドは、頻繁に場所を変え、アプリケーションも場所に依存する可能性がある（例えば、資産／貨物追跡報告ソフトウェア）、移動する機器の場合に特に有用である可能性がある。コンポーネントが特定の時に測定されたことを報告する代わりに、このメタデータは測定が行われた場所も含むことができる。

【０３４０】

ＴＦＶの実際的な実現のために、コンポーネント部分木は、ほぼ「静的な」性質のデータを含むことにより、下記の認識の問題を回避することができる。所望のメタデータが頻繁に変化するデータまたはプラットフォームに多少特有なデータを含む場合は、そのようなデータに対して、専用の根の検証データレジスタで保護される全く別個のＳＭＬツリー、またはコンポーネントＳＭＬツリーの指定された部分木を保持することが意味をなす可能性がある。そのような一時的なメタデータＳＭＬツリー内での位置を、元のＳＭＬツリーにバインドすることができる。バインドは、単に順序により、またはコンポーネント識別子若しくはコンポーネント測定葉の繰り返し、または一時的なメタデータＳＭＬツリー

の内容を元のＳＭＬツリーの対応コンポーネント部分木に結び付ける他の手段による。
【０３４１】
線形にチェーン化されたＳＭＬとその結果得られるＰＣＲと同様に、ツリー型ＳＭＬおよび根の検証データレジスタは、測定値の順番の小さな変化の影響を受けやすい。従って、ツリーの値は正確な測定結果の葉の順序に依存するため、既知のツリーおよび部分木の根を効率的に認識する際に問題が生じる。この問題に対処するいくつかの異なる可能性を説明する。それらは、測定結果の葉をプラットフォームおよびバリデータに既知の順序で編成して、それにより順番の影響を受けやすいことから発生する計算量を低減するための方策である。効率的なＴＦＶのための技術基準に従って葉の測定結果を編成する方法および技術手順についても説明する。
【０３４２】
Merkle-Damgard変換で得られるＰＣＲ値、すなわち非結合的で非可換の演算による一方向関数の結果を組み合わせて得られるＰＣＲ値は、入力の順序の影響を非常に受けやすい。この原理は、セキュリティの観点から望ましく、この変換の特徴であり、入力データ中のごく些少な変化を検出する能力を反映する。このプロパティは、ツリー型のＳＭＬおよび線形にチェーン化されたＳＭＬ、並びに根の検証データレジスタ（ＰＣＲ）にも、それぞれ同様に当てはまる。Merkle-Damgard変換のこの情報保持性は、含まれる情報の解釈に関しては複雑性を伴う。
【０３４３】
順序化の複雑性の問題は、提出されたツリー型ＳＭＬに従ってバリデータがプラットフォームの評定を試みる時に特にＴＦＶで生じる。不合格コンポーネントを探索するための汎用的なツリートラバース手順を本明細書で述べる。この手順は、不合格の（すなわちバリデータに知られていない）葉の測定結果の割合が高い場合でも、線形ＳＭＬの評価と比較して効率的であることが示された。しかし、この妥当性検査モデルの障害も、変更または順序の破綻、すなわちバリデータが使用する参照ツリーを基準として未知の位置に出現する測定結果が原因で生じる可能性がある。２つの既知の測定葉を１回置換すると、２つの未知の測定葉が生成される。ある位置に測定葉を挿入する、または挿入しないと、続く全ての葉、すなわちその位置の右側にある葉位置が無効になる。この問題を十分に考慮しないと、ＴＦＶの効率性の利得が失われる可能性がある。
【０３４４】
公正を期すれば、これは線形ＳＭＬと比較した時にはそれほど問題にはならない。何故ならば、線形ＳＭＬにも同じ順序の影響を受ける問題があり、ツリー型妥当性検査は、線形の場合より複雑かつ／または計算的に高費用ではないためである。線形ＳＭＬから得られるＰＣＲの場合は、これと同じ問題が、入力された測定結果の全ての置換について参照ＰＣＲ値を維持する必要性、すなわち空間的複雑性として発生する。一方、ＴＦＶは、順序化の問題によりその有利な特徴の一部を失う可能性がある。
【０３４５】
Merkleハッシュツリーは、各種の研究および実証のシステムでメモリの実行時検証に使用することができる。この場合は順序化の問題は発生せず、メモリ保護ツリーの葉へのメモリアドレスの静的な割り当てにわずかに発生する。部分的であってもプラットフォームとバリデータの双方に既知のそのような順序を回復することは、下記方法の背後にある発想の１つである。
【０３４６】
ここで検討している問題は、プラットフォーム（送信側）とバリデータ（ツリー型妥当性検査データの受信側）間の情報の非対称性に関するため、情報理論的な性質である。測定されたコンポーネントの順番は両者間で効率的に共有することが可能な情報であり、ここで効率的とは、特に、バリデータが参照妥当性検査データ（ＳＭＬツリー）の過度に大きいデータベースを維持する必要がなく、バリデータが高費用の事前計算を実行時に行う必要もないことを意味する。
【０３４７】

情報の効率的な事前共有は、ここに記載される要件を満たす１つの方法である。そのために、ＴＦＶプロセスで両者が従う規約を通じて確立された符号化がプラットフォームとバリデータで用いられる。下記に記載する方法は、この一般的な方式の実現を含む。下記の方法は、適用性において、ツリー型の検証および妥当性検査データの場合に制限されない。線形ＳＭＬの従来の事例にも直接適用することができるが、その場合に得られる利益は制限される可能性がある。プラットフォーム妥当性検査のための従来の検証データのソートによる効率の利得は、後者のデータの本質的な構造化（検証データの適切な実現は順序に依存することを前提とする）により増大する。従って、（有向、非周期）グラフや注釈付きのスキップリスト（高速の探索のために最適化された構造でもある）等の他の構造に、順序付けの方法を適用することができる。

【０３４８】

記載する順序付け方法の他に、バリデータが検証データを認識する問題に対する別の手法がある。すなわち、バリデータは、典型的なプラットフォーム構成を時間と共に学習することができる。これは、パーソナルコンピュータ等、頻繁に変化する構成とそれほど頻繁でない妥当性検査を有するプラットフォームには有効性が低い可能性がある。パーソナルコンピュータの場合は、構成が実質的にブート周期のたびに異なる場合があり、または週に１回の更新が一般的であり、在宅勤務の場合は企業ＶＰＮへの再接続が週に何日かしか行われない。ＰＣＲ値を介して送信され、遠隔アテステーションとＴＮＣ等の追加的なプロトコルを介してネットワークに対して立証されるそのようなＰＣの構成は、その都度異なる場合がある。一方、移動機器、Ｍ２Ｍゲートウェイ、およびＨｅＮＢやマルチメディアＳＴＢ等の特殊な組み込み機器はより静的な構成を有し、同時にネットワークに接続し、したがってより頻繁に妥当性検査される。また、それらの更新は、既知の機器構成を常に把握することができるネットワークエンティティによって管理される。そのような機器の場合は、機器構成の学習方式がバリデータで適用されれば十分である場合がある。例えば、初回の妥当性検査時にバリデータの参照ＳＭＬツリーを既知のＴＲＶ（例えば一致する証明書を伴うＲＩＭ）から機器によって構築することができる。そのようなツリーは実際には、同じ構成の機器の参照として使用されない可能性がある。別のプラットフォームの初回の妥当性検査時にその根の値を特定することにより、学習された参照ツリーの部分木も見つけることができ、新しく学習された参照ツリーの妥当性検査と作成のために適宜使用することができる。

【０３４９】

ツリー型ＳＭＬの望ましい順序付けを提供するには２つの基本的な選択肢がある。ＲＴＭによって測定が実施される前に既知の順序を適用（事前の順序付け）する（その場合は正しくないＳＭＬが形成される）か、または、測定プロセスが左側を対象としないようにし、完成したＳＭＬツリーに後からセキュアな方式で順序付けを適用する（事後の順序付け）かのいずれかである。

【０３５０】

事前順序付けの変更の実現の１つは、所与のＳＭＬの順序化に従うようなプログラムの実際のロードの順序である。これが特定のプラットフォームで実現可能である場合は、その他の点ではセキュアな起動と測定プロセスを元のままに保つことができる。別の選択肢は、測定結果のシーケンスをセキュアなキャッシュに記憶し、プラットフォームが起動されてから、保護されたＳＭＬツリーに挿入するものである。ただし、この方法は、測定値をキャッシュにセキュアに転送し、全ての測定が行われるまでキャッシュ内で完全性を保護することを必要とする可能性がある。その後、データの変更を許すことなく、順序付けの動作を記憶データに作用させなければならない。従って、この選択肢は追加的な要件を有する可能性があり、その要件が適正に満たされないとセキュリティ上の脆弱性をもたらす可能性がある。

【０３５１】

事後の順序付けは、システムがすでに稼動している時にバックグラウンドのタスクとして「怠惰な（lazy）実行」を可能にする点が興味深い。該当する時間的制約は、妥当性検

査に使用される時にＳＭＬを順序付けできることである。ツリー型ＳＭＬの葉をセキュアに入れ替える直接的な方式があるので、事後の順序付けは、別の根の検証データレジスタで保護された、第２の空のＳＭＬツリーに作用することができる。元のＳＭＬツリーからこの事後の順序付けされたＳＭＬツリーを構築するために適用されるステップは、プラットフォーム上の何らかの信頼されるエージェントによって実行され、以下の通りである。事後順序付けを行うエージェントが所望のＳＭＬシーケンスのコンポーネント識別子のシーケンスに繰り返されて、１．次の特定されたコンポーネントの葉の位置を特定する（そのためには実際に測定されたコンポーネントの識別子が必要となる。識別子は、測定プロセスのイベントログに記録するか、例えばダイジェスト形態等でコンポーネントの測定葉のシブリングとして直接付加することができる。）２．例えば他箇所で説明したTPM_Tree_Node_Verifyコマンドを使用して、根の検証データレジスタと比較して葉の測定結果を検証する。３．検証に合格すると、TPM_Tree_Extendを使用して、測定結果を順序付けされたツリーに挿入する。そして、４．例えば依存関係の理由で、プラットフォームの全てのコンポーネントに事前順序付けが実施可能でない場合もある。

【０３５２】

広範囲にわたる事後順序付けは、計算費用を招くため望ましくない場合がある。そのため、事前順序付けと事後順序付けを組み合わせた形態が示唆的である。具体的には、プラットフォームのコンポーネントの部分集合を事前に順序付けし、別の恐らくはより小さいコンポーネント部分を測定後に事後順序付けする。その結果、測定で少なくとも２つのＳＭＬツリーが生成され、例えば部分木証明でそれらを結合して完全な測定ツリーにすることができる。

【０３５３】

事前順序付けおよび事後順序付けはプラットフォームで行われ、妥当性検査データを送信する前に実行される動作であるのに対し、第３の別の選択肢は、バリデータにいくらかの計算負荷を課して参照ツリーの正しい順序化を得るものである。そのために、プラットフォームは、コンポーネント識別子のシーケンスを送信して、ツリー型ＳＭＬまたはその部分木の測定結果の順序化を知らせることができる。あるいは、プラットフォームは、葉の測定値の単純なリストを送信し、既知である場合バリデータが各葉のコンポーネントを一意に特定できるようにしてもよい。そして、バリデータは、そのコンポーネント識別子のシーケンスを受け取ってから実際の妥当性検査を行うまでの時間経過を使用して、予想される順序で参照ツリーを構築することができる。この場合、少なくとも置換された葉の割合が過度に大きくない場合は、プラットフォームの完全な参照ツリーを構築する場合よりも必要とされるハッシュ演算が少ない可能性がある。この場合、プラットフォームのＳＭＬツリーの多くの部分木がすでにバリデータのツリーに含まれており、再使用できる可能性がある。プラットフォームから順序付け情報を受け取ると、バリデータの作業は、既知の部分列に対応し、順序が補正されたツリー内で対応する既知の部分木を挿入できる箇所にある葉の部分列を特定することである。

【０３５４】

この手順は、妥当性検査する完全なＳＭＬツリーの大きな部分が規約により固定された順序になっており、より小さな部分木がバリデータに未知の順序である場合に有効である場合がある。この場合、プラットフォームは、妥当性検査対象の完全なＳＭＬツリーと一緒に、またはその前に、部分木のシーケンスおよび完全なツリー内におけるその位置を提出し、バリデータは、参照値から未知のシーケンスの部分木を構築し、自身の参照ツリーの正しい位置に挿入することができる。

【０３５５】

１つの実装ではコンポーネントの辞書式（lexicographic）順序付けを適用することができ、その場合は、ソート後のデータをコンポーネントの参照測定値、例えばＲＩＭ値（ＲＴＭによって行われた実際の測定結果ではない。実際の測定結果は、コンポーネントのセキュリティが損なわれた場合は異なる可能性がある）とすることができる。別のソート基準は、コンポーネントの一意の識別子またはそのダイジェスト値である。このデータも

、上記方法に従ってツリー型ＳＭＬに含めることができる。辞書式順序付けの変形は、測定値および／または他のデータに符号化を割り当て、その結果得られるコードを辞書式ソート基準として使用するものである。

【０３５６】

測定されたコンポーネントは、各自が不合格になる確率に従って順序付けすることができる。これを実用的にするために、バリデータは、例えば妥当性検査されるプラットフォームの大きな母集団から推定される不合格率から、既知のコンポーネントに対して不合格の確率クラスを定義することができる。そして、ある不合格率クラスのコンポーネントをプラットフォームに伝え、そのクラスの用意された部分木に含めることができる。その結果、ツリーのトラバースによる妥当性検査に最適化されたＳＭＬツリーが得られる。これは、このＳＭＬツリーは、トラバースの際に探索から除かれる可能性が高い部分木（不合格の確率が低いコンポーネント）を含んでいるためである。

【０３５７】

プラットフォームのコンポーネントに階層的な編成がある場合は、それに応じて測定結果を順序付けすることができる。例えば、プラットフォームが下位モジュールからなる機能で構成される場合、機能のための部分木を用意することができ、それら個々のモジュールの測定結果を含むことができる。１つの機能に属する部分木の内部では、適用可能な他の順序付けを使用することができる。そして、機能の部分木を順序付けし、組み合わせてプラットフォームの完全なツリー型ＳＭＬとすることができる。

【０３５８】

例えば、機能モジュール間の関係に従ったシステムコンポーネントの階層的な編成では、結果として得られる測定結果の順番に影響を与えうる依存関係がある場合がある。特に一般的な事例は、２つ以上の機能に属するモジュールである（共有ライブラリに似るか、または同様）。そのようなモジュールは、プラットフォームのセキュアな起動時に１回ロードし、測定することができる。機能に従って階層的に編成されたＳＭＬツリー内でそのようなモジュールを正しく捉える１つの方法は以下の通りである。セキュアな起動を行うエージェントは、ロードされた全モジュールを常に把握し、例えばあるモジュール識別子が所定の測定結果のシーケンス内で再度出現した場合に、繰り返しを特定することができる。階層的に順序付けされたＳＭＬツリーを構築する際、モジュールの繰り返しに遭遇すると、エージェントは、まずＳＭＬツリーにおける最初の出現箇所を探し、次いでその最初の出現位置にある実際の測定値を検証し、次いでその値をツリー中の新たな位置に挿入する。

【０３５９】

プラットフォームが、（参照値に照らして）完全性が正しく検証されているもののバリデータに未知のコンポーネントをロードする時には特殊な状況が発生する。順序なしにＳＭＬツリーに挿入すると、不合格のコンポーネント、および後続コンポーネントの配置の障害が生じる可能性がある。バリデータとプラットフォームがどのコンポーネントがバリデータに既知であるかについての知識を共有する場合、プラットフォームは、未知の測定値をソートして別の部分木とすることができ、一方、既知の測定結果の部分木はバリデータに既知の順序で作成される。そしてこの２つのツリーをツリートラバースによる妥当性検査のために結合して、例えば、既知のＳＭＬツリーが左部分木となり、未知のコンポーネントツリーが右のツリーとなるようにする。後者はバリデータの参照ツリー内で、ニル値を含む空のツリーとして作成される。それにより、既知の（左）部分木のために影響されずにツリーのトラバースによる妥当性検査を進行させることができる。

【０３６０】

上記のシステム、方法および装置は、検証データおよび妥当性検査に、ツリー構造等の階層的な構造を生成し、使用する。以下に、そのようなツリー構造をネットワーク妥当性検査で使用する実施形態をさらに記載する。

【０３６１】

ツリー型妥当性検査（ＴＦＶ）は、複雑な種類のネットワーク側プラットフォーム妥当

性検査を可能にするものである。ＴＦＶは構造化された妥当性検査の実現であり、これは、バリデータが構造化データを使用してプラットフォームの状態を評定することを意味する。データの構造は、ある程度プラットフォームの構造を反映し、妥当性検査を支援することができる。構造化された妥当性検査は、他箇所で述べたプラットフォーム妥当性検査および管理（ＰＶＭ）で定義される抽象的なプラットフォーム妥当性検査および管理プロセスの実装である。効果的で効率的な遠隔ＰＶＭを可能にするプラットフォームは、モジュール式の階層アーキテクチャを示す。プラットフォームの妥当性検査が動作するためのデータのカテゴリが導入される。

【０３６２】

ＰＶＭの説明では、プラットフォーム、バリデータおよびプラットフォーム管理機能を助ける他のエンティティの間で転送されるデータの基本的な概念を定義することができる。それらのデータ概念は、セキュリティに関連するプロパティおよびセマンティック内容に従って３つのカテゴリに分けることができる。

【０３６３】

プラットフォームで作成され、全ＰＶＭプロセスで使用されるＰＶＭのデータには４つのカテゴリがある。それらのカテゴリは、妥当性検査を行うプラットフォームに固有である。別の追加的なカテゴリは信頼される参照データであり、これを使用して妥当性検査データを既知のｇｏｏｄな値と比較する。参照データは、プラットフォームの妥当性検査で使用するためにバリデータまたはプラットフォーム内部に存在することができ、セキュアな起動に使用されるか、または妥当性検査および／または管理データとしてバリデータに送ることができる。これを図２５に示し、同図では概念間の相互関係を示している。

【０３６４】

検証データは、プラットフォーム状態または部分的な状態を明確に定義された保証レベルで検証可能に識別するデータである。検証データは検証プロセスで、例えばプラットフォームのセキュアな起動時および／または実行時にプラットフォーム内部で生成される。検証データを保護する目的の１つは完全性であり、その生成、すなわち検証時、および少なくともプラットフォームの動作周期（例えばブート周期）にわたって維持することができる。保護の別の目的は新しさである。検証データの保証レベルを定義する（そして高める）１つの方式は、検証データの一部を、例えばハードウェア保護により特別な保護レベルで保護された検証データとして切り分けるものである。その場合、記憶空間を検証データレジスタと呼ぶ。

【０３６５】

検証データは、保護のために例えば暗号による保護されたバインドを必要とする場合がある。バインドの強度と保護の強度が、検証データの保証レベルを定義する。検証データおよび検証のいくつかの実現例を以下に説明する。

【０３６６】

ＳＭＬは、ロードされたコンポーネントの測定結果から認証ブートプロセスで生成される１６０ビットのＳＨＡ－１値のシーケンスを含むことができる。これらのハッシュ値は、ロードされたコンポーネントを一意に識別する。ＳＭＬ値の保証レベルは、測定を実施したＲＴＭのセキュリティ、および例えば検証データレジスタ、例えばＰＣＲによるＳＭＬの完全性保護に応じて決まる。変形の１つは、ＴＦＶのツリー型ＳＭＬである。

【０３６７】

ＴＰＭのＰＣＲは、保護された検証データレジスタを規範的に表す。ＴＳを内部で検証するための一般的な方法の１つは認証ブートであり、ＴＳが初期化される時、例えば電源投入時に、ＴＣＢの能力を使用して、ロードまたは起動されたソフトウェアまたはハードウェアコンポーネントの信頼性を評定する。認証ブートは、ＴＳの他の部分より前にＲｏＴおよびＴＣＢの特定の機能を開始することによって実現される。それらの部分は、ＲＴＭ（RoT for Measurement）として機能する。これは、後で起動またはロードされるコンポーネントが測定されることを意味する。すなわち、例えばハードウェアコンポーネントの埋め込まれたコードおよびロードされるプログラムの（２進）表現に暗号ダイジェスト

値を形成することにより、コンポーネントと起動後のそれらのステータスおよび構成が一意に識別される。特定の実施形態によれば、測定値は、セキュアな記憶域、例えばＰＣＲに記憶することができ、それが検証データの保護された部分を形成する。

【０３６８】

セキュアブートは、認証ブートの拡張である。これは、スタンドアロンおよびオフラインの機能要件を有する場合がある、セットトップボックスや携帯電話機等の機器に特に重要である。セキュアブートを備える機器の一般的な特性は、例えばネットワークにアクセスする前等、自身の信頼性についての表明を外部に伝えることができない時に信頼される状態のセットで動作できることである。セキュアブートでは、ＴＳはローカルのベリファイア（検証エンティティ）と、ブートプロセスを監視するローカルエンフォーサ（enforcer）とを備え、エンフォーサは、ポリシー施行ポイント（ＰＥＰ：Policy Enforcement Point）とポリシー決定ポイント（ＰＤＰ：Policy Decision Point）の組合せを確立して、セキュアブートプロセスを制御する。ローカルのベリファイアは、新たにロードまたは起動されるコンポーネントの測定値を参照完全性測定（ＲＩＭ）値と比較し、そのコンポーネントをロードまたは起動するか否かを決定する。ＲＩＭ値は、ＴＣＢに常駐するか、または、ＴＲによりＴＳ内部で保護される、例えば記憶空間に置かれる。このようにして、システムがブートした結果、定義された信頼される状態になることが保証される。

【０３６９】

認証ブートの拡張であるセキュアブートでは、プラットフォームは、ローカルベリファイア（検証エンティティ）と、ＰＥＰとＰＤＰの組合せを確立してセキュアブートプロセスを制御する、ブートプロセスを監視するローカルエンフォーサとを備える。ローカルベリファイアは、新たにロードまたは起動されたコンポーネントの測定値をＲＩＭ値と比較し、コンポーネントをロードまたは起動するか否かを決定する。ＲＩＭ値はＴＣＢに常駐するか、ＴＲによるＴＳ内部で保護される、例えば保護された記憶空間に置かれる。このようにして、システムがブートした結果、定義された信頼される状態となることが保証される。セキュアブートで使用されるＲＩＭのシーケンスは検証データからなる。測定シーケンスは、追加的に、保護された検証データとして検証データレジスタ中に拡張することができる。その場合、ＲＩＭ証明書（より正確にはその暗号フィンガープリント）も検証データの保護された部分となり、証明書を発行したＴＴＰおよび基礎となる証明書インフラストラクチャにより保護が与えられる。

【０３７０】

ＷＳＮのために一部の研究者によりプラットフォーム検証の異質の変形が提案され、その変形は「ソフトウェアアテステーション」と呼ばれる。これは、遠隔のバリデータからプラットフォームに送信されるプローブコードを実行するものであり、プラットフォームは、プラットフォーム状態、例えばメモリの内容を調べる。ゼロ知識による証明の慣行では、検証データとしてバリデータに返される情報は調査の実行時間である場合がある。難読化により、測定可能な遅延後に攻撃者が戻り信号を生成できる可能性があることをある程度保証する。

【０３７１】

検証データと異なり、妥当性検査データは、収集されて別のパーティ、すなわちバリデータに提出され、特に、含まれる検証データに照らして検査することによりプラットフォームの状態の信頼性を評定するために使用できるデータの検証データの上位集合である。例えばＴＣＧによる遠隔アテステーションおよびバリデータによるその評価として実現される妥当性検査データをバリデータに提出するプロセスは適正に妥当性検査と呼ばれる。妥当性検査データは、クオートされた検証データレジスタ（例えばＰＣＲ）値等の検証データからなることができる。妥当性検査は、検証データの暗号による検証以外に、ポリシーの評価と、例えば妥当性検査データに関連付けることができる可能な追加的な管理データを使用したバリデータによるアクションのトリガとを含むことができる。

【０３７２】

検証データと同様に妥当性検査データは完全または部分的なシステム状態を識別するが

、加えてより多くの情報を提供してバリデータに対して妥当性検査を効率的にし、妥当性検査結果の粒度のレベルを判定する。妥当性検査データの例には以下がある。ＰＢＡ等により生成される対象システムのプロパティ、またはコンポーネントの名前およびバージョン、コンポーネントのステータスおよびパラメータ、ＴＣＧで定義されたＴＳのプラットフォーム証明書、並びにシステムコンポーネントのベンダ証明書。ＴＴＰの名前。この場合は、例えば証明書チェーン情報等のさらなるデータを取り出すことができる。例えばＴＣＧ　ＩＷＧで規定される入れ子構造のＸＭＬデータ構造で捕捉されたコンポーネントと下位コンポーネント間の関係。妥当性検査対象のプラットフォームの識別。これは、妥当性検査識別と呼ばれ、例えばＡＩＫの鍵対によって実現される。

【０３７３】

妥当性検査の側面の１つは、妥当性検査データを対応する検証データにバインドすることである。このバインドは、推移的な信頼チェーン化の主旨において、検証データの保証レベルを妥当性検査データに引き継ぐ。従って、このバインドは、例えばデジタル署名で検証することができる。妥当性検査データの概念は、バインドの必要性のために制約される場合がある。妥当性検査データは、検証データを使用して、特にデータの完全性をバリデータにより検証することが可能なデータである。そのため、妥当性検査データは、下記で定義する一般の管理データに比べると恣意的でない。

【０３７４】

遠隔アテステーションは、妥当性検査の最初の段階、すなわち、プラットフォームにより署名された妥当性検査データのバリデータへのセキュアな送信の実現である。そして、このバインドの一例は、ＡＩＫでＰＣＲ値に署名するTPM_Quote動作である。その署名を検証し、ＳＭＬからＰＣＲを再計算し、ＳＭＬの測定値を妥当性検査データに示された既知のコンポーネントのＲＩＭに関連付けることにより、バリデータは、示されたコンポーネントが、プラットフォームが例えば認証ブート時に測定したコンポーネントであることを確かめることができる。

【０３７５】

管理データは、妥当性検査データを含み、妥当性検査データを補う。管理データは、妥当性検査データおよび結果に基づいてプラットフォームの管理専用の他のデータに表現力を追加する。管理データを妥当性検査データにバインドすることは理にかなっており、すなわち、管理データの要素は妥当性検査データの関連する要素に記号的にリンクし、その逆も同様である。管理データの信頼性は、特に管理データがＴＴＰから得られる場合は、プラットフォームの妥当性検査とは別個に（例えばその供給元により）評定することができる。

【０３７６】

管理データの典型例は以下である。妥当性検査の結果から管理アクションを推測するポリシー。コードの更新を取り出すことができる場所。事象報告データ。ユーザ通知データ。妥当性検査の結果を条件とする、妥当性検査を行うプラットフォームに提供されるサービス資格証明。

【０３７７】

信頼される参照データは、妥当性検査データを既知のｇｏｏｄな値と比較するために使用されるデータである。信頼される参照データを構成する値は、信頼される参照値（ＴＲＶ）と呼ばれる。最もよく知られる例は、例えばＴＣＧのＭＰＷＧ仕様で規定されるＲＩＭである。信頼される参照値は、ａ）測定結果がＴＲＶに準拠するコンポーネントが起動されることを保証するために、セキュアな起動時にプラットフォーム自体により、または、ｂ）妥当性検査データを既知のｇｏｏｄな値と比較し、それにより妥当性検査時にプラットフォーム状態を評定するためにバリデータにより、純粋に使用することができる。

【０３７８】

そのようなものとして、信頼される参照データは、そのデータについての何らかのセキュリティ表明を通じて信頼されるようになり、表明は、対象となるＴＲＶを使用してバリデータまたはエージェントが検証することができる。そのような検証可能な表明は、例え

ばＴＴＰから発行されるデジタル証明書で実現することができ、この具体例では、いわゆるＲＩＭ証明書を生じさせる。信頼される参照データの信頼の表明は、例えばコンポーネントまたはプラットフォームの外部評価（例えば共通基準ＥＡＬによる）についての追加的な情報も含むことができる。

【０３７９】

分割妥当性検査は、２つ（またはそれ以上）のネットワーク化されたエンティティに妥当性検査作業を分散できるようにする概念である。この発想は妥当性検査手順に的を絞り、特定のアーキテクチャに結び付けられない。しかし、一般的な概念を提示するためのアーキテクチャモデルが選択される。それは、主として、Ｍ２Ｍゲートウェイに接続されたＭ２Ｍ機器のＭ２Ｍゲートウェイとして機能するＭ２Ｍ機器のアーキテクチャであり、Ｍ２ＭゲートウェイはＭＮＯのネットワークに接続される。この概念自体は、そのアーキテクチャに限定されず、何らかの階層的構造を示す異なるアーキテクチャに適用することができる（例えばＨｅＮＢおよび接続されたＵＥ／ＷＴＲＵや、マスタノードがクラスタ中にある機器のクラスタ等）。

【０３８０】

部分木の証明手順および／または発見の選択肢等の本明細書に記載される一般的な方法は、ここで説明する分割妥当性検査を実装するための基本技術である。

【０３８１】

クライアント－サーバモデルを仮定し、クライアントＣがサービスＳへのアクセスを要求し、サーバはＣが信頼される状態にあればそのサービスをＣに提供するとする。Ｃは、妥当性検査データｖ（例えば測定値）をＳに伝達することができる。

【０３８２】

このシナリオでは、Ｓは、受け取った妥当性検査データｖからＣの状態を導出するためのツールを有することができ、ツールは、Ｓがｖを比較する参照値ｒにより実現される。参照値ｒは、ｖの既知のｇｏｏｄな値を提供する参照判断基準に限定されない。参照値は、受け取ったデータｖと機器の信頼性との対応付けを提供する。従って、ｒは、ｖとサービスのアクセスポリシーとの対応付けと見なすことができ、これは、ｒは、受け取ったＶからＳがＣの信頼性についてのステートメントを導出できるようにする任意種の参照であってよいことを意味する。

【０３８３】

シナリオによっては、Ｃは、ゲートウェイ機器Ｇ（例えばＷＴＲＵのゲートウェイとしてのＭ２Ｍゲートウェイ、ＨｅＮＢ）を介してＳに接続することができる。Ｇが、接続されたＣの妥当性検査の最初の部分を行うことができる場合、すなわちＧが接続機器のｒを備えている場合、妥当性検査作業の負荷はＧとＳに分散することができる。

【０３８４】

この２つのネットワーク化エンティティ間の妥当性検査の分散を分割妥当性検査と呼ぶ。分割妥当性検査を機能させるには、Ｇが接続された個々のＣの妥当性を検査し、ＳがＧを信頼してその妥当性検査をセキュアな方式で行わせることができなければならない。さらに、妥当性検査データｖが内部の（階層）構造を示す場合は、効率的な分割妥当性検査を行うのがより容易である可能性がある。

【０３８５】

分割妥当性検査の実装の１つとして、各種エンティティがツリー型妥当性検査（ＴＦＶ）データを使用できる場合がある。Ｇは、自身の構造化妥当性検査データを生成できる場合がある。Ｇは、部分木の削除と追加を含むツリーの更新を行える場合がある。Ｇは、測定エージェント（ＭＡ）を備える場合があり、ＭＡはＣの妥当性検査データをＧの妥当性検査ツリーに統合することができる。それらの部分木は、直接Ｇにより、またはＴＴＰにより証明することができる（従って、妥当性検査および署名できる）場合がある。

【０３８６】

Ｇ内部のＭＡは、接続されたＣから測定結果を集め、それを新しい部分木としてＧのツリーに統合する。測定値の収集は、Ｇに対するＣのローカルの検証時（例えばＳＡＶ、混

合型妥当性検査、遠隔妥当性検査をＣとＧ間のリンクに行うことによる）、または追加的なプロトコルステップとして認証後に行うことができる。そのステップは、Ｃの機器認証にバインドして、リプレイおよび偽造された妥当性検査データを防止することができる。しかし、この認証は、ＳにアクセスするためにＣに使用される認証（例えばネットワークにアクセスするために使用されるＭＮＯ資格証明）を表す必要がない場合もある。何故ならば、この認証は、Ｇがその特定のＣのためにＳとの接続を確立する時に追加的なステップで行うことができるためである。これを図３１に示す。

【０３８７】

関連データを収集すると、ＧはＴＴＰに接触し、ＴＴＰはＧに部分木の証明書を発行する。そして、Ｇは、その証明書を自身の妥当性検査ツリーに組み込んで、新しい、更新された妥当性検査ツリーを得ることができる。この変形として、ＴＴＰをＧの一部としてもよく、例えば、ＧのＴｒＥ内部で実行される、またはＴｒＥにより完全性が保護されるアプリケーションとして実装することができる。これを図３２に示す。

【０３８８】

最後のステップで、Ｇは、接続された機器の部分木の一部または全てに取って替わる、証明書を含む更新後の妥当性検査ツリーをＳに送信する。説明のために、Ｓを、妥当性検査エンティティＶＥとサービスエンティティＳＥの２つの下位ユニットに分解することができる。ＶＥはＧから受け取ったデータを（自律的に、またはＴＴＰの支援により）妥当性検査することができるのに対し、ＳＥは、Ｃに提供される実際のサービス（および場合によってはＧに提供されるサービス）を担う。ＧがＳにアクセスするための資格証明でＣを認証することができない変形例では、ＳＥも認証を行ってよく、例えば、Ｓが３Ｇ認証を行い、Ｇは機器のＴｒＥを認証して、受け取った妥当性検査データの真正性および接続されたＣの機器／ＴｒＥ識別を検証する。これを図３３に示す。妥当性検査に合格すると、Ｃにサービスを提供することができる。

【０３８９】

分割妥当性検査では、接続機器の部分木の代わりとなる証明書をＳが受け取ることができるため、Ｇは、接続されたＣのプライバシーをＳに対して保護することができる。ＳはＧのＴＴＰおよびＭＡを信頼することができ、ＭＡの信頼は、Ｇから受け取る妥当性検査データから導出することができる。これは、妥当性検査データがＭＡの測定結果を含んでいるためである。

【０３９０】

さらなる変形例では、分割妥当性検査を拡張して動的な更新に対応することができ、これには、Ｇ間でＣの部分木をハンドオーバーすることを伴う、異なるＧ間を移動するＣへの対応が含まれる。分割妥当性検査の別の変形例は、ゲートウェイＧで１つのＣを部分的に妥当性検査して、部分木の一部のみを証明書に置き換えるものである。

【０３９１】

本明細書に記載される方法およびアルゴリズムは、分割妥当性検査の動作を効率的に実装することを可能にする。それら方法およびアルゴリズムは、機器がツリーまたは部分木を構築、報告、更新、およびクオートすることを可能にするＴＰＭの拡張バージョンに基づいて、分割妥当性検査の使用を可能にする。さらに、アルゴリズムで、部分木証明により部分木ノードを証明書に効率的に置き換えられるようにする。

【０３９２】

分割妥当性検査における問題の１つは、どのＣをＴＴＰで妥当性検査できる（およびＧでその部分木を置き換える）か、およびＣのどの部分をＳで妥当性検査できるか、すなわち接続されたＣの部分木を証明するためにＧが使用できるＴＴＰで利用可能な参照値を判定することである。一般に、この発見問題の解決は、ＴＴＰを利用する発見、ゲートウェイによる発見、および分担型の発見、の３つのクラスに分割することができる。

【０３９３】

ＴＴＰを利用する手法は、証明すべき該当する部分木を見つけるための複雑な分析を行う計算力を持たないＧに適する。この手法では、ＴＴＰは、Ｇから完全なツリーを受け取

り、ＴＴＰが検証できる部分木の部分木証明書のセットの形で結果を返す。そして、ＴＴＰは、証明することができない残りのデータを破棄することができる。証明書は、ツリー中で該当する箇所についての標識と共にＧに送り返される。ＧのＭＡは、証明書の取り込みを行う。すなわち、証明された部分木を受け取った証明書に置き換える。ＴＴＰは証明書と共に部分木の場所についての標識をメッセージで渡すので、Ｇは発見を再度行う必要はない。

【０３９４】

変形例の手順では、Ｇは完全なツリーを送信するのではなく、ツリーの一部をＴＴＰに送信して、ＴＴＰが接続機器から得た測定結果の部分集合から、証明できる可能性のある部分木を探索できるようにする。このような機構は、例えば、Ｇが何らかの機器特性（例えば機器クラス）に基づいてツリーを部分木に分解することを可能にする何らかの構造をツリー型妥当性検査データが示す場合に有用である可能性がある。機器クラスに応じて、異なるＴＴＰが特定の機器の集合を妥当性検査できる可能性がある。

【０３９５】

ゲートウェイによる発見では、Ｇは、どのＣがＴＴＰで証明できるかを判定するための判断基準を有し、ＴＴＰは必要なデータを受け取ることができる。この手法では、全ての転送されるデータが証明可能な部分木に属するため、ＧとＴＴＰの間の通信量を最小にすることができる。ただし、Ｇは、Ｇが右部分木およびその部分木を証明できるＴＴＰを発見することを可能にする適切な判断基準を予め備えることができる。さらに、この方法では、Ｇは自身の妥当性検査ツリーから適切な部分木を探すことができ、Ｇにより高い計算労力が課される。

【０３９６】

分担発見モデルでは、ＴＴＰとＧが協働して証明すべき機器および部分木を判断する。そのような発見ステッププロトコルの結果は、証明可能な部分木のリストとなる。この発見プロトコルは、機器クラス、能力、機器プロファイル、負荷や帯域幅等の動的なパラメータ等、追加的な（メタ）データの交換を含むことができる。ＧからＴＴＰに不必要なデータが送信されないため、証明可能な部分木の集合を判定するために必要な初期通信にも関わらず、この手法でもデータ量が最小に抑えられる。通信負荷は、ゲートウェイによる発見より高い可能性があるが、ＴＴＰによる発見よりは低い可能性がある。Ｇは、ネゴシエーション段階の結果を記憶できる場合があり、その場合Ｇは後にゲートウェイにおける決定を行うことができる。分担発見モデルでは、Ｇは、証明可能な部分木の集合を事前に与えられ、それを分担発見段階で更新することができる。

【０３９７】

一変形例では、ＧはいくつかのＣを完全に妥当性検査し、Ｃの選択は、例えばＧがフェムトセルである場合には、ＣＳＧ（Closed Subscriber Group）に基づくことができる。そして、Ｓが残りのＣを完全に妥当性検査する。Ｇで妥当性検査すべきＣの集合は、静的または動的なデータに基づくことができ、Ｓで妥当性検査すべき機器の集合は残りのＣであるか、または重複がある場合があり、一部の機器が２度妥当性検査される場合があることを意味する。

【０３９８】

ＧおよびＳは、アスペクトのセットに定義されたアスペクトを妥当性検査することができ、アスペクトのセットは、事前に規定されるか、動的に調整され、重複がある場合がある。そのようなアスペクトのセットは、機器プロファイルの完全性、ブートコード／ＯＳ／ドライバの完全性、キャピラリ（capillary）ネットワーク機能／コンポーネントの完全性、ＷＡＮ接続機能／コンポーネントの完全性、高レベルアプリケーションの完全性、およびＧで使用される信頼される参照値の完全性、を含むことができる。

【０３９９】

妥当性検査は、負荷に基づいて動的に分割することができる。この動的な手法では、ＧおよびＳ、またはＧ、並びにＴＴＰが各自の計算負荷と、接続されたＣとＧとの間のリンク、並びにＧおよびＳまたはＧとＴＴＰとの間のリンクの計算負荷を比較する。その負荷

に基づいて、ＴＴＰまたはＳは、Ｇで直接妥当性検査できる機器の集合を判定する。予め規定された集合が存在してもよく、その集合が動的に更新される。

【０４００】

知識に基づく分割では、知識のセットおよび妥当性検査ポリシーに基づいて、Ｓ（またはＴＴＰ）とＧとの間で分割を決定することができる。知識のセットは、挙動（過去に観察された挙動、または将来の予測される挙動）、信頼性（過去の履歴、現在の値、または将来の予測）、コスト対利益の分析、妥当性検査の重要性等の数個の要素を含むことができ、それによりＳおよびＧは妥当性検査ポリシーを導出または更新することができる。

【０４０１】

一例では、Ｈ（ｅ）ＮＢがＵＥの分割妥当性検査を使用することができる。ＨｅＮＢの場合、アーキテクチャは、図３４に示す下記手順により接続されたＵＥの一部または全てを妥当性検査できるＨ（ｅ）ＮＢを備えることができる。

【０４０２】

図３４に示すように、符号１で、ＷＴＲＵが完全性の測定結果をＨ（ｅ）ＮＢに送信する。符号２で、Ｈ（ｅ）ＮＢが受け取ったデータを組み込み、一部（または全て）のＷＴＲＵを妥当性検査する。符号３で、ＳｅＧＷ／ＰＶＥが接続機器を含むＨ（ｅ）ＮＢツリーの妥当性検査を行う。ＰＶＥは、符号４で、まだ妥当性検査されていない機器を検査し、参照証明書を発行する。証明書はＨ（ｅ）ＮＢに送り返されて将来の通信に含められ、Ｈ（ｅ）ＮＢが将来接続を試みる際にＷＴＲＵを妥当性検査できるようにする。

【０４０３】

別の変形は企業展開シナリオにおけるものであり、ＨｅＮＢが各ＷＴＲＵのコンポーネント、例えば社用ソフトウェアの部分集合を妥当性検査することができ、コンポーネントは、ＭＮＯ（ＳｅＧＷおよびＰＶＥ）に公開されない。ＰＶＥは、ＷＴＲＵの基礎となるプラットフォームコンポーネントのネットワーク側の妥当性検査を行うことができる。

【０４０４】

Ｈ（ｅ）ＮＢに接続するローグ（rogue）機器の場合、Ｈ（ｅ）ＮＢは、報告の目的でＵＥの完全性ツリーを含むことができる。企業ネットワーク上のＰＶＥまたは妥当性検査を行うネットワークエンティティは、受け取ったデータを検証できる場合があり、その機器のアクセスを阻止する、またはコンポーネントまたはソフトウェアの入れ替えやアンチウィルスの更新の実施等の修復ステップを行うようにＨ（ｅ）ＮＢに指示することができる。これを図３５に示す。

【０４０５】

Ｍ２Ｍアーキテクチャは、Ｈ（ｅ）ＮＢと多少の類似点があるため、分割妥当性検査はＭ２Ｍシナリオに利益をもたらす可能性がある。典型的なＭ２Ｍの展開では、小型であるがより多くの機器が、ネットワークへのアクセスを提供する１つのゲートウェイに接続する可能性がある。さらにＭ２Ｍシナリオでは、幅広い種類の機器が１つのゲートウェイを介して接続される場合があり、ゲートウェイが全ての機器（または全ての機器種別）を妥当性検査できない場合がある。分割妥当性検査を行うと、ネットワークが労力の一部をＭ２Ｍゲートウェイに移すことが可能になる。

【０４０６】

さらなる変形では、Ｍ２Ｍ　ＧＷが接続機器を、それらの種類、機器クラス、機器プロパティ、または接続プロファイル（プロトコル、接続ネットワーク、帯域幅）に基づいてグループに分類し、機器の妥当性検査ツリーについてのグループ証明書を提供することができる。これを図３６に示す。

【０４０７】

別の変形例はピアツーピアの手法であり、複数のＭ２Ｍゲートウェイ（ＧＷ）がクラスタ構造を示し、それにより、より好適なリンク（例えば、高い帯域幅、低待ち時間等）で通信することができる。各Ｍ２Ｍ　ＧＷは、ネットワークとの専用のバックホール（backhaul）リンクを有する。ただし、ローカルの通信（例えばＷｉＦｉ、Ｂｌｕｅｔｏｏｔｈ（登録商標）、ＺｉｇＢｅｅ（登録商標）、ＭＢＵＳを介する）は、バックホールリンク

（例えば３Ｇ）を使用するよりも低コストである可能性がある。これは、ローカルネットワークにトラフィックを移すことで多少の利益が得られることを意味する。Ｍ２Ｍ　ＧＷ間のローカルネットワークをローカルエリアエクスチェンジネットワーク（ＬＡＥＮ）と呼ぶ。ＬＡＥＮ上の全てのＭ２Ｍ　ＧＷは相互に認証することができ、それによりＭ２Ｍ　ＧＷは、ＬＡＥＮ上の他のＭ２Ｍ　ＧＷから送られてくるメッセージを信頼することができる。ＬＡＥＮ内の通信は、完全性および真正性を保護することができる。プライバシーを要求するシナリオでは、完全に暗号化されたＬＡＥＮを確立して秘密性の保護を提供することができる。

【０４０８】

Ｍ２Ｍ　ＧＷが部分木を証明することができない場合（例えば部分木を証明するＴＴＰが見つからない場合や、機器がＭ２Ｍ　ＧＷに未知である場合）、Ｍ２Ｍ　ＧＷは、適切な証明書を取得するための要求メッセージと共に部分木をＬＡＥＮにプッシュする（図３７のステップ１）。別のＭ２Ｍ　ＧＷが部分木を証明することができる場合（独力で、または第１のＭ２Ｍ　ＧＷからは接触できない、若しくは知られていないＴＴＰを介して）、そのＭ２Ｍ　ＧＷは、要求元のＧＷに証明書を返す（図３７のステップ２）。そして、Ｍ２Ｍ　ＧＷは証明書を自身の妥当性検査ツリーに組み込むことができる。Ｍ２Ｍ　ＧＷは、それと同じ機器クラスの接続機器の妥当性を将来検査するために、証明書（または証明書を作成するために必要な参照値、若しくはＴＴＰのアドレス）を記憶することもできる。

【０４０９】

部分木の交換は、Merkleハッシュツリーの交換を可能にするツリーハッシュエクスチェンジ（ＴＨＥＸ）形式を使用して行うことができる。この交換形式は、ハッシュツリーのシリアライズ表現からなる。この交換形式では、テキストまたはバイナリの複数のペイロードを可能にするダイレクトインターネットメッセージカプセル化（ＤＩＭＥ）形式を使用する。Merkleハッシュツリーのシリアライズ形式は、２つの異なるペイロードからなる。第１のペイロードはハッシュツリーに関するＸＭＬ符号化されたメタデータであり、第２のペイロードは、ツリー自体の２値のシリアライズである。

【０４１０】

所与の部分木を妥当性検査することが可能なＬＡＥＮ上のネットワークエンティティ、例えば他のＭ２Ｍ　ＧＷの特定では、分散ハッシュテーブル（ＤＨＴ）構造を使用して、特定の機器種別または機器クラスを妥当性検査することができたＬＡＥＮ上のノードのネットワークアドレスを記憶することができる。Ｐ２ＰネットワークからＤＨＴの概念により、妥当性検査に関連するノードを高速に検出することができる。さらに、ＬＡＥＮ内の全てのノードにハッシュテーブルを記憶し、ノード間のＬＡＥＮマルチキャストメッセージを介してテーブルを更新し、ＬＡＥＮ内の全てのノードがどこで特定の部分木を妥当性検査できるかが分かるようにすることが可能な場合もある。

【０４１１】

上記で分割妥当性検査および部分木証明等の他の構造化された妥当性検査技術について開発された概念および方法論は、システムの妥当性検査だけでなく、他のセキュリティ関連機能にも使用することができる。

【０４１２】

例えば、ネットワークエンティティＮＥ、ゲートウェイＧ、および複数の機器Ｄからなるネットワークの分割認証モデルでは、ゲートウェイＧが、ＮＥがＧを認証できるようにＮＥと対話する時に、Ｄについての認証情報を含めて、ＮＥがＧとＤの集まりを同時に認証できることを構想することができる。変形例では、ＧがＤを認証することができ、ＮＥがＧを認証するために必要な情報を伝達するために後に送信するメッセージに、認証ステップおよびその結果についての報告を含める。上記のように、Ｄを分割して、数個の異なる基準に従って、ＮＥが認証できるＤをＧが認証することも可能である場合がある。

【０４１３】

同様に、鍵の導出や鍵管理（例えば配布、更新、および使用停止を含む）等の他のセキ

ュリティ機能、認証、およびアクセス制御も同様にして行うことができる。

【０４１４】

部分木検証のために開発されたこれらの技術は、他のセキュリティ機能でも同様に使用することができる。

【０４１５】

方法、アルゴリズムおよび装置は、セキュリティ／プライバシー／権利管理、ハイブリッドネットワーク、協調通信（cooperative communication）を含む任意の技術分野に適用することができ、ユーザ機器（ＵＥ）、ＷＴＲＵ、電話機、データカード、ラップトップ／ネットブック、ゲーム機、インフラストラクチャ機器、基地局／ノードＢ、フェムト基地局、アクセスポイント、ＢＳＣ／ＲＮＣ、ゲートウェイ、アプリケーションサーバ、システム、セッション層、プレゼンテーション層、アプリケーション層、ＤＳＰ、ソフトウェア、ハードウェア、ＡＳＩＣで使用することができる。方法およびアルゴリズムは、アプリケーション、システム情報、プラットフォームセキュリティ、システム全体のセキュリティ、呼受付け、ホームｅＮｏｄｅ　Ｂ（ＨｅＮＢ）、ＨＮＢ、フェムトセル、ハードウェアまたはソフトウェアの実装、課金管理、加入者管理、ポリシー制御、サービス品質（Ｑｏｓ）、セキュリティ、信頼、暗号化、登録／ＡＡＡ等にも適用可能である。

【０４１６】

上記では特徴および要素について特定の組合せで説明したが、各特徴および要素は、他の特徴を用いずに単独で、または他の特徴若しくは要素との各種組合せで若しくは他の特徴若しくは要素を用いずに組み合わせて使用することができる。本明細書に提供される方法およびフローチャートは、汎用コンピュータまたはプロセッサによる実行のためにコンピュータ可読記憶媒体に組み込まれたコンピュータプログラム、ソフトウェア、またはファームウェアとして実施することができる。コンピュータ可読記憶媒体の例には、ＲＯＭ、ＲＡＭ、レジスタ、キャッシュメモリ、半導体メモリ装置、内蔵ハードウェアディスクやリムーバブルディスク等の磁気媒体、光磁気媒体、およびＣＤ－ＲＯＭディスクやデジタル多用途ディスク（ＤＶＤ）等の光学媒体が含まれる。

【０４１７】

適切なプロセッサには、例えば、汎用プロセッサ、特殊目的プロセッサ、従来のプロセッサ、デジタル信号プロセッサ（ＤＳＰ）、複数のマイクロプロセッサ、ＤＳＰコアと関連した１つまたは複数のマイクロプロセッサ、コントローラ、マイクロコントローラ、特定用途集積回路（ＡＳＩＣ）、特定用途向け標準製品（ＡＳＳＰ）、利用者書き換え可能ゲートアレイ（ＦＰＧＡ）回路、任意の他の種の集積回路（ＩＣ）、および／または状態機械が含まれる。

【０４１８】

ソフトウェアと関連したプロセッサを使用して、ＷＴＲＵ、ユーザ機器（ＵＥ）、端末、基地局、移動管理エンティティ（ＭＭＥ）若しくはＥＰＣ（Evolved Packet Core）、または任意のホストコンピュータで使用するための無線周波トランシーバを実装することができる。ＷＴＲＵは、ハードウェアおよび／またはソフトウェアとして実装されたモジュールと併せて使用することができ、それらには、ソフトウェア定義無線（ＳＤＲ）、および他のコンポーネント、例えば、カメラ、ビデオカメラモジュール、テレビ電話、スピーカ電話、振動装置、スピーカ、マイクロフォン、テレビトランシーバ、ハンドフリーヘッドセット、キーボード、Ｂｌｕｅｔｏｏｔｈモジュール、周波数変調（ＦＭ）無線ユニット、近距離通信（ＮＦＣ）モジュール、液晶ディスプレイ（ＬＣＤ）表示装置、有機発光ダイオード（ＯＬＥＤ）表示装置、デジタル音楽プレーヤ、メディアプレーヤ、ビデオゲームプレーヤモジュール、インターネットブラウザおよび／または任意の無線ローカルエリアネットワーク（ＷＬＡＮ）若しくは超広帯域（ＵＷＢ）モジュール、が含まれる。

【０４１９】

さらなる研究でこの方向性を追求し、ツリー型の検証および妥当性検査データ、いわゆるツリー型妥当性検査（ＴＦＶ）によるプラットフォーム妥当性検査のための具体的なアーキテクチャを検討することができる。想到可能な選択肢の１つは、既知のコンポーネン

トの下部構造、例えば順にロードされる依存プログラムや一律の関連セキュリティポリシーを有するコンポーネントの部分木を使用するモジュール式の構築を可能にするような方式で、ＳＣＡおよび／またはバリデータにより参照ツリーのデータベースを効率的に編成するものである。部分木の発見、妥当性検査されるプラットフォームコンポーネント間の依存関係の表現、およびＴＦＶの結果に基づくプラットフォームの管理（更新、修復）のためのアーキテクチャおよび方法は、現在研究が行われている課題である。

【０４２０】

本明細書に記載されるシステム、方法および装置は、下記で説明し、図３８、図３９、および図４０に示すような通信システムで実施することができる。

【０４２１】

図３８は、１つまたは複数の開示実施形態を実施することが可能な例示的通信システム１００の図である。通信システム１００は、音声、データ、映像、メッセージング、放送等のコンテンツを複数の無線ユーザに提供する多重接続システムとすることができる。通信システム１００は、複数の無線ユーザが、無線帯域幅を含むシステム資源の共有を通じてそのようなコンテンツにアクセスすることを可能にすることができる。例えば、通信システム１００は、符号分割多重接続（ＣＤＭＡ）、時分割多重接続（ＴＤＭＡ）、周波数分割多重接続（ＦＤＭＡ）、直交ＦＤＭＡ（ＯＦＤＭＡ）、単一キャリアＦＤＭＡ（ＳＣ－ＦＤＭＡ）等の１つまたは複数のチャネルアクセス方法を用いることができる。

【０４２２】

図３８に示すように、通信システム１００は、ＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１０２ｄ、無線アクセスネットワーク（ＲＡＮ）１０４、コアネットワーク１０６、公衆交換電話網（ＰＳＴＮ）１０８、インターネット１１０、および他のネットワーク１１２を含むことができる。ただし、開示される実施形態では、任意数のＷＴＲＵ、基地局、ネットワーク、および／またはネットワーク要素を企図することが理解されよう。各ＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１０２ｄは、無線環境で動作および／または通信するように構成された任意種の装置であってよい。例えば、ＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１０２ｄは、無線信号を送信および／または受信するように構成することができ、ユーザ機器（ＵＥ）、移動局、固定型または移動型の加入者ユニット、ページャ、携帯電話、携帯情報端末（ＰＤＡ）、スマートフォン、ラップトップ機、ネットブック、パーソナルコンピュータ、無線センサ、消費者電子製品を含むことができる。

【０４２３】

通信システム１００は、基地局１１４ａおよび基地局１１４ｂも含むことができる。各基地局１１４ａ、１１４ｂは、ＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１０２ｄの少なくとも１つとワイヤレスにインタフェースをとって、コアネットワーク１０６、インターネット１１０、および／またはネットワーク１１２等の１つまたは複数の通信ネットワークへのアクセスを容易にするように構成された任意種の装置であってよい。例えば、基地局１１４ａ、１１４ｂは、ベーストランシーバ局（ＢＴＳ）、ノードＢ、ｅノードＢ、ホームノードＢ、ホームｅノードＢ、サイトコントローラ、アクセスポイント（ＡＰ）、無線ルータ等である。図では基地局１１４ａ、１１４ｂは１つの要素として図示するが、基地局１１４ａ、１１４ｂは任意数の相互接続された基地局および／またはネットワーク要素を含んでよいことは理解されよう。

【０４２４】

基地局１１４ａは、ＲＡＮ１０４の一部とすることができ、ＲＡＮ１０４は、基地局コントローラ（ＢＳＣ）、無線ネットワークコントローラ（ＲＮＣ）、中継ノード等、他の基地局および／またはネットワーク要素（図示せず）も含むことができる。基地局１１４ａおよび／または基地局１１４ｂは、セルと呼ぶ場合もある特定の地理領域（図示せず）内で無線信号を送信および／または受信するように構成することができる。セルはさらにセルセクタに分割することができる。例えば、基地局１１４ａに関連付けられたセルを３つのセクタに分割することができる。そのため、一実施形態では、基地局１１４ａは、セルの各セクタに１つの計３つのトランシーバを含むことができる。一実施形態では、基地

局１１４ａは、ＭＩＭＯ技術を用いることができ、セルの各セクタに複数のトランシーバを利用することができる。

【０４２５】

基地局１１４ａ、１１４ｂは、エアインタフェース１１６を通じてＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１０２ｄの１つまたは複数と通信することができる。エアインタフェース１１６は、任意の適切な無線通信リンク（例えば無線周波（ＲＦ）、マイクロ波、赤外線（ＩＲ）、紫外線（ＵＶ）、可視光等）。エアインタフェース１１６は、適切な無線アクセス技術（ＲＡＴ）を使用して確立することができる。

【０４２６】

より具体的には、通信システム１００は、多重接続システムであってよく、ＣＤＭＡ、ＴＤＭＡ、ＦＤＭＡ、ＯＦＤＭＡ、ＳＣ－ＦＤＭＡ等の１つまたは複数のチャネルアクセス方式を用いることができる。例えば、ＲＡＮ１０４の基地局１１４ａとＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃは、ＵＴＲＡ（Universal Mobile Telecommunication System (UMTS) Terrestrial Radio Access）等の無線技術を実装することができ、その場合は広帯域ＣＤＭＡ（Ｗ－ＣＤＭＡ（登録商標））を使用してエアインタフェース１１６を確立することができる。Ｗ－ＣＤＭＡは、ＨＳＰＡ（High-Speed Packet Access）および／またはＨＳＰＡ＋（Evolved HSPA）等の通信プロトコルを含むことができる。ＨＳＰＡはＨＳＤＰＡ（High-Speed Downlink Packet Access）および／またはＨＳＵＰＡ（High-Speed Uplink Packet Access）を含むことができる。

【０４２７】

一実施形態では、基地局１１４ａおよびＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃは、Ｅ－ＵＴＲＡ（Evolved UMTS Terrestrial Radio Access）等の無線技術を実装することができ、その場合、エアインタフェース１１６はＬＴＥおよび／またはＬＴＥ－Ａｄｖａｎｃｅｄ（ＬＴＥ－Ａ）を使用して確立することができる。

【０４２８】

他の実施形態では、基地局１１４ａおよびＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃは、ＩＥＥＥ８０２．１６（すなわちＷｉＭＡＸ（Worldwide Interoperability for Microwave Access））、ＣＤＭＡ２０００、ＣＤＭＡ２０００　１Ｘ、ＣＤＭＡ２０００　ＥＶ－ＤＯ、ＩＳ－２０００（Interim Standard 2000）、ＩＳ－９５（Interim Standard 95）、ＩＳ－８５６（Interim Standard 856）、ＧＳＭ（登録商標）（Global System for Mobile Communication）、ＥＤＧＥ（Enhanced Data rates for GSM Evolution）、ＧＳＭＥＤＧＥ（ＧＥＲＡＮ）等の無線技術を実装することができる。

【０４２９】

図３８の基地局１１４ｂは、例えば無線ルータ、ホームノードＢ、ホームｅノードＢ、またはアクセスポイントであり、職場、家庭、乗り物、学校構内等の局所的な領域内での無線接続を容易にする任意の適切なＲＡＴを利用することができる。一実施形態では、基地局１１４ｂおよびＷＴＲＵ１０２ｃ、１０２ｄは、ＩＥＥＥ８０２．１１等の無線技術を実装して無線ローカルエリアネットワーク（ＷＬＡＮ）を確立することができる。一実施形態では、基地局１１４ｂおよびＷＴＲＵ１０２ｃ、１０２ｄは、ＩＥＥＥ８０２．１５等の無線技術を実装して無線パーソナルエリアネットワーク（ＷＰＡＮ）を確立することができる。さらに一実施形態では、基地局１１４ｂおよびＷＴＲＵ１０２ｃ、１０２ｄは、セルラ方式のＲＡＴ（例えば、Ｗ－ＣＤＭＡ、ＣＤＭＡ２０００、ＧＳＭ、ＬＴＥ、ＬＴＥ－Ａ等）を利用してピコセルまたはフェムトセルを確立することができる。図３８に示すように、基地局１１４ｂは、インターネット１１０への直接の接続を有することができる。そのため、基地局１１４ｂは、コアネットワーク１０６を介してインターネット１１０にアクセスする必要がない場合もある。

【０４３０】

ＲＡＮ１０４はコアネットワーク１０６と通信状態にあることができ、コアネットワークは、ＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１７０２ｄの１つまたは複数に音声、データ、アプリケーションおよび／またはＶｏＩＰ（voice over Internet Protocol）サー

ビスを提供するように構成された任意種のネットワークであってよい。例えば、コアネットワーク１０６は、呼制御、課金サービス、モバイル位置を利用するサービス、料金前払いの通話、インターネット接続、ビデオ配布等を提供する、かつ／またはユーザ認証等の高レベルなセキュリティ機能を行うことができる。図３８には示さないが、ＲＡＮ１０４および／またはコアネットワーク１０６は、ＲＡＮ１０４と同じＲＡＴまたは異なるＲＡＴを用いる他のＲＡＮと直接通信状態にあっても、または間接的な通信状態にあってもよいことが理解されよう。例えば、Ｅ－ＵＴＲＡ無線技術を利用する可能性のあるＲＡＮ１０４に接続されるのに加えて、コアネットワーク１０６は、ＧＳＭ無線技術を用いる別のＲＡＮ（図示せず）とも通信状態にあることができる。

【０４３１】

コアネットワーク１０６は、ＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１１２ｄがＰＳＴＮ１０８、インターネット１１０および／または他のネットワーク１１２にアクセスするためのゲートウェイの役割を果たすこともできる。ＰＳＴＮ１０８は、従来の電話サービス（ＰＯＴＳ）を提供する回路交換電話網を含むことができる。インターネット１１０は、ＴＣＰ／ＩＰインターネットプロトコル群のＴＣＰ、ＵＤＰ、ＩＰ等の一般的な通信プロトコルを使用する相互接続されたコンピュータおよび装置からなる世界規模のシステムを含むことができる。ネットワーク１１２は、他のサービス提供者に所有および／または運営される有線または無線の通信ネットワークを含むことができる。例えば、ネットワーク１１２は、ＲＡＮ１０４と同じＲＡＴまたは異なるＲＡＴを用いる可能性のある１つまたは複数のＲＡＮに接続された別のコアネットワークを含むことができる。

【０４３２】

通信システム１００内のＷＴＲＵ１０２ａ、１０２ｂ、１０２ｃ、１０２ｄの一部または全ては、マルチモード機能を備えることができる。すなわち、ＷＴＲＵ１０２ａ、１０２ｂ、１１２ｃ、１０２ｄは、種々の無線リンクを通じて種々の無線ネットワークと通信するための複数のトランシーバを含むことができる。例えば、図３８に示すＷＴＲＵ１０２ｃは、セルラ方式の無線技術を用いる可能性のある基地局１１４ａ、およびＩＥＥＥ８０２無線技術を用いる可能性のある基地局１１４ｂと通信するように構成することができる。

【０４３３】

図３９は、例示的なＷＴＲＵ１０２のシステム図である。図３９に示すように、ＷＴＲＵ１０２は、プロセッサ１１８、トランシーバ１２０、送受信要素１２２、スピーカ／マイクロフォン１２４、キーパッド１２６、ディスプレイ／タッチパッド１２８、非リムーバブルメモリ１３０、リムーバブルメモリ１３２、電源１３４、ＧＰＳチップセット１３６および他の周辺機能１３８を備えることができる。ＷＴＲＵ１０２は、実施形態との整合性を保ちながら、上述の要素のサブコンビネーションを含みうることが理解されよう。

【０４３４】

プロセッサ１１８は、汎用プロセッサ、特殊目的プロセッサ、従来のプロセッサ、デジタル信号プロセッサ（ＤＳＰ）、複数のマイクロプロセッサ、ＤＳＰコアと関連した１つまたは複数のマイクロプロセッサ、コントローラ、マイクロコントローラ、ＡＳＩＣ、ＦＰＧＡ回路、任意の他の種類の集積回路（ＩＣ）、状態機械等である。プロセッサ１１８は、信号の符号化、データ処理、電力制御、入出力処理、および／またはＷＴＲＵ１０２が無線環境で動作することを可能にする他の機能を行うことができる。プロセッサ１１８はトランシーバ１２０に結合することができ、トランシーバ１２０は送受信要素１２２に結合することができる。図３９ではプロセッサ１１８とトランシーバ１２０を別個の構成要素として示すが、プロセッサ１１８とトランシーバ１２０は電子パッケージやチップに共に一体化してよいことが理解されよう。

【０４３５】

送受信要素１２２は、エアインタフェース１１６を通じて基地局（例えば基地局１１４ａ）との間で信号を送信または受信するように構成することができる。例えば、一実施形態では、送受信要素１２２は、ＲＦ信号を送信および／または受信するように構成された

アンテナとすることができる。一実施形態では、送受信要素１２２は、例えばＩＲ、ＵＶ、または可視光信号を送信および／または受信するように構成されたエミッタ／検出器とすることができる。一実施形態では、送受信要素１２２は、ＲＦ信号と光信号の両方を送受信するように構成することができる。送受信要素１２２は、各種無線信号の任意の組合せを送信および／または受信するように構成してよいことが理解されよう。

【０４３６】

また、図３９では送受信要素１２２を１つの要素として示すが、ＷＴＲＵ１０２は任意数の送受信要素１２２を含んでよい。より具体的には、ＷＴＲＵ１０２はＭＩＭＯ技術を用いることができる。そのため、一実施形態では、ＷＴＲＵ１０２は、エアインタフェース１１６を通じて無線信号を送受信するために２つ以上の送受信要素１２２（例えば複数のアンテナ）を含むことができる。

【０４３７】

トランシーバ１２０は、送受信要素１２２から送信される信号を変調し、送受信要素１２２に受信された信号を復調するように構成することができる。上記のように、ＷＴＲＵ１０２はマルチモード機能を有することができる。そのため、トランシーバ１２０は、ＷＴＲＵ１０２が例えばＵＴＲＡやＩＥＥＥ８０２．１１等の複数のＲＡＴを介して通信することを可能にする複数のトランシーバを含むことができる。

【０４３８】

ＷＴＲＵ１０２のプロセッサ１１８は、スピーカ／マイクロフォン１２４、キーパッド１２６および／またはディスプレイ／タッチパッド１２８（例えば液晶ディスプレイ（ＬＣＤ）表示装置または有機発光ダイオード（ＯＬＥＤ）表示装置）に結合し、それらからユーザ入力を受け取ることができる。プロセッサ１１８は、スピーカ／マイクロフォン１２４、キーパッド１２６および／またはディスプレイ／タッチパッド１２８にユーザデータを出力することもできる。また、プロセッサ１１８は、非リムーバブルメモリ１３０および／またはリムーバブルメモリ１３２等の任意種の適切なメモリの情報にアクセスし、データを記憶することができる。非リムーバブルメモリ１３０には、ＲＡＭ、ＲＯＭ、ハードディスク、または他の任意種のメモリ記憶装置が含まれる。リムーバブルメモリ１３２には、ＳＩＭカード、メモリスティック、セキュアデジタル（ＳＤ）メモリカード等が含まれる。他の実施形態では、プロセッサ１１８は、サーバや家庭コンピュータ（図示せず）等、物理的にＷＴＲＵ１０２に位置しないメモリの情報にアクセスし、データを記憶することができる。

【０４３９】

プロセッサ１１８は、電源１３４から電力を受け取り、その電力をＷＴＲＵ１０２中の他の構成要素に分配および／または制御するように構成することができる。電源１３４は、ＷＴＲＵ１０２に電力を供給するのに適した任意の装置でよい。例えば、電源１３４は、１つまたは複数の乾電池（例えばニッケルカドミウム（ＮｉＣｄ）、ニッケル亜鉛（ＮｉＺｎ）、ニッケル水素（ＮｉＭＨ）、リチウムイオン（Ｌｉ－ｉｏｎ）等）、太陽電池、燃料電池等を含むことができる。

【０４４０】

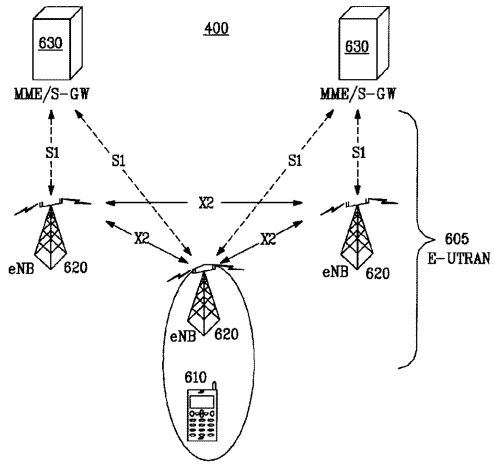
プロセッサ１１８は、ＧＰＳチップセット１３６にも結合することができ、ＧＰＳチップセット１３６は、ＷＴＲＵ１０２の現在地に関する位置情報（例えば経度および緯度）を提供するように構成することができる。ＧＰＳチップセット１３６からの情報に加えて、またはその代わりに、ＷＴＲＵ１０２は、基地局（例えば基地局１１４ａ、１１４ｂ）からエアインタフェース１１６を介して位置情報を受信し、かつ／または、２つ以上の近隣の基地局から信号が受信されるタイミングに基づいて自身の位置を判定することもできる。ＷＴＲＵ１０２は、実施形態との整合性を保ちながら、任意の適切な位置判定方法で位置情報を取得してよいことが理解されよう。

【０４４１】

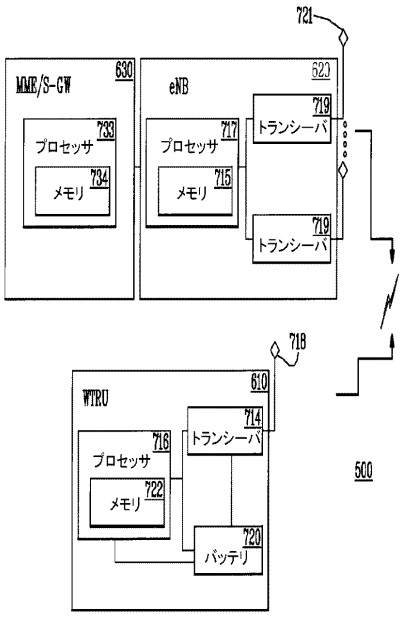
プロセッサ１１８はさらに、他の周辺機能１３８に結合することができ、それらには、追加的な機能、機能性、および／または有線若しくは無線接続を提供する１つまたは複数

のソフトウェアおよび／またはハードウェアモジュールが含まれる。例えば、周辺機能１３８には、加速度計、電子コンパス、衛星トランシーバ、デジタルカメラ（写真または映像用）、ＵＳＢポート、振動装置、テレビトランシーバ、ハンドフリーヘッドセット、Ｂｌｕｅｔｏｏｔｈモジュール、周波数変調（ＦＭ）無線ユニット、デジタル音楽プレーヤ、メディアプレーヤ、ビデオゲートウェイプレーヤモジュール、インターネットブラウザ等が含まれる。

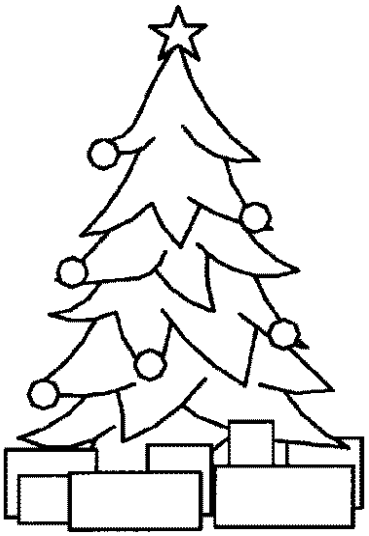
【図１】



【図２】



【図３】



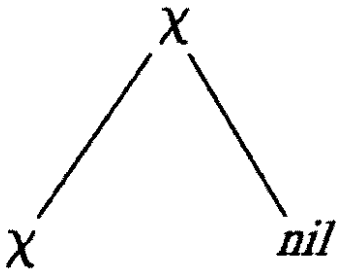
【図４】

アルゴリズム1 ツリー形成アルゴリズム

必要とする引数：V_1 ,..., V_d $\in \mathcal{V}$, $m \in \{0,1\}^{160}$
保証する内容： V_1 ,..., V_d = *nil*

```
1: n ← 0
2: while (m RoTM) ≠ nil do                ► 新しい測定結果を取得する
3:     m → SML                            ►空でない場合は、新しい葉として加える
4:     if (n)_d = 1 then                  ► 右からの値は
5:         V_d ← V_d ◇ m                  ► 深さd−1の根を拡張し、
6:         V_d → SML                      ► それをSMLにパージする
7:     else                               ► 左からの場合は
8:         V_d ← m                        ► 深さd−1の根に入れる
9:         goto line 22                   ► ただしそれ以上の
10:    end if                                深さにはしない
11:    k ← d - 1                          ► 右からの値である限り
12:    while (n)_k = 1 do                 ► 深さ2の部分木を更新する
13:        V_k ← V_k ◇ V_k+1
14:        V_k → SML
15:        k ← k - 1
16:    end while
17:    if k > 0 then
18:        V_k ← V_k+1
19:    else
20:        break while and return "tree full"
21:    end if
22:    n ← n + 1
23: end while
```

【図５】

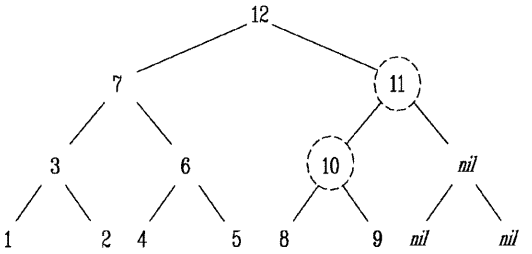


【図６】

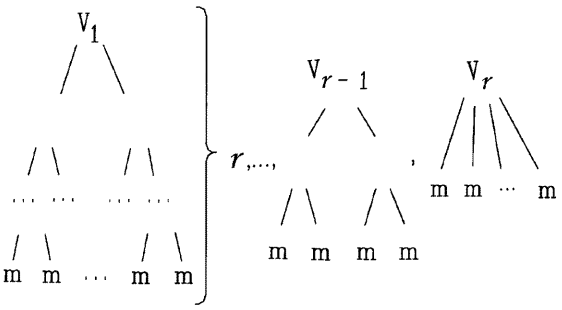
アルゴリズム2 不完全なツリーのクリーンアップ

```
24: for k ← k - 1 to 1 do
25:     if (n)_k = 1 then
26:         V_k ← V_k ◊ V_k+1
27:         V_k → SML
28:     else
29:         V_k ← V_k+1
30:         V_k → SML
31:     end if
32: end for
```

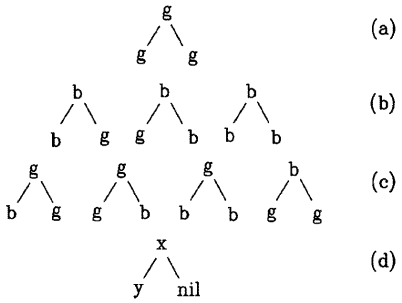
【図７】



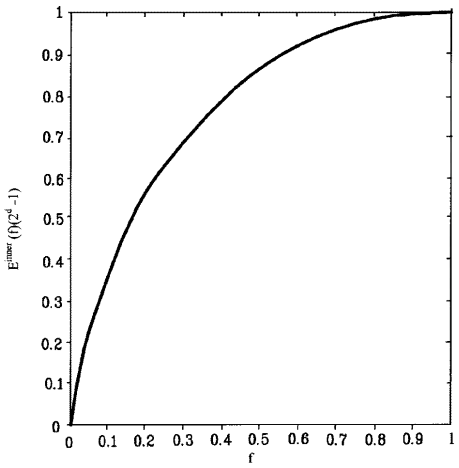
【図８】



【図９】



【図１０】



【図１１】

【図１２】

アルゴリズム1：線形にリンクされたハッシュチェーン内で最初の破綻点を見つける

1. 検査対象の測定チェーンの開始インデックスをiとし、
最後のインデックスをjとする。したがって、開始時はi=0, j=n−1となる。

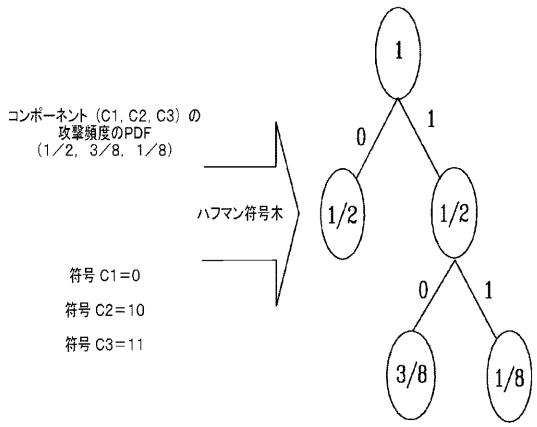
2. i=jの場合はステップ5に進む。それ以外の場合は、
V_0および測定結果 $\{m_k; k=0, 1, \ldots, (J+i)/2\}$ を使用して$V_{(j+1)}/2$を計算し、
対応する参照測定値$R_{(j+i)}/2$と比較する。

3. それらが等しい場合は、値$V_i \sim V_{(j+1)}/2$がgoodであることが分かる。
i=1+((j+i)/2) に設定し、ステップ2に戻る。

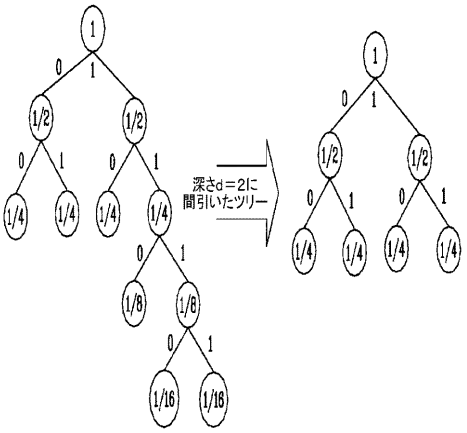
4. 等しくない場合は、値、値$V_i \sim V_{(j+1)}/2$の中に欠陥があることが分かる。
j=(j+i)/2に設定し、ステップ2に戻る。

5. インデックスiは、線形チェーン中のV_iとその参照値R_iとの最初の不一致点である。

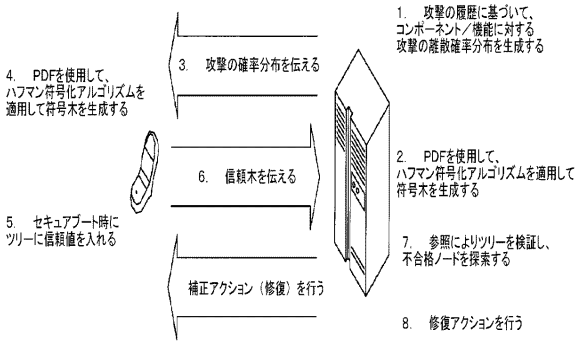
【図１３】



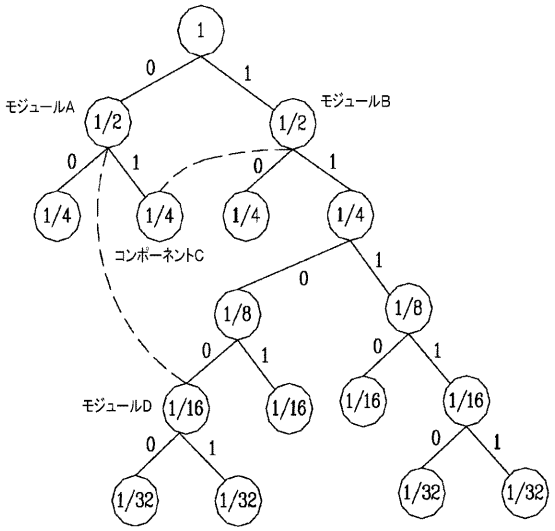
【図１４】



【図１５】



【図１６】



【図１７】

アルゴリズム2：2分木への判断基準の投入

```
1. Populate_metric (Node)
2. {
3.       If (Node is leaf)
4.            Return H(component);
5.
6.       left_metric = Populate_metric(Node->Left Child);
7.       right_metric = Populate_metric(Node->Right Child);
8.
9.       Node.metric = F(left_metric, right_metric);
10.
11.      Return Node_metric;
12. }
```

【図１８】

アルゴリズム3：TPMコマンドを使用したツリーへの判断基準の投入

```
1. Reg_Number = 0
2. Populate_metric (Node)
3. {
4.    If (Node is leaf)
5.    {
6.       Node.metric = H(component);
7.       PCR_Register Reg_Number = Node.metric;
8.       Reg_Number++;
9.       Return;
10.   }
11.
12.   Populate_metric(Node->Left Child);
13.   Populate_metric(Node->Right Child);
14.   Reg_Number--;
15.
16.   PCR_Register Reg_Number-1  =PCR_Extend(PCR_Register Reg_Number, PCR_Register Reg_Number-1 );
17.   Node.metric = PCR_Register Reg_Number-1 ;
18.
19.   Return;
20.}
```

【図１９】

アルゴリズム4：n分木への判断基準の投入

```
1. Populate_metric (Node)
2. {
3.    If (Node is leaf)
4.       Return H(component);
5.    Metric_0 = Populate_metric(Node->Child_0);
6.    for (i=1; i<n; i++)
7.    {
8.       Metric_i = Populate_metric(Node->Child_i);
9.       Node.metric = F(Metric_i, Metric_(i-1));
10.   }
11.
12.   Return Node_metric;
13.}
```

【図２０】

アルゴリズム5：代理子リンクを含む2分木への判断基準の投入

```
1. Populate_metric (Node)
2. {
3.    If (Node is leaf)
4.       Return H(component);
5.
6.    left_metric = Populate_metric(Node->Left Child);
7.    right_metric = Populate_metric(Node->Right Child);
8.
9.    Node.metric = F(left_metric, right_metric);
10.
11.   If (surrogate_link_present)
12.     Surrogate_metric = Populate_metric(Node->Surrogate Child);
13.
14.   Node.metric = F(Node.metric, Surrogate_metric);
15.
16.   Return Node_metric;
17. }
```

【図２１】

アルゴリズム6：完全性検査に合格しないノードおよび葉を見つける比較および間引きのアルゴリズム

1. 比較は根から開始する。根の判断基準が一致する場合は、
すべてのノードと葉は損なわれておらず、変更されていない。アルゴリズムを終了する。
2. 判断基準が一致しない場合は、以下のステップに進む。
3. j=1とする。
4. 深さjのすべてのノードを比較する。
5. 深さjの特定のノードn_{ij}が一致する場合は、そのノードおよびノードn_{ij}より下の部分木を間引く。
深さjの特定のノードn_{ij}が一致しない場合は、j=j+1を設定することにより、
深さjのノードの子を比較し、ステップ3以降を繰り返し実行する。
6. ツリーの全深さに到達するまでそれらのステップを実行する。
7. ツリーの残りの部分は、一致しないノードまたは葉の測定結果
（完全性が損なわれたノードまたは葉を意味するものと考えられる）を有する
ノードおよび葉のみを含んでいる。

【図２２】

アルゴリズム1　TPM_Reduced_Tree_Verify_Load

必要とする引数：B_i,..., B_ℓ, $V^* \in \mathcal{V}$, ((n),n, R(n))
保証する内容：$B_i \leftarrow r_i$,..., $B_\ell \leftarrow r_\ell$, $V^* \leftarrow n$
▷ 縮小ツリーと検証対象のノードでバッファを初期化する

```
1. for k = ℓ,..., 1 do
2.    V* ⟶ TSS
3.    V* ⟵ (B_k|(n)_k|V*)
4. end for
5. if V ≡ V* then
6.    return V
7. else
8.    return "verification error"
9. end if
```

【図２３】

アルゴリズム2　TPM_Reduced_Tree_Verify

必要とする引数：$B \in \mathcal{V}$, ((n),n, R(n))
保証する内容：$B \leftarrow n$　▷ 検証対象のノードでバッファを初期化する

```
1. for k = ℓ,..., 1 do
2.    B ⟶ TSS
3.    B ⟵ (r_k|(n)_k|B)
4. end for
5. if V ≡ B then
6.    return V
7. else
8.    return "verification error"
9. end if
```

【図２４ａ】

アルゴリズム3　TPM_Tree_Node_Verify

必要とする引数：$B, C, D \in \mathcal{V}$, ((n), R(n), T(n))
保証する内容：$C \leftarrow V$　▷ 比較レジスタを根で初期化する

```
1. for k = 1 ,..., ℓ do
2.    B, D ⟵ t_k                ▷ トレースの子をバッファにロードする
3.    B ⟵ (r_k|(n)_k|B)
4.    if C ≡ B then
      ▷ 結果を現在の親と比較する。OKの場合は
5.       C ⟵ D
      ▷ 検証されたトレース要素を新しい親にする
6.    else
7.       return "verification error at level" || k, B
8.    end if
9. end for
10. return "OK"
```

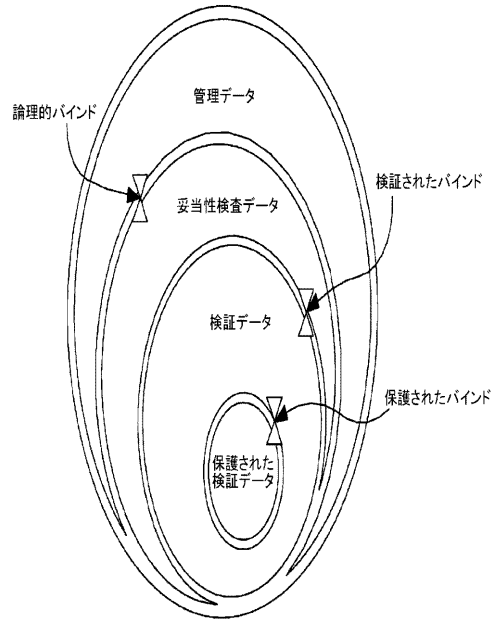
【図２４ｂ】

アルゴリズム4　TPM_Reduced_Tree_Update

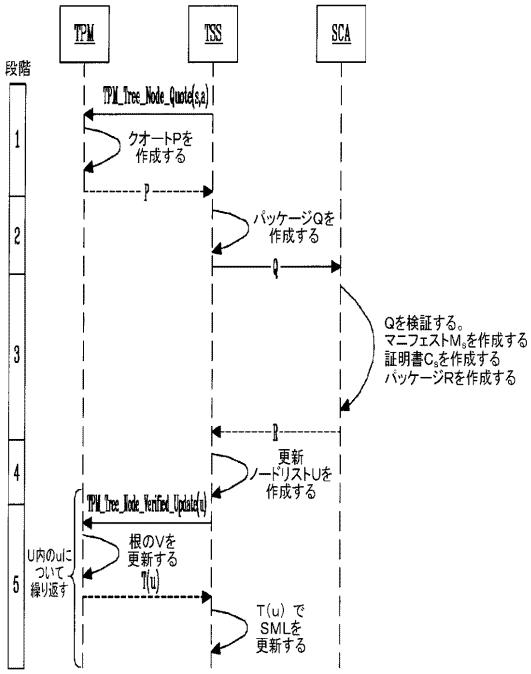
必要とする引数：(n), n'
保証する内容：$B_1 = r_1$,..., $B_\ell = r_\ell$

```
1. V ⟵ n'
2. for k = ℓ,..., 1 do
3.    V ⟶ TSS
4.    V ⟵ (B_k|⟨n⟩_k|V)
5. end for
6. return V
```

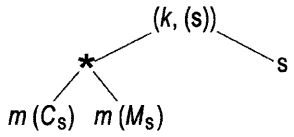
【図２５】



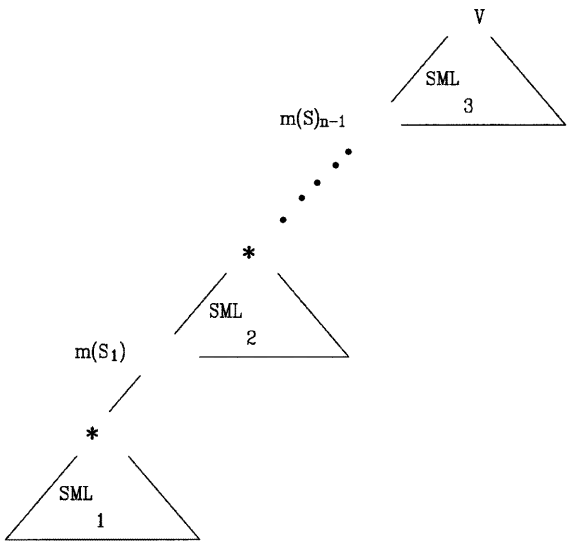
【図２６】



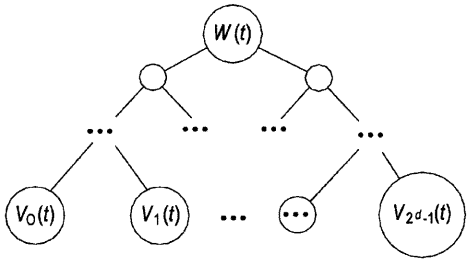
【図２７】



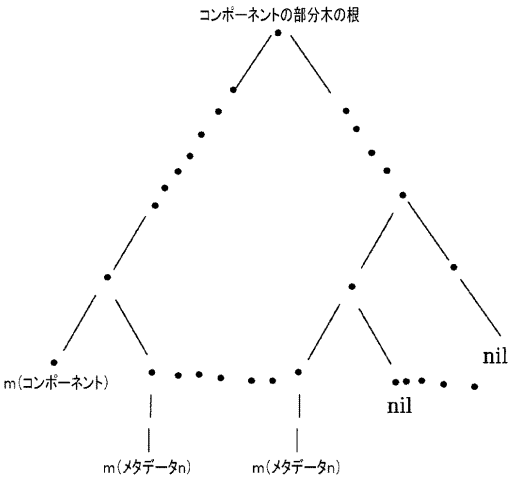
【図２８】



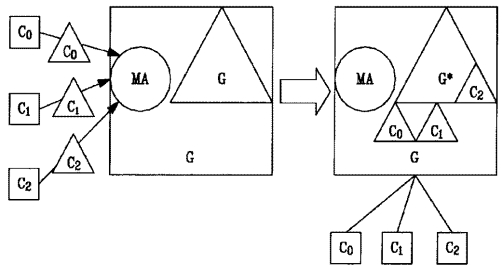
【図２９】



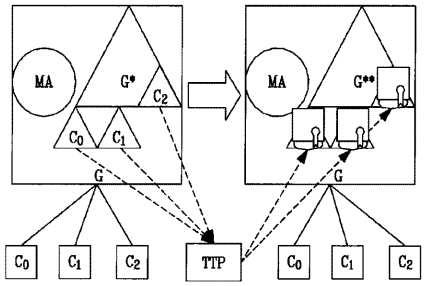
【図３０】



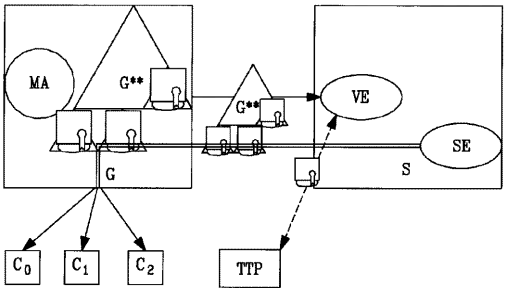
【図３１】



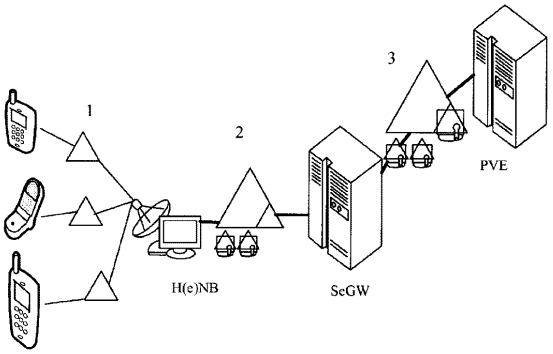
【図３２】



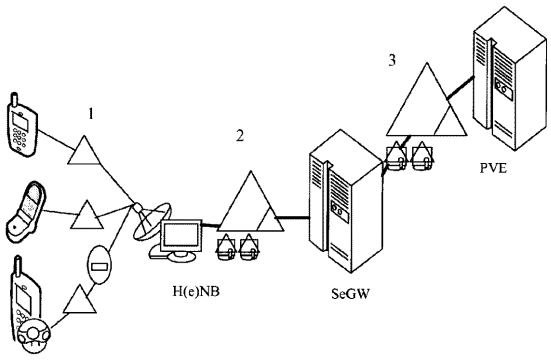
【図３３】



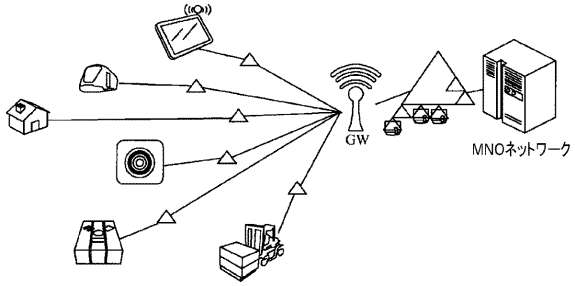
【図３４】



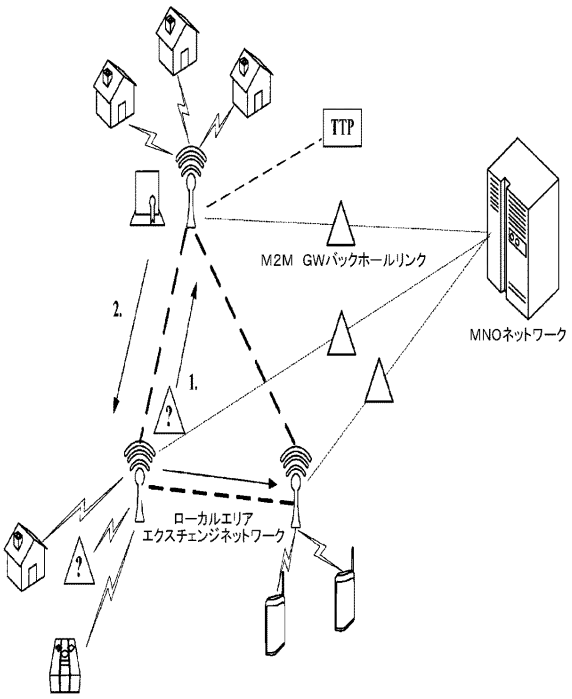
【図３５】



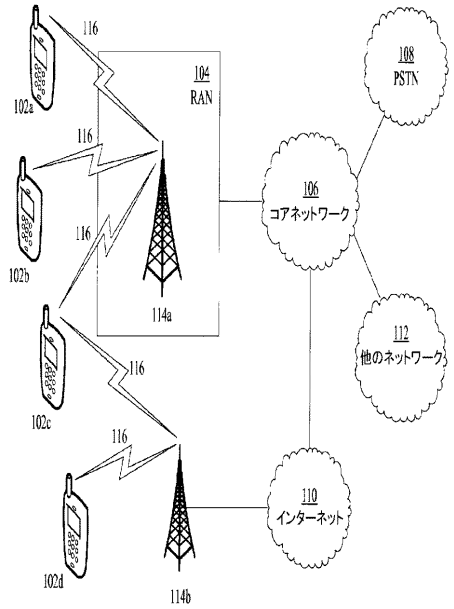
【図３６】



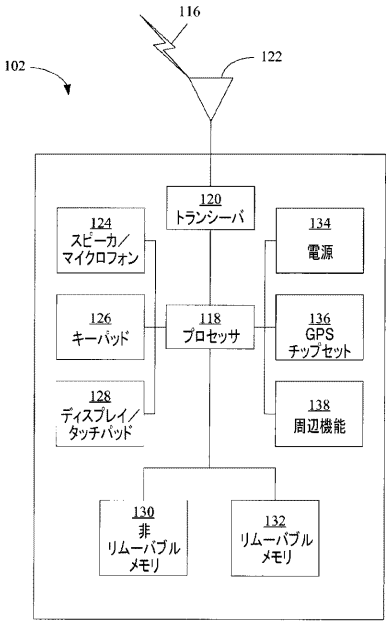
【図３７】



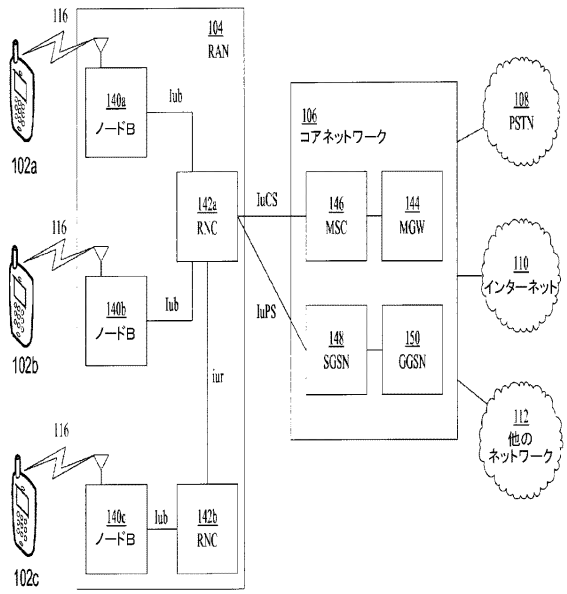
【図３８】


102a
102b
102c
102d
116
104
RAN
114a
114b
106
コアネットワーク
108
PSTN
112
他のネットワーク
110
インターネット

【図３９】


116
102
122
120
トランシーバ
124
スピーカ／
マイクロフォン
126
キーパッド
128
ディスプレイ／
タッチパッド
118
プロセッサ
134
電源
136
GPS
チップセット
138
周辺機能
130
非
リムーバブル
メモリ
132
リムーバブル
メモリ

【図４０】


102a
102b
102c
116
104
RAN
140a
ノードB
140b
ノードB
140c
ノードB
Iub
Iub
Iub
Iur
142a
RNC
142b
RNC
IuCS
IuPS
106
コアネットワーク
146
MSC
144
MGW
148
SGSN
150
GGSN
108
PSTN
110
インターネット
112
他の
ネットワーク

フロントページの続き

(72)発明者 インヒョク チャ
大韓民国 ソウル カンナムーグ サムスンードン １４－１ ジュンーアン ハイツ ビレッジ １０２－ドン ２０２－ホ
(72)発明者 サディール ビー. パッター
アメリカ合衆国 ０８０５４ ニュージャージー州 マウント ローレル アン ドライブ １７
(72)発明者 ヨゲンドラ シー. シャー
アメリカ合衆国 １９３４１ ペンシルベニア州 エクストン リージェンシー コート １０

審査官 中里 裕正

(56)参考文献 米国特許出願公開第２０１０／０１６１９９８（ＵＳ，Ａ１）
米国特許出願公開第２００８／０２８８７８３（ＵＳ，Ａ１）
中国特許出願公開第１０１３５００４４（ＣＮ，Ａ）
特開平７－３１２５９２（ＪＰ，Ａ）
Williams, D. and Sirer, E. G., Optimal parameter selection for efficient memory integrity verification using Merkle hash trees, Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications, ２００４年, p.383-388

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ２１／５７
Ｈ０４Ｗ １２／００
ＪＳＴＰｌｕｓ／ＪＭＥＤＰｌｕｓ／ＪＳＴ７５８０（ＪＤｒｅａｍＩＩＩ）
ＩＥＥＥ Ｘｐｌｏｒｅ