

(19)日本国特許庁(JP)

(12)公表特許公報(A)

(11)公表番号
特表2024-527556
(P2024-527556A)

(43)公表日 令和6年7月25日(2024.7.25)

(51)国際特許分類

F I

テーマコード (参考)

H 0 4 L 9/32 (2006.01) H 0 4 L 9/32 2 0 0 A 5 B 1 7 5

G 0 6 F 16/28 (2019.01) H 0 4 L 9/32 2 0 0 E

G 0 6 F 16/28

審査請求 未請求 予備審査請求 未請求 (全49頁)

(21)出願番号	特願2023-580769(P2023-580769)	(71)出願人	590000248
(86)(22)出願日	令和4年7月1日(2022.7.1)		コーニンクレッカ フィリップス エヌ
(85)翻訳文提出日	令和5年12月28日(2023.12.28)		ヴェ
(86)国際出願番号	PCT/EP2022/068316		Koninklijke Philips
(87)国際公開番号	WO2023/280721		N. V.
(87)国際公開日	令和5年1月12日(2023.1.12)		オランダ国 5 6 5 6 アーヘー アイン
(31)優先権主張番号	63/218,525		ドーフエン ハイテック キャンパス 5 2
(32)優先日	令和3年7月6日(2021.7.6)		High Tech Campus 52,
(33)優先権主張国・地域又は機関			5 6 5 6 AG Eindhoven, N
	米国(US)		etherlands
(31)優先権主張番号	21207844.8	(74)代理人	110001690
(32)優先日	令和3年11月11日(2021.11.11)		弁理士法人M&Sパートナーズ
(33)優先権主張国・地域又は機関		(72)発明者	ガルシア モーション オスカー
	欧州特許庁(EP)		オランダ国 5 6 5 6 アーヘー アイン
(81)指定国・地域	AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA		ドーフエン ハイ テック キャンパス 5
	最終頁に続く		最終頁に続く

(54)【発明の名称】 完全性保護のためのハッシュツリーを使用した階層データ構造中のデータの符号化

(57)【要約】

いくつかの実施形態はデータ構造を対象とする。データ構造は複数のブロックと第1のハッシュツリーの一部を含む。ハッシュツリーは複数のゲノムブロックの複数のハッシュ値から計算される。第1のハッシュツリーの一部は、第1のハッシュツリーの少なくとも最初の2つの最も高いレベルを含むが、第1のハッシュツリーの1つ又は複数のより低いレベルを含まない。

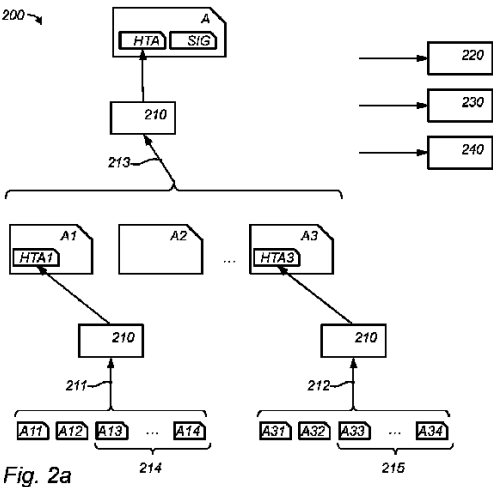


Fig. 2a

【特許請求の範囲】

【請求項 1】

データ構造中のデータを符号化するための符号化システムであって、前記符号化システムは、

前記データを受信するための入力インターフェースと、

プロセッサシステムとを備え、前記プロセッサシステムが、

前記データを複数のデータブロックとして取得することと、

前記複数のデータブロックにハッシュ関数を適用することによって、前記複数のデータブロックのための複数のハッシュ値を計算することと、

前記複数のハッシュ値についての第 1 のハッシュツリーを計算することであって、前記複数のハッシュ値が前記第 1 のハッシュツリーのリーフに割り当てられ、前記第 1 のハッシュツリーの 1 つ又は複数のより高いレベルが生成される、第 1 のハッシュツリーを計算することと、

前記データ構造中に前記複数のデータブロックと前記第 1 のハッシュツリーの全部又は一部とを含めることであって、前記第 1 のハッシュツリーの前記一部が前記第 1 のハッシュツリーの前記リーフの少なくとも一部を含まない、前記データ構造中に前記複数のデータブロックと前記第 1 のハッシュツリーの全部又は一部とを含めることとを行う、符号化システム。

【請求項 2】

前記プロセッサシステムが第 2 のハッシュツリーを計算することであって、前記第 1 のハッシュツリーのルートが前記第 2 のハッシュツリーのリーフに割り当てられ、複数のさらなるハッシュ値が前記第 2 のハッシュツリーの複数のさらなるリーフに割り当てられ、さらなるハッシュ値が、さらなるハッシュツリーのルートとして生成される及び／又はさらなるデータブロック上でのハッシングによって生成される、第 2 のハッシュツリーを計算することと、前記プロセッサシステムが前記データ構造中に前記第 2 のハッシュツリーの少なくとも前記ルートを含めることとを行う、請求項 1 に記載の符号化システム。

【請求項 3】

前記データ構造が、コンテナの階層中に編成されたデータブロックを含み、前記プロセッサシステムが前記階層中の各コンテナのためのハッシュツリーを計算し、ハッシュツリーの各リーフが、前記コンテナ内のデータブロックの前記ハッシュ値、又は下位のコンテナに対応するハッシュツリーの前記ルートのいずれかである、請求項 1 又は 2 に記載の符号化システム。

【請求項 4】

前記プロセッサシステムが、ハッシュツリーのルート、特に、ハッシュツリーのリーフの間のハッシュツリールートに有するハッシュツリーの前記ルートにわたるデジタル署名を計算することと、前記デジタル署名を前記データ構造中に含めることとを行う、請求項 1 から 3 のいずれか一項に記載の符号化システム。

【請求項 5】

前記プロセッサシステムが、前記データ構造を記憶すること及び／又は前記データ構造若しくは前記データ構造の一部をストリーミングすることを行い、前記データ構造の前記一部が、前記データブロックの少なくとも一部と、前記データブロックに対応するハッシュツリーの少なくとも一部とを含む、請求項 1 から 4 のいずれか一項に記載の符号化システム。

【請求項 6】

前記複数のデータブロック及び／又はデータコンテナのサブセットが完全性保護付きと標示され、前記複数のデータブロック及び／又はデータコンテナの残りが、完全性保護なしと標示され、完全性保護付きと標示された部分のみがハッシュツリー中に含まれる、請求項 1 から 5 のいずれか一項に記載の符号化システム。

【請求項 7】

前記入力インターフェースが 1 つ又は複数のツリーパラメータを受信し、前記プロセッ

10

20

30

40

50

システムが、前記 1 つ又は複数のツリーパラメータに応じて、1 つのハッシュツリー又はハッシュツリーの階層から選択されたノードのセットを前記データ構造中に含める、請求項 1 から 6 のいずれか一項に記載の符号化システム。

【請求項 8】

前記入力インターフェースが前記データに対する修正を受信し、前記修正が、追加、削除、及び／又は変更のうちの 1 つ又は複数を含み、前記プロセッサシステムが、前記修正を適用することと、前記データの修正された前記一部に対応するハッシュツリーの一部を選択的に再計算し、更新することとを行う、請求項 1 から 7 のいずれか一項に記載の符号化システム。

【請求項 9】

ハッシュツリーのリーフが、非圧縮形態のデータブロックのハッシュと、圧縮形態の前記データブロックのハッシュとをさらに含み、前記データブロックが圧縮形態で前記データ構造中に含まれるか、又は、

ハッシュツリーのリーフが、非圧縮及び非暗号化形態のデータブロックのハッシュと、圧縮及び非暗号化形態の前記データブロックのハッシュと、圧縮及び暗号化形態のデータブロックのハッシュとをさらに含み、前記データブロックが圧縮及び暗号化形態で前記データ構造中に含まれる、

請求項 1 から 8 のいずれか一項に記載の符号化システム。

【請求項 10】

データ構造中の選択されたデータを検証するための検証システムであって、前記検証システムは、

前記データ構造の少なくとも一部を受信するための入力インターフェースであって、前記データ構造が、複数のデータブロックとハッシュツリーの一部とを含み、第 1 のハッシュツリーのリーフの少なくとも一部を含まない、入力インターフェースと、

プロセッサシステムとを備え、前記プロセッサシステムが、

データ完全性検証のために選択された前記データ構造中のデータブロックにハッシュ関数を適用することによって、選択された前記データブロックのための複数のハッシュ値を計算することと、

検証のために選択されたデータブロックから始まる、対応するハッシュツリーのルートまでの経路を特定することと、

前記データ構造中で利用可能な場合に、前記経路に沿った前記ハッシュツリーのための前記ハッシュ値を取り出すこと、又はそのような前記ハッシュ値を計算することと、

少なくとも計算された前記複数のハッシュ値から前記ハッシュツリーの前記ルートを検証することと

を行う、検証システム。

【請求項 11】

前記データ構造がハッシュツリーの階層を含み、前記プロセッサシステムが、

リーフから始まる、ハッシュツリーの前記階層のルート全体までの経路を特定する、請求項 10 に記載の検証システム。

【請求項 12】

前記符号化及び／又は検証システムがデバイスである、請求項 1 から 11 のいずれか一項に記載の符号化及び／又は検証システム。

【請求項 13】

前記複数のデータブロックがゲノムデータを含む、請求項 1 から 12 のいずれか一項に記載の符号化及び／又は検証システム。

【請求項 14】

データ構造中のデータを符号化するための符号化方法であって、前記符号化方法は、

前記データを複数のデータブロックとして取得するステップと、

前記複数のデータブロックにハッシュ関数を適用することによって、前記複数のデータブロックのための複数のハッシュ値を計算するステップと、

10

20

30

40

50

前記複数のハッシュ値についての第1のハッシュツリーを計算するステップであって、前記複数のハッシュ値が前記第1のハッシュツリーのリーフに割り当てられ、前記第1のハッシュツリーの1つ又は複数のより高いレベルが生成される、第1のハッシュツリーを計算するステップと、

前記データ構造中に前記複数のデータブロックと前記第1のハッシュツリーの一部とを含めるステップであって、前記第1のハッシュツリーの前記一部が前記第1のハッシュツリーの前記リーフの少なくとも一部を含まない、前記データ構造中に前記複数のデータブロックと前記第1のハッシュツリーの一部とを含めるステップとを有する、符号化方法。

【請求項15】

データ構造中の選択されたデータを検証するための検証方法であって、前記検証方法は、

前記データ構造の少なくとも一部を受信するステップであって、前記データ構造が、複数のデータブロックとハッシュツリーの一部とを含み、第1のハッシュツリーのリーフの少なくとも一部を含まない、前記データ構造の少なくとも一部を受信するステップと、

データ完全性検証のために選択された前記データ構造中のデータブロックにハッシュ関数を適用することによって、選択された前記データブロックのための複数のハッシュ値を計算するステップと、

検証のために選択されたデータブロックから始まる、対応するハッシュツリーのルートまでの経路を特定するステップと、

前記データ構造中で利用可能な場合に、前記経路に沿った前記ハッシュツリーのための前記ハッシュ値を取り出すステップ、又はそのような前記ハッシュ値を計算するステップと、

少なくとも計算された前記複数のハッシュ値から前記ハッシュツリーの前記ルートを検証するステップとを有する、検証方法。

【請求項16】

前記データ構造がハッシュツリーの階層を含み、前記検証方法は、リーフから始まる、ハッシュツリーの前記階層のルート全体までの経路を特定するステップを有する、請求項15に記載の検証方法。

【請求項17】

プロセッサシステムによって実行されたときに、前記プロセッサシステムに請求項14及び／又は15に記載の方法を実行させる命令を表すデータを含む、一時的又は非一時的コンピュータ可読媒体。

【発明の詳細な説明】

【技術分野】

【0001】

開示されている本主題は、デジタルデータ構造中のゲノムデータを符号化するための符号化システムと、デジタルデータ構造中の選択されたゲノムデータを検証するための検証システムと、デジタルデータ構造中のゲノムデータを符号化するための符号化方法と、デジタルデータ構造中の選択されたゲノムデータを検証するための検証方法と、コンピュータ可読媒体とに関する。

【背景技術】

【0002】

ゲノムデータの量は絶えず増加しているので、そのような情報が適切なデータ構造中に記憶されることが重要である。参照によって本明細書中に含まれるISO/IEC 23092は、ゲノムデータを符号化し、圧縮し、保護するための規格を定義している。特に、同様に参照によって本明細書中に含まれるISO/IEC DIS 23092-1、「Information technology-Genomic information representation-Part 1: Transport and st

10

20

30

40

50

orage of genomic information」は、ゲノム情報を記憶及び／又はストリーミングするためのデータ構造を定義している。

【0003】

知られているデータ構造は、ゲノムデータ、例えばシーケンスデータを記憶し、そのゲノムデータに関係する他の情報に関連付けることができる階層データ構造を開示している。例えば、ISO/IEC DIS 23092-1の表4はフォーマット構造と階層カプセル化レベルとを開示している。表4は、様々なタイプのデータのためのボックスと、それらの可能な格納とを示している。

【0004】

ゲノムデータについてのファイルは非常に大きくなり得る。サイズは、数百ギガバイト、又はテラバイトにもなり得る。従来の完全性測定は、そのように多くのデータにわたって計算するのに長い時間がかかる。

【発明の概要】

【発明が解決しようとする課題】

【0005】

より良い完全性制御を可能にする、ゲノムデータのための改善されたデータ構造を有することが有利である。例えば、上述のISO/IEC 23092は、特にファイル全体について、ファイルの完全性の証拠を効率的に与えるすべてのデータ構造と一緒にグループ化するやり方について説明していない。また、完全性を考慮に入れながら、データ構造をファイルに追加すること又はファイルから削除すること、或いはゲノムファイルを更新することは実現不可能である。ファイルがどのように更新されるか、又はそれらの変更について誰が責任を負うのかを追跡することはない。ゲノムデータは、ユーザのヘルスケアデータだけでなく、彼／彼女の子孫のヘルスケアデータにも関係するので、長期間にわたってゲノムデータの完全性を保護することが有利である。選択的なデータ構成要素上の個々のデジタル署名を使用することは、この目的のためには十分でない。例えば、そのことはデータ構造間の関係を保護しない。攻撃者は、構成要素を削除すること、又はそれらの順序を変更することがある。これらの問題のいずれも個々に対処するに値する。他の問題について、本明細書で特定し、対処する。

【課題を解決するための手段】

【0006】

いくつかの実施形態はデジタルデータ構造を対象とする。データ構造は複数のゲノムブロックと第1のハッシュツリーの一部とを含む。ハッシュツリーは複数のゲノムブロックの複数のハッシュ値から計算される。第1のハッシュツリーの含められた一部はノードの選択されたサブセットを含み、ノードの選択されたサブセットは、ノードの最も高い1つ又は複数のレベルと、第1のハッシュツリーの選択された数のリーフとの組合せであり得る。第1のハッシュツリーは、データ構造中に現れる第1のハッシュツリーである必要はないことを理解されたい。

【0007】

ゲノムデータは、一般に大きく、階層的でもあるので、特に有利な適用例である。しかしながら、実施形態は、任意のタイプのデータ、特に階層的に編成されたデータに適用することができる。多くの実施形態についてゲノムデータのコンテキストにおいて説明するが、本発明はゲノムデータに限定されない。

【0008】

一般に、ハッシュツリー又はマークルツリー(Merkle Tree)は、リーフノードの各々が、データのデータブロックのハッシュ、又は下位コンテナのハッシュツリーのルートを含み、非リーフノードが、次に低いレベルにあるノードにわたるハッシュを含み、後者のノードがリーフノード又は非リーフノードであり得る、ツリー構造である。ハッシュツリーの特種なタイプはVerkleeツリーであり、内部ノード(非リーフノード)をその子ノードから計算するために使用されるハッシュ関数は正則ハッシュではなく、ベクトルコミットメントである。実施形態は、正則ハッシュ、特に、SHAファミリー

10

20

30

40

50

、例えばSHA-3など、マークルダンガード(Merkle-Damgard)タイプのハッシュ関数(MD-タイプ)を使用することがある。Verkleタイプツリーのリーフは、例えばMD-タイプの正則ハッシュ関数を使用する。Verkleツリーのさらなる説明は、Kuszmahlによる文書「Verkle Trees」、Technical report; Massachusetts Institute of Technology: Cambridge, MA, USA、2018中に見つけることができる。複数のゲノムデータブロックが、例えばゲノムデータのパーティションとして受信されていることがある。例えば、一実施形態では、ゲノムシーケンス及び／又は他のゲノムデータなど、ゲノムデータが受信される。ゲノムデータは、すでにブロックに区分されていることもあり、符号化システムによってブロックに分割されることもある。

10

【0009】

トップレベル部分を含めることによって、迅速な検証又は更新が維持されるが、より低いレベル部分を含めないことによって、ストレージサイズが低減される。含められなかったより低いレベル部分は、リーフの一部、すべてのリーフ、又はいくつかのより低いレベルでさえあり得る。

【0010】

一実施形態では、データ構造は階層データ構造である。より高いレベルにあるブロックは、より低いレベルにある複数のブロックを指すことがある。より高いレベルのブロックは、より低いレベルにわたって計算されたハッシュツリーの一部を含み得る。このことは2回以上起こり得る。例えば、第1のレベルは、第2のより低いレベルにあるブロックにわたって計算されたハッシュツリーの一部を含む。第2のレベルは、第3のより低いレベルにあるブロックにわたって計算されたハッシュツリーの一部を含み、以下同様である。第1のレベルは、第2のより低いレベルにある他のブロックにわたって計算された別のハッシュツリーの一部を含み得る。

20

【0011】

データ構造又はその一部は、記憶され、取り出され、ストリーミングされ、受信され、符号化され、検証される。データ構造をストリーミングするときに、含められなかった一部が再計算され、ストリーミング中に含まれることがある。データ構造をストリーミングするとき、ストリーミングは、選択されたゲノムデータブロックと、ストリーミングされないデータにアクセスすることなしにハッシュツリーのルートを再計算するために必要とされるハッシュツリーの一部とのみを含み得る。

30

【0012】

一実施形態では、ハッシュツリーが全部記憶されない。このことはまた、複数のレベルにわたって起こり得る。興味深いことに、第1のレベルは第2のより低いレベルのための部分的なハッシュツリーを含み、第2のレベルも、より低い第3のレベルのための部分的なハッシュツリーを含んでいる。部分的に含められたハッシュツリーの階層が構築される。一実施形態では、ツリーパラメータが受信される。含められているハッシュツリーの一部のサイズはツリーパラメータから決定される。例えば、ツリーパラメータは、含めるべきレベルの数である。一実施形態では、ツリーパラメータは2以上である。

【0013】

ハッシュツリーの他の態様、例えば、ノードごとの子ノードの数、すなわちk進性(k-aryness)もツリーパラメータによって設定される。

40

【0014】

一態様は、例えば、符号化システムの一実施形態によって符号化される、データ構造中の選択されたデータを検証するための検証システムである。検証は、ハッシュツリーのルートを検証することによって行われる。これは、一般に、ルートを再計算することによって行われる。とはいえ、より高度のタイプのハッシュツリー、例えばVerkleツリーの場合、非対称アルゴリズムが使用される。ハッシュを再計算するか、さもなければ検証するために、ハッシュツリーの一部がデータ構造から取り出されるか、又は再計算されるかのいずれかである。例えば、リーフ値はないことがあり、再計算されることがある。よ

50

り低いレベル（又はその一部）のうちのいくつかは、データ構造中にないこともあり、再計算されることもある。しかしながら、いくつかのハッシュ値は、データ構造中に存在することがあり、取り出すことができる。どの値が必要とされるかを決定するために、検証のために選択されたデータブロックから始まる、対応するハッシュツリーのルートまでの経路が特定される。ルートの事前計算、及び／又は検証のために必要とされる、経路に沿ったハッシュツリーのためのハッシュ値は、経路上のノードの値を含むだけでなく、経路上のノードの子ノードをも含む。興味深いことに、ルートは、検証されるデータブロックに直接関連付けられたハッシュツリー、例えば、そのリーフ中のデータブロックのハッシュ値を含むハッシュツリー、例えば第1のハッシュツリーのルートであり得るが、代わりに（又は同様に）階層的により高いハッシュツリー、例えば第2のハッシュツリーのルートであり得る。ルートが再計算されると、それをデータ構造中のルートと比較することができる。他のタイプの検証は、ルートにわたって署名を検証すること、又はVerkshetツリーの場合と同様に、ベクトルコミットメント検証を実行することを含む。

10

【0015】

符号化システムは、場合によっては、1つ又は複数の電子デバイス中に実装される電子システムであり得る。検証システムは、場合によっては、1つ又は複数の電子デバイス中に実装される電子システムであり得る。

【0016】

さらなる態様は符号化方法及び検証方法である。本方法の一実施形態は、コンピュータ実装方法としてコンピュータ上に実装されるか、又は専用のハードウェアにおいて、又は両方の組合せにおいて実装される。本方法の一実施形態のための実行可能コードはコンピュータプログラム製品上に記憶される。コンピュータプログラム製品の例としては、メモリデバイス、光記憶デバイス、集積回路、サーバ、オンラインソフトウェアなどがある。好ましくは、コンピュータプログラム製品は、前記プログラム製品がコンピュータ上で実行されたときに、本方法の一実施形態を実行するためのコンピュータ可読媒体上に記憶された非一時的プログラムコードを含む。

20

【0017】

一実施形態では、コンピュータプログラムは、そのコンピュータプログラムがコンピュータ上で実行されたときに本方法の一実施形態のステップの全部又は一部を実行するように適応されたコンピュータプログラムコードを含む。好ましくは、コンピュータプログラムはコンピュータ可読媒体上で実施される。

30

【0018】

さらなる詳細、態様、及び実施形態について、単に例として、図面を参照しながら説明する。図中の要素は、簡単及び明快のために示されており、必ずしも一定の縮尺で描かれているとは限らない。図において、すでに説明した要素に対応する要素は同じ参照番号を有することがある。

【図面の簡単な説明】

【0019】

【図1a】符号化システムの一実施形態の一例を概略的に示す図である。

【図1b】検証システムの一実施形態の一例を概略的に示す図である。

40

【図2a】記憶のための符号化されたデータフォーマットの一実施形態の一例を概略的に示す図である。

【図2b】記憶のための符号化されたデータフォーマットの一実施形態の一例を概略的に示す図である。

【図2c】記憶のための符号化されたデータフォーマットの一実施形態の一例を概略的に示す図である。

【図3】階層ハッシュツリーの一実施形態の一例を概略的に示す図である。

【図4】ハッシュツリーの一実施形態の一例を概略的に示す図である。

【図5a】ハッシュツリーの一実施形態の一例を概略的に示す図である。

【図5b】ハッシュツリーの一実施形態の一例を概略的に示す図である。

50

【図 6】ストリーミング又はトランスポートのための符号化されたメッセージの一実施形態の一例を概略的に示す図である。

【図 7 a】符号化方法の一実施形態の一例を概略的に示す図である。

【図 7 b】検証方法の一実施形態の一例を概略的に示す図である。

【図 8 a】一実施形態による、コンピュータプログラムを含む書込み可能部分を有するコンピュータ可読媒体を概略的に示す図である。

【図 8 b】一実施形態による、プロセッサシステムの表現を概略的に示す図である。

【0020】

参照符号リスト

参照符号及び省略形の以下のリストは図 1 a から図 3 に対応し、図面の解釈を容易にするために与えられており、特許請求の範囲を限定するものとして解釈されるべきでない。

110 符号化システム
130 プロセッサシステム
140 ストレージ
150 通信インターフェース
160 検証システム
170 プロセッサシステム
180 ストレージ
190 通信インターフェース
200 符号化システム
210 ハッシュツリーユニット
211 ~ 213 ハッシュツリー符号化
A、A1、…、A34 階層データ
220 記憶ユニット
230 ストリーミングユニット
240 検証ユニット
300 ハッシュツリー
310 最も高いレベル
311 2 番目に最も高いレベル
313 3 番目に最も高いレベル
320 リーフレベル
330 選択された一部
1000 コンピュータ可読媒体
1010 書込み可能部分
1020 コンピュータプログラム
1110 集積回路
1120 処理ユニット
1122 メモリ
1124 専用集積回路
1126 通信要素
1130 相互接続
1140 プロセッサシステム

【発明を実施するための形態】

【0021】

開示されている本主題は多くの異なる形式の実施形態を受け容れる余地があるが、本開示は、開示されている本主題の原理の例示として考えられるべきであり、図示し、説明する特定の実施形態にそれを限定するものではないという理解とともに、1つ又は複数の特定の実施形態が図面に示されており、それについて本明細書で詳細に説明する。

【0022】

以下では、理解のために、動作中の実施形態の要素について説明する。しかしながら、

10

20

30

40

50

それぞれの要素は、それらによって実行される説明される機能を実行するように構成されることが明らかになる。

【0023】

さらに、開示されている本主題は、実施形態のみに限定されず、本明細書で説明するか又は相互に異なる従属請求項において具陳される特徴のあらゆる他の組合せをも含む。

【0024】

図1aは、符号化システム110の一実施形態の一例を概略的に示している。図1bは、検証システム160の一実施形態の一例を概略的に示している。符号化システム110は、プロセッサシステム130と、ストレージ140と、通信インターフェース150とを備える。検証システム160は、プロセッサシステム170と、ストレージ180と、通信インターフェース190とを備える。

10

【0025】

例えば、通信インターフェース150は、ゲノムデータ、例えばゲノムシーケンス及び／又は他のゲノムデータを受信するために構成された入力インターフェースを備える。プロセッサシステム130は、例えば、データ構造を生成するために、ストレージ140に記憶されたソフトウェアによって構成される。データ構造は、複数のゲノムブロックと、第1のハッシュツリーの一部とを含む。ゲノムデータの代わりに、他のタイプのデータ、特に他の階層データが使用され得ることに留意されたい。

【0026】

例えば、通信インターフェース190は、データ構造を受信するために構成された入力インターフェースを備える。プロセッサシステム170は、取得されたデータ構造中の第1のハッシュツリーの最も高いレベルを用いて、第1のハッシュツリーの再計算されたルートを検証するように構成される。

20

【0027】

ストレージ140及び／又は180は電子メモリ中に含まれる。ストレージは不揮発性ストレージを含み得る。ストレージ140及び／又は180は、非ローカルストレージ、例えばクラウドストレージを含み得る。後者の場合、ストレージは非ローカルストレージへのストレージインターフェースとして実装され得る。

【0028】

システム110及び／又は160は、コンピュータネットワークを介して、互いと、外部ストレージ、入力デバイス、出力デバイス、及び／又は1つ又は複数のセンサーと通信する。コンピュータネットワークは、インターネット、イントラネット、LAN、WLANなどであり得る。コンピュータネットワークはインターネットであり得る。本システムは、本システム内で、又は必要に応じて本システム外で通信するように構成された接続インターフェースを備える。例えば、接続インターフェースは、コネクタ、例えばワイヤードコネクタ、例えば、イーサネットコネクタ、光コネクタなど、又はワイヤレスコネクタ、例えばアンテナ、例えば、Wi-Fi、4G若しくは5Gアンテナを備える。センサーは、サンプルからゲノムシーケンシングデータを取得するためのシーケンシングデバイスであり得る。

30

【0029】

本システムはデジタル通信のために構成され、デジタル通信は、例えば、ゲノムデータを受信すること、データ構造を記憶すること、データ構造をストリーミングすること、例えば検証のためにデータ構造を取得すること、例えば、データ構造を受信すること又は取り出すことを含む。

40

【0030】

システム110及び160の実行は、本明細書中にそれらの例が示されている、プロセッサシステム、例えば1つ又は複数のプロセッサ回路、例えば、マイクロプロセッサにおいて実施される。図及び説明では、プロセッサシステムの機能ユニットであり得る機能ユニットについて説明している。例えば、図2aは、プロセッサシステムの可能な機能的組織の青写真として使用される。プロセッサ回路はこれらの図中のユニットとは別個に図示

50

されていない。例えば、図示されている機能ユニットは、システム 110 及び 160 において、例えばシステム 110 及び 160 の電子メモリに記憶されたコンピュータ命令に全体的に又は部分的に実装され、システム 110 及び 160 のマイクロプロセッサによって実行可能である。ハイブリッド実施形態では、機能ユニットは、一部はハードウェアにおいて、例えばコプロセッサ、例えば暗号コプロセッサとして実装され、一部は、システム 110 及び 160 上で記憶され、実行されるソフトウェアにおいて実装される。

【0031】

システム 110 及び 160 の様々な実施形態では、通信インターフェースは様々な選択肢から選択される。例えば、インターフェースは、ローカル又はワイドエリアネットワーク、例えばインターネットへのネットワークインターフェース、内部又は外部データストレージへのストレージインターフェース、キーボード、アプリケーションインターフェース (API) などである。

【0032】

システム 110 及び 160 はユーザインターフェースを有し、ユーザインターフェースは、1 つ又は複数のボタン、キーボード、ディスプレイ、タッチスクリーンなど、よく知られている要素を含む。ユーザインターフェースは、本システムを構成すること、ゲノムデータを取り出すこと、ゲノムデータを表示すること、ゲノムデータを検証することなどのためにユーザ対話を提供するために構成される。

【0033】

ストレージは、電子メモリ、例えばフラッシュメモリ、又は磁気メモリ、例えばハードディスクなどとして実装される。ストレージは、ストレージ 140、180 を一緒に構成する複数のディスクリットメモリを含み得る。ストレージは、一時メモリ、例えば RAM を含み得る。ストレージはクラウドストレージであり得る。

【0034】

システム 110 は単一のデバイス中に実装され得る。システム 160 は単一のデバイス中に実装され得る。一般に、システム 110 及び 160 は、それぞれ、システムに記憶された適切なソフトウェアを実行するマイクロプロセッサを備え、例えば、そのソフトウェアは、対応するメモリ、例えば RAM などの揮発性メモリ、又は Flash などの不揮発性メモリにダウンロード及び／又は記憶される。代替的に、本システムは、例えばフィールドプログラマブルゲートアレイ (FPGA) として、プログラマブル論理に全体的に又は部分的に実装される。本システムは、いわゆる特定用途向け集積回路 (ASIC)、例えば、それらの特定の用途のためにカスタマイズされた集積回路 (IC) として全体的に又は部分的に実装され得る。例えば、回路は、例えば、Verilog、VHDL など、ハードウェア記述言語を使用して CMOS に実装される。特に、システム 110 及び 160 は、ハッシュ関数又は署名など、暗号関数のための回路を備える。

【0035】

プロセッサ回路は、例えば複数のサブプロセッサ回路として、分散様式で実装され得る。ストレージは複数の分散型サブストレージにわたって分散され得る。メモリの一部又は全部は、電子メモリ、磁気メモリなどであり得る。例えば、ストレージは揮発性部分と不揮発性部分とを有する。ストレージの一部は読取り専用であり得る。

【0036】

符号化システム 110 は、単一のデバイスとして、又は複数のデバイスとして実装される。検証システム 160 は、単一のデバイスとして、又は複数のデバイスとして実装される。

【0037】

図 2a は、符号化システム 200 の一実施形態の一例を概略的に示している。図 2a に示されているのは、複数のレベルを有する階層データ構造である。最も高いレベルには、A と標示された単一のデータブロックが示されている。2 番目に最も高いレベルには、複数のブロックが示されており、この例では、A1、A2、及び A3 と標示された 3 つのデータブロックが示されている。2 番目に最も高いレベルにあるデータブロックのうちの 2

10

20

30

40

50

つは、その下にさらなる階層レベルをも有する。3番目に最も高いレベルには、2つのデータブロックシーケンスが示されており、データブロックA 1 1、A 1 2からA 1 4まではデータブロックA 1に従属し、データブロックA 3 1、A 3 2からA 3 4まではデータブロックA 3に従属する。階層データ構造中には4つ以上のレベルがあり得る。どのデータブロックの下にも、5つ以上の下位ブロック、例えば、8つ以上、20個以上などの下位ブロックがあり得る。

【0038】

シーケンス中の複数のブロックのために、ハッシュツリーが構築される。図2はハッシュツリーユニット210を示している。ハッシュツリーユニットは符号化システム200の関数ユニットであり得る。ハッシュツリーユニット210は、データブロックのシーケンスのためのハッシュツリーを構築するように構成される。ハッシュツリーはマークルツリーとしても知られる。ハッシュツリーはバイナリハッシュツリーであり得る。ハッシュツリーの例について、例えば、図3、図4、及び図5を参照しながら、本明細書でさらに示す。

10

【0039】

例えば、ゲノムデータなどのデータは複数のゲノムデータブロック214のシーケンスに分割される。シーケンス214は3つのブロックとともに示されているが、しばしば、4つ以上のブロック、例えば、一層多くのブロック、例えば、10個を超えるブロック、100個を超えるブロックなどを有する。ゲノムデータに加えて、複数のブロックは、追加のデータ、例えばゲノムデータに関連する情報、例えば、その起源、意味、目的などをもつ追加のデータブロックを含み得る。図2には、追加のブロックA 1 1、A 1 2及びゲノムデータブロックA 1 3からA 1 4までが示されている。図2には、追加のブロックA 3 1、A 3 2及びゲノムデータブロックA 3 3からA 3 4までが示されている。

20

【0040】

入力インターフェースは、例えばデータブロックの形態のゲノムデータを受信するために構成されるが、さらなるデータ項目を受信するようにも構成され得る。例えば、さらなるデータ項目及び／又はゲノムデータは、データ構造中で組み合わせられた複数のファイルから取り出される。さらなるデータ項目は第1のハッシュツリーのさらなるリーフに割り当てられ得、例えば、それらのハッシュ値はハッシュツリーのリーフ中に含まれる。

【0041】

マークルツリーとしても知られているハッシュツリーは、リーフノードがデータブロックの暗号ハッシュを用いて標示され、あらゆる非リーフノードがその子ノードのラベルの暗号ハッシュを用いて標示されるツリーである。この場合、211において構築されたハッシュツリーはそのリーフのためのブロックA 1 1からA 1 4までのハッシュを有する。大部分のノードは一般に2つの子ノードを有するが、1つ又は複数のノードは、例えば、2のべき乗でないブロックの数を考慮するために、1つの子ノードのみを有することがある。変形態として、ハッシュツリーは、4つ以上の子ノードを有するノードを有する。

30

【0042】

シーケンスA 1 1～A 1 4の上方のデータ構造の階層部分には、ハッシュツリーの一部が含まれる。例えば、ハッシュツリーユニット210は、少なくとも複数のゲノムデータブロック中のゲノムデータにハッシュ関数を適用することによって、複数のゲノムデータブロックA 1 3からA 1 4までのための複数のハッシュ値を計算する。ハッシュ関数は、好ましくは、暗号学的に強いハッシュ関数、例えばSHA-3である。随意には、追加のデータブロック、例えばブロックA 1 1及びA 1 2のための追加のハッシュが計算される。ハッシュツリーユニット210は、複数のハッシュ値のための第1のハッシュツリーを計算し、それにより、複数のハッシュ値を第1のハッシュツリーのリーフに割り当てる。一実施形態では、第1のハッシュツリーは少なくとも3つのレベルを有する。一実施形態では、第1のハッシュツリーは少なくとも4つのリーフを有する。

40

【0043】

50

ハッシュ関数は、特に、一方向圧縮関数を含むマークルードンガード構成の正則ハッシュ関数である。ハッシュツリー又はマークルツリー中のノードは、必ずしもそのようなハッシュ関数を適用することによって計算されるとは限らない。例えば、Verkleeツリー又は他の認証タグのためのベクトルコミットメントなど、他のタイプのフィンガープリントアルゴリズムが使用される。ゲノムデータブロックの複数のシーケンスを含む、データブロックの複数のシーケンスがあり得る。図2は、そのようなシーケンスの第2の例であるブロックA31～A34を示している。シーケンスA11～A13の場合と同様に、これらのブロックのためにも、ハッシュ値が計算され、ハッシュツリーが構築される。

【0044】

興味深いことに、データブロックのシーケンスのために計算されたハッシュツリーについて取得されたハッシュツリー情報は、階層的にその上方にあるブロック中に含まれる。例えば、ブロックA1はブロックA11～A14の上方の階層にある。ハッシュツリー情報は、より高いレベルの1つ又は複数のブロック中に含まれる。その例では、ブロックA1はハッシュツリー情報HTA1を含む。同様に、この例では、ブロックA3は、シーケンスA31～A34のためのハッシュツリーユニット210によって計算されたハッシュツリーから取得されたハッシュツリー情報HTA3を含む。

【0045】

一実施形態では、ハッシュツリー情報は、少なくともハッシュツリーのルートを含み、好ましくは、そのすぐ下のレベルをも含む。例えば、ハッシュツリー情報はハッシュツリーの2つの最も高いレベルを含む。興味深いことに、完全なハッシュツリーはハッシュツリー情報中に含まれない。ハッシュツリーのより大きい又はより小さい部分は除外される。例えば、一実施形態では、リーフから始まる最も低いレベルのうちの1つ又は複数がハッシュツリー情報から除外される。ハッシュツリーの一部を除外することにより、データ構造のためのストレージ又はストリーミング要件が低減される。除外された情報は、例えば検証のために、それらが必要とされる場合に、後で再計算することができるが、これらのレベルはリーフに近いので、それらは比較的少ない情報にわたって計算される。例えば、リーフ中のハッシュ値を再計算するためには、単一のデータブロックにわたるハッシュのみを計算すればよい。しかしながら、例えば、ハッシュツリーのルートに近いノードを再計算するためには、はるかに多いブロックを計算中に含める必要がある。

【0046】

一実施形態では、より低いレベルのハッシュツリーがデータ構造中に全部含まれるが、より低いハッシュツリーのルートがさらなるハッシュツリーのリーフ中で使用される。さらなるハッシュツリーは部分的に含まれることがあり、特に、より低いハッシュツリーのルートを含むリーフは省略されることがある。また、さらなるハッシュツリーの他の部分、特に、より低い又は最も低いレベルのうちの1つ又は複数は省略されることがある。

【0047】

最上位に近いレベル、例えば、より多くのレベルが可能であるが、少なくとも最初の2つのレベルを含めることによって、計算時間が最も低減され、最下位に近いレベルを除外することによって、ストレージ要件が最も低減される。

【0048】

ハッシュツリーを構築することは2回以上行うことができる。例えば、図2aは、ブロックA31～A34のシーケンスの別の例を示している。ハッシュツリーユニット210は、ブロックのためのハッシュ値を計算し、ハッシュツリーを計算し、ツリーの一部、例えば最初のいくつかのレベル、例えば最初の2つのレベルを含めるが、ツリーの少なくとも一部、例えば、階層的により高いブロックについてのハッシュツリー情報、この場合はHTA3中の最も低い又はより低いレベルを除外するように構成される。

【0049】

ハッシュツリー情報を含むブロック、この場合はブロックA1及びA3は、ハッシュツリー情報を記憶することに専用であり得る。ハッシュツリー情報は、他の情報、例えば、

10

20

30

40

50

階層的により低い及び／又は階層レベル中のデータに関する情報とともに含まれることがある。

【0050】

ハッシュツリーを構築することは複数のレベルにおいて行うことができる。例えば、図2aは、ブロックのシーケンスの別の例であるA1～A3を示している。ブロックA1～A3は、ブロックA11～A14及びA31～A34よりも階層的に高いレベルにある。これらのブロックのうちの1つ又は2つ以上、或いはこれらのブロックのすべては、より低い階層レベル、この場合はブロックA1及びA3におけるハッシュツリーのハッシュツリー情報を含んでいる。ハッシュツリーユニット213は、このシーケンス中のブロックのためのハッシュ値を計算し、ハッシュツリーを構築し、ハッシュツリーの一部、例えば最上位部分を、階層的により高いブロック、この場合はブロックA中に含める。ハッシュツリー情報はHTAと標示されている。このレベル2には、複数のシーケンス、例えば、（図2aに示されていない）シーケンスB1、B2などがあり得る。ブロックAのレベルには、複数のブロック、例えば、（図2aに示されていない）ブロックB、Cがあり得る。

10

【0051】

ハッシュツリー情報は、ハッシュツリー情報を計算するために使用された情報の完全性を検証するために、検証において使用され得る。概して、より多くのデータを検証することができるように、1つ又は複数のハッシュツリー中により多くのデータを含めることが有利であり、好ましくは、1つ又は複数のハッシュツリーを計算するために、すべてのゲノム情報が使用される。しかしながら、例えば、参照データ、命令データなど、他のデータほど敏感でないデータが含まれることが起こり得る。ゲノムのいくつかの部分は、他の部分、例えばいわゆるジャンクDNAよりもかなり敏感である。ゲノムデータは非常に大きくなり得るので、完全性保護からデータを除外することにより、データ構造の検証を著しくスピードアップすることができる。一実施形態では、複数のゲノムブロックのある部分は完全性保護付きと標示され、複数のゲノムブロックのある部分は完全性保護なしと標示されており、完全性保護付きと標示された部分のみが第1のハッシュツリー中に含まれる。

20

【0052】

ハッシュツリーは、そのハッシュツリーがそれにわたって計算されたデータの変更の検出を可能にする。興味深いことに、ハッシュツリーはデータの選択的な検証を可能にする。例えば、部分的な検証の場合、ハッシュツリーは、それが、再計算のために選択されたブロックに依存する限りにおいて、及びハッシュツリー中のハッシュ値がストレージから除外されたが、選択されなかったデータブロックのみに依存するハッシュツリー中の記憶されたハッシュ値を使用する限りにおいて、再計算される。

30

【0053】

ハッシュツリー単独では、しかしながら、悪意のある変更を防がない。例えば、悪意のある変更は、データを変更し、すべてのハッシュツリーを再計算し、置き換える。これを回避するために、符号化システムは、ハッシュツリーのルートにわたってデジタル署名を計算し、デジタル署名をデータ構造中に含めるように構成される。図2aはこれの一例を示している。ブロックAは、ブロックA1～A3のためのハッシュツリー情報を含み、ハッシュツリー情報にわたる署名をも含む。計算を低減するために、署名は、ハッシュツリー情報全体にわたってではなく、ハッシュツリーのルートのみにわたって計算される。

40

【0054】

一実施形態では、1つ又は複数の或いはすべてのゲノムブロックが圧縮形態で記憶される。例えば、ブロックA13～A14のうちの又はブロックA33～A34のうちの1つ又は複数、或いはブロックA13～A14の又はブロックA33～A34のすべてが圧縮される。完全性保護を助けるために、ハッシュツリーは、追加のハッシュ値、すなわち、非圧縮ブロックにわたって計算されたハッシュ値、並びに圧縮ブロックにわたって計算されたハッシュ値にわたって計算される。

50

【0055】

一実施形態では、1つ又は複数の或いはすべてのゲノムブロックが圧縮及び暗号化された形態で記憶される。一般に、ブロックは暗号化される前に圧縮される。例えば、ブロックA13～A14のうちの又はブロックA33～A34のうちの1つ又は複数、或いはブロックA13～A14の又はブロックA33～A34のすべてが圧縮され、次いで暗号化される。完全性保護を助けるために、ハッシュツリーは、追加のハッシュ値、すなわち、非圧縮及び非暗号化ブロックにわたって計算されたハッシュ値と、圧縮及び非暗号化ブロックにわたって計算されたハッシュ値と、圧縮及び暗号化ブロックにわたって計算されたハッシュ値とにわたって計算される。これらは随意の拡張であり、一実施形態は、例えば、圧縮及び暗号化ブロックにわたるハッシュ値のみを含むこともあり、例えば、非圧縮及び非暗号化ブロックにわたるハッシュ値のみを含むこともある。

10

【0056】

ハッシュツリーの別の利点は、選択されたブロックのみが変更された、例えば修正された場合に、修正されなかったブロックのために計算され、記憶されたハッシュツリー中のハッシュ値を使用することによって、ハッシュツリーを迅速に再計算することができることである。1つ又は複数の署名が使用される場合、それらは同様に再計算される。

【0057】

一実施形態では、デバイスはそのような修正を追跡する。例えば、ゲノムデータについての修正、例えば、追加、削除、及び／又は変更のうちの1つ又は複数を含む修正が受信される。1つ又は複数のハッシュツリーが全体的に又は部分的に再計算され、ツリーは、例えばストレージ中で更新される。興味深いことに、修正は追加のブロックとして記憶され得る。ゲノムデータを使用するときに、修正が適用され得る。修正は、新しいハッシュツリー中で使用されるか又は既存のハッシュツリー中に含められる追加のブロック中に含めることができる。このことは、ゲノムデータに対する修正を追跡することができるという利点を有する。

20

【0058】

ゲノムデータに対する修正は、例えば、メタデータに対する修正を含む。ゲノムデータに対する修正は、例えば、ゲノムシーケンスデータに対する修正を含む。ゲノムデータに対する修正は、例えば、新しいメタデータを含む、新しいデータを含む。データ構造に対する修正が適用され、それらに依拠する、例えば修正された部分から計算されたハッシュツリーの任意の部分が再計算される。追加又は代替として、例えばアカウントビリティを助けるために、修正自体がデータ構造中に記録される。修正は新しいデータブロック中に記憶され得る。リーフレベルにある新しいデータブロックはまた、ハッシュツリーの大部分が再計算されることを回避する。例えば、修正前にブロック1～100があった場合、修正はブロック101中に配置される。有利には、修正は、ハッシュツリーの大部分を再計算なしに再利用することができるように、末尾のブロック中に配置される。この場合、ブロック1及び2にわたるハッシュ、ブロック3及び4にわたるハッシュなどを再利用することができる。このことはより高いレベルにおいても働き、次のレベルのブロック1～4に依拠するハッシュもこの例などにおいて再利用することができる。データを読み出すときに、ブロック101中の修正は、データ中で必要とされる場合に適用される。

30

40

【0059】

システム200中に構築されたデータ構造は様々なやり方で使用される。

【0060】

例えば、システム200は記憶ユニット220を備える。記憶ユニット220は、データ構造をコンピュータ可読媒体、例えば非一時的コンピュータ可読媒体に書き込むように構成される。

【0061】

例えば、システム200はストリーミングユニット230を備える。ストリーミングユニット230は、コンピュータ可読媒体、例えば一時的コンピュータ可読媒体上のデータ構造又はその一部をストリーミングするように構成される。例えば、ストリーミングは

50

コンピュータネットワークを介して行われる。

【0062】

図2b及び図2cは、記憶された符号化の一実施形態の一例を概略的に示している。これらの例はまた、ストリーミングのために使用され得るが、ストリーミングにおいて、データ構造の一部を除外することが選択され得る。例えば、ストリーミングにおいて、符号化デバイスは、データ構造の選択された部分を示すパラメータを受信する。符号化デバイスは、次いで、選択された部分だけをストリーミングし、随意に、選択された部分の階層的に上方にあるブロックをもストリーミングすることがある。例えば、ストリーミングにおいて、符号化デバイスは、記憶されていないハッシュツリーのその部分を計算し、それらをストリーミング中に含める。例えば、送信デバイスが計算的により強力である場合、これにより受信デバイスにおける検証がスピードアップする。

10

【0063】

階層データ構造を記憶及び／又はストリーミングのために線形化することができる多くのやり方がある。例えば、階層関係は、ポインタ、又はラベルなどを用いてブロック中に示される。ブロックは様々な順序で書き込むことができる。図2bは、ブロックが深さ優先で書き出される一例を示している。図2cは、ブロックが幅優先で書き出される一例を示している。

【0064】

一実施形態では、符号化システムは、データ構造又はその一部をストリーミングするために構成される。完全性保護付きデータブロックをストリーミングする際に、例えばストリームの開始時に、トランスポートメッセージがストリーム中に含まれ得る。トランスポートメッセージは、

20

ハッシュツリーのうちの1つ又はハッシュツリーの階層の署名全体と、

受信端上でのトランスポートされたデータブロックの受信後直ちに完全性検証を実行することができるように、トランスポートされた1つ又は複数のノードから始まる、ルート全体までの1つ又は複数の経路中のノードを計算するために必要とされるすべてのハッシュノードとを含む。

【0065】

図2aに戻る。例えば、システム200は検証ユニット240を備える。検証ユニット240は、データ構造又はその一部を検証するように構成される。例えば、検証ユニット240は、図2b又は図2c中の記憶又はストリーミングされたものなど、データ構造を検証する。検証はすべての完全性データの検証を伴い得る。例えば、記憶されたハッシュツリーのすべての部分が再計算され、再計算された部分は、記憶された部分と比較される。完全性検証は署名の検証をも含み得る。興味深いことに、検証はデータ構造の選択された部分にも適用され得る。例えば、検証ユニットは、少なくとも選択されたゲノムデータブロック中のゲノムデータにハッシュ関数を適用することによって、データ構造中の複数のゲノムブロックの選択された部分のための複数のハッシュ値を再計算し、複数のゲノムブロックの選択された部分のための第1のハッシュツリーの再計算された複数のハッシュ値と、複数のゲノムブロックの選択されていない部分のための第1のハッシュツリーの

30

40

【0066】

ハッシュツリーは、それが、検証のために選択されたブロックに依存する限りにおいて、及びハッシュツリー中のハッシュ値がストレージから除外された限りにおいて、再計算される。選択されなかったが、そのためのハッシュ値が利用可能であるブロックの部分の場合、記憶／ストリーミングされたハッシュツリーからのハッシュ値が使用され得る。

【0067】

検証は、サーバに記憶されたデータ構造全体と、検証がそこで実行されるクライアントにおいて利用可能な検証されるべきデータブロックとを用いて、クライアントサーバ検証システムにおいて行われる。例えば、一実施形態では、以下の検証プロセスが使用される

50

。順序は異なることがあることに留意されたい。

1. クライアントは、検証されるべきデータ構造中のデータブロックのIDをサーバに通知する。

2. サーバは、検証されるデータブロックに関連付けられたノードから始まる、ハッシュツリーのうちの1つ又はハッシュツリーの階層のルート全体までの経路を特定する。

3. サーバは、第2項において特定された経路中のノードを計算するために必要とされるデータブロック又はコンテナを特定する。

4. サーバは、利用可能な場合、第3項において特定されたデータブロック又はコンテナに関連付けられたハッシュノードを取り出すか又は計算する。

5. サーバは、第4項において計算された取り出された／計算されたハッシュノードをクライアントに送信する。

6. クライアントは、選択されたデータブロックにハッシュ関数を適用することによって、データ完全性検証のために選択されたデータ構造中のデータブロックのための複数のハッシュ値を計算する。

7. クライアントは、ルート全体につながる経路に沿ってノードを計算するために必要とされる、検証されるデータブロックの計算された複数のハッシュ値（第6項）と、サーバから受信された計算された／取り出されたハッシュノード（第5項）とから、ハッシュツリーの階層のルート全体を再計算する。

8. クライアントは、取得されたデータ構造中のハッシュツリーのうちの1つ又はハッシュツリーの階層の最上位における再計算されたルート全体を検証する。

【0068】

図3は、ハッシュツリー300の一実施形態の一例を概略的に示している。図3中のものなど、ハッシュツリーはハッシュツリー計算211、212、213などにおいて使用される。ハッシュツリーは複数のレベルを有する。最も高いレベル310は、単一のノードである、ハッシュツリーのルートを有する。2番目に最も高いレベル311は少なくとも2つのノードを含んでいる。一実施形態では、レベル311の2つのノードがストレージ中に含まれる。また、図3には、3番目に最も高いレベル312と、リーフレベル320、例えば最も低いレベルとが示されている。ツリーの選択された部分はデータ構造中に記憶される。図3は、ストレージのための選択された部分330を示している。選択された部分330は最初の2つのレベルを含み得る。選択された部分330は、最も高い3つ以上のレベルを含み得る。選択された部分330はレベルの一部を含み得る。例えば、3番目に最も高いレベルの一部である。

【0069】

一実施形態では、ツリーパラメータ、例えば、ツリーサイズパラメータが符号化中に使用され、例えば、入力インターフェースにおいて受信される。ツリーサイズパラメータは、ハッシュツリーのどのくらいがデータ構造中に記憶されるべきであることを示す。ツリーサイズパラメータは、ハッシュツリーのより大きい部分が記憶されるべきであることを示すことがある。ツリーサイズパラメータは、ハッシュツリーのより小さい部分が記憶されるべきであることを示すことがある。このようにして、ユーザは、選択された部分についてのファイルサイズ又は検証速度が最適化されるべきであるかどうかを示すことができる。

【0070】

対応する検証システム、例えば検証システム160は、符号化システム110及び／又は符号化システム200に類似して働き得る。検証システムは、エンドユーザのレベルで働くだけでなく、中間のレベル、例えば、データ構造のソースとデータ構造のエンドユーザとの間のサーバのレベルにおいても働き得る。検証は、サーバにおいて、並びにエンドユーザのデバイスにおいて行われ得る。

【0071】

例えば、検証システムは、デジタルデータ構造の少なくとも一部を受信するために構成された入力インターフェースを備え、データ構造は、複数のゲノムブロックと、第1のハ

10

20

30

40

50

ッシュツリーの最初の2つの最も高いレベルを含むが、第1のハッシュツリーの1つ又は複数のより低いレベルを含まない、第1のハッシュツリーの部分とを含む。検証システムは、符号化デバイスによって生成されたデータ構造全体を受信する必要はないことに留意されたい。例えば、データ構造中のゲノムデータブロックの一部のみ、例えば、現在重要であるデータブロックのみが受信される。検証デバイスはまた、ハッシュツリーのすべてを受信する必要はない。例えば、送信も受信もされていないデータブロックのみに依拠するハッシュツリーの部分は、受信されていないデータのみに依存するハッシュツリーの最も高いレベルのハッシュ値のみを送ることによって要約することができる。例えば、ブロック1～4が受信されていない場合、ブロック1及び2又はブロック3及び4に依存するハッシュではなく、ブロック1～4のハッシュが必要とされる。ブロック5が受信されたと仮定した場合、ブロック1～4に依存するハッシュは、ハッシュツリールートにわたって署名を検証するためのブロック1～4についての十分な情報である。言い換えれば、特定のハッシュツリーノードが、受信されていないブロックのみに依存する場合、送信又は受信される必要がある前記特定のハッシュツリーノードの下方のハッシュツリーノードはない、すなわち、特定のハッシュツリーノードがそれに依存するハッシュツリーノードはない。特定のハッシュツリーノードがそれに依存するより高いハッシュツリーノードが送信され、また、より高いノードが、送信されていないデータブロックのみに依存する場合、特定のハッシュツリーノードでも必要とされないことがある。これは随意であるが、完全なハッシュツリーがデータ構造中で利用可能である限りにおいて、その完全なハッシュツリーは送られることがある。例えば、受信されたハッシュツリーは、いくつかのより高いレベルのすべてを含み、いくつかのより低いレベルのすべてを省略することがある。

10

20

【0072】

取得されたゲノムデータブロック、例えば受信されたゲノムデータブロックと、受信されたハッシュツリーの一部とを使用して、ハッシュツリーのルートが再計算される。受信されたブロックに依存する、ハッシュツリーの部分のために、ハッシュ値を再計算することができ、受信されていないブロックに依存するハッシュ値のために、受信されたハッシュ値を使用することができる。再計算されたハッシュツリーノード、特にハッシュツリールートを受信されたハッシュツリーノードと比較することができる。さらに、ハッシュツリールート上に署名がある場合、それを同様に検証することができる。ハッシュ値及び／又は署名に不一致が発見された場合、例えば、エラーがユーザに報告される、ファイルが拒否されるなど、適切なエラー処理が行われる。

30

【0073】

すべての受信されたゲノムデータブロックを検証する代わりに、受信されたゲノムデータブロックの部分を検証するために、同じ手法が使用される。

【0074】

以下に、いくつかのさらなる随意の改良、詳細、及び実施形態を示す。以下の実施形態は、ゲノムファイル又はビッグデータファイル全般の長期的な完全性保護を可能にするために適用することができる。実施形態についてISO/IEC 23092のコンテキストにおいて説明する。ISO/IEC 23092は、ゲノムデータを符号化、圧縮、及び保護するための規格を定義している。実施形態はそのコンテキストにおいて有利であるが、実施形態はそのコンテキスト以外でも適用することができる。

40

【0075】

現在のMPEG-GセキュリティソリューションISO/IEC FDIS 23092-3:2019(E)には、いくつかのセキュリティ問題がある。

MPEG-Gファイル全体をカバーする署名がない。

データセットグループの全体的完全性を検査するための署名がない。7.4.2において、rfmd、dgm d、及びdtprボックスが保護される。しかしながら、他のボックス、例えば、dtcn、dghd、rfgn、lab l及びlb lは保護されない。

署名は、ゲノムデータを含んでいるアクセスユニットの集合、又は関数データを含んでいる他のより低いレベルのデータ構造を保護し、それにより（ISO/IEC 23092

50

ー 1 : 2 0 1 9、6 . 3 において定義されている) 値フィールドのバイトに分解することができる。しかしながら、それがアクセスユニットの集合である場合、これは、アクセスユニットの各々が最大 2 5 6 個のブロックを含む、最大 2^{32} 個のアクセスユニットを含み得る。これにより、同じ集合中で保護される他のアクセスユニットに関して計算する必要なしに、ランダムアクセスのための選択されたアクセスユニットの妥当性を検査することが困難になる。

デジタル署名によって選択的に個々に保護されるデータ構造を用いると、現在のソリューションは、一般に、侵入者によるデータ構造の不正な追加、削除又は並べ替えを防ぐことができない。

データ構造が所与のファイルに属することを検証するためのプロセスは指定されていない。2つのデータ構造が同じファイルの一部として一緒に属していることをどのように検証するかは指定されていない。

【 0 0 7 6 】

実施形態はこれらの問題のうちの 1 つ又は複数に対処する。以下のうちの 1 つ又は複数が含まれる。

問題 1 : 現在の M P E G - G ソリューションはデジタル署名を個々のデータ構造 (又はコンテナ) に適用する。しかしながら、ファイル全体の完全性の証拠を与えるすべての要素を効率的にグループ化するやり方はない。好ましくは、これは、ランダムアクセスを保ちながら、低いオーバーヘッドを用いて達成される。

問題 2 : ゲノムデータが圧縮及び圧縮解除され、また、暗号化及び復号される。圧縮アルゴリズム、及び新しいソフトウェア実装はソフトウェアバグ及び／又は設計ミスが起こりやすいことがある。したがって、暗号化圧縮データ (E C D)、プレーンテキスト圧縮データ (P C D)、及びプレーンテキスト圧縮解除データ (P D D) の完全性及び正当性を検証することを可能にする必要がある。

問題 3 : ゲノムファイルが更新される。データファイル完全性検証のための構造は、好ましくは、この必要を可能にする。

問題 4 : ゲノムファイルが更新される。データファイル完全性検証のための構造は、好ましくは、変更と、それらの変更についてのアカウントビリティとを追跡することを可能にする。

問題 5 : データセットグループ I D が非常に短い (8 ビット) 。これにより、マーキング又は取出しなど、いくつかの使用事例を可能にする一意識別子を有することが実現不可能になる。

【 0 0 7 7 】

実施形態は、各マークルツリーがファイル中のデータ構造に結合された、マークルツリーの階層を用いて、M P E G - G ファイルなど、ゲノムファイルを向上させ、それにより以下を可能にすることを提案している。

ランダムアクセスを妨げることがない、ストレージ及び計算オーバーヘッドが低い、完全な全体としてのファイルのすべての又は選択されたデータ構造の長期的な完全性保護。これにより、データセットグループにおけるものを含む、任意のレベルにおけるデータ構造の効率的な検証と、それらのすべてをつなぎ合わせることが可能になる。

暗号化及び圧縮されている間、圧縮されている間、並びに圧縮解除されている間のゲノムデータの完全性保護。

トレーサビリティ及びアカウントビリティ。

現在のソリューションと比較したパフォーマンスの改善。

【 0 0 7 8 】

上記の最初の項目において長期的であることが強調されている理由は、ゲノム情報は、ユーザ及び彼／彼女の親族の健康にとって重要であるためである。このことは、ゲノム情報は、好ましくは、ユーザの一生の間だけでなく、彼／彼女の子供及び孫の一生の間も、内密で非公開のままであることを意味する。現在の M P E G - G ソリューションは、耐量子 (q u a n t u m r e s i s t a n t) でなく、予見できる時間内に解読され得る、

10

20

30

40

50

旧来のデジタル署名に依拠している。この問題を処理するための1つの選択肢は、既存のデジタル署名を耐量子のデジタル署名に置き換えることであるが、耐量子署名はE C D S Aよりも大きく、遅い。したがって、このことは、完全性保護の下で各データ構造のために個々の署名を必要とする、あまり効率的でないソリューションにつながる。提案されるソリューションは、（ハッシュ関数に基づく）マークルツリーに依拠しており、したがって、ツリーのルートのみが署名されればよいという追加の利益をもつ、長期的に完全性を保証するための自然なソリューションである。

【0079】

この提案される手法がランダムアクセスを妨げない理由は、それが、ファイル全体のデータにアクセスする必要なしに、コンテナ中のデータが変更されていないことと、コンテナがファイル全体の一部であることとを検証するためにコンテナにアクセスし、必要とされるMTノードを取り出すことが可能であるためである。

【0080】

データ構造のその特定の階層を参照しながら、MPEG-G規格のコンテキストにおいて実施形態について説明するが、特徴及び機能の大部分は、一般に、データを個々の構成要素に編成する任意のデータフォーマットに適用可能である。実施形態は、より小さいデータユニットの階層中に分配され、記憶される大量のデータを処理するために特に有益である。

【0081】

以下でいくつかの実施形態について説明するが、例えば、実施形態2は実施形態1に立脚し、実施形態3は実施形態2に立脚するなど、各実施形態は前の実施形態に立脚する。実施形態1、2、3、4及び5は、それぞれ上述の問題1、2、3、4及び5に対処する。実施形態は合計14個ある。

【0082】

さらに、実施形態を提示した後に、実施形態が代替ソリューションとどのように比較されるか、及び、実施形態が、ファイル完全性を保護するために、どのようにGA4GHセキュリティソリューションに適用され得るかについて説明する。

【0083】

実施形態1

この実施形態は問題1に対処する。それは、MPEG-Gファイル中の階層データ構造（又はコンテナ）をマークルツリー（MT）に関連付ける。一般に、より高いレベルにあるコンテナが、より低いレベルにあるデータ構造をカプセル化する。MPEG-Gファイル中のデータ構造は階層的な様式で編成されるので、より低いレベルにあるMTのルートは、より高いレベルにあるMTのリーフとして働く。

【0084】

この実施形態では、マークルツリーの5つのレベル{1、2、3、4、5}について説明する。最も低いレベルは1であり、最も高いレベルは5である。あらゆるレベルにおいて、マークルツリーのルートは、リーフデータが記憶された順序など、事前定義された順序で連結されたツリーの選択されたリーフハッシュコードのハッシングによって取得される。Hash（）は、ブラケット中で指定されたデータ構造のハッシュコードを生成する関数を示す。

【0085】

レベル1（MT__AU；MT__DS）は、MPEG-Gファイル中のアクセスユニット（Access Unit）（AU）レベル及びディスクリプタストリーム（Descriptor Stream）（DS）レベルにあるMTに関する。

各アクセスユニットは、以下を含み得るリーフをもつマークルツリーを形成する。

Hash（AU Header）、

Hash（AU Information）、

Hash（AU Protection）、及び

アクセスユニット中の各データブロックのためのHash（block）。

10

20

30

40

50

一般に、リーフは、ISO/IEC DIS 23092-1の表 25において定義されているAUコンテナ中のすべての要素又は要素のサブセットを含む。各アクセスユニットMTのルートはMTR__AUとして示される。

各ディスクリプタストリームは、以下を含み得るリーフをもつマークルツリーを形成する。

Hash (DS Header)、

Hash (DS Protection)、及び

ディスクリプタストリーム中の各ブロックのためのHash (block)。

一般に、リーフは、ISO/IEC DIS 23092-1の表 32において定義されているDSコンテナ中のすべての要素又は要素のサブセットを含む。各ディスクリプタストリームMTのルートはMTR__DSとして示される。

10

【0086】

レベル2-aは、MPEG-Gファイル中の属性グループ (Attribute Group) (AG) レベルにあるMTに関する。

各属性グループは、以下を含み得るリーフをもつマークルツリーを形成する。

Hash (AG Header)

1つ又は複数のMTR__AU

一般に、リーフは、ISO/IEC DIS 23092-6において定義されている属性グループコンテナ中のすべての要素又は要素のサブセットであり得る。

各属性グループMTのルートはMTR__AGと示される。

20

【0087】

レベル2-bは、MPEG-Gファイル中の注釈テーブル (Annotation Table) (AT) レベルにあるMTに関する。

各注釈テーブルは、以下を含み得るリーフをもつマークルツリーを形成する。

Hash (AT Header)

Hash (AT Metadata)

Hash (AT Protection)

1つ又は複数のMTR__AG

一般に、リーフは、ISO/IEC DIS 23092-6において定義されている注釈テーブルコンテナ中のすべての要素又は要素のサブセットであり得る。

30

各注釈テーブルMTのルートはMTR__ATと示される。

【0088】

レベル3 (MT__DT) は、MPEG-Gファイル中のデータセット (Dataset) (DT) レベルにあるMTに関する。

各データセットは、以下を含み得るリーフをもつマークルツリーを形成する。

Hash (Dataset Header)

Hash (Dataset Metadata)

Hash (Dataset Protection)

Hash (Dataset Parameter Set)

Hash (Master Index Table)

40

1つ又は複数のMTR__AU/MTR__DS/MTR__AT

一般に、リーフは、ISO/IEC DIS 23092-1の表 19において定義されているデータセットコンテナ中のすべての要素又は要素のサブセットであり得る。

各データセットMTのルートはMTR__DTとして示される。

【0089】

レベル4 (MT__DSG) は、MPEG-Gファイル中のデータセットグループ (Dataset Group) (DG) レベルにあるMTに関する。

各データセットグループは、以下を含み得るリーフをもつMTを形成する。

Hash (DG Header)

Hash (Reference)

50

Hash (Reference Metadata)

Hash (Label List)

Hash (DG Metadata)

Hash (DG Protection)

1つ又は複数のMTR__DT

一般に、リーフは、ISO/IEC DIS 23092-1の表9において定義されているデータセットグループコンテナ中のすべての要素又は要素のサブセットであり得る。

各データセットグループMTのルートはMTR__DGとして示される。

【0090】

レベル5 (MT__MPEGG) は、複数のデータセットグループをまとめるMPEG-Gファイルの最上位にあるMTに関する。

ファイルレベルにあるMTは、以下を含むリーフを含む。

Hash (File Header)

1つ又は複数のMTR__DG

ファイルレベルにあるMTのルートはMTR__MPEGGとして示される。

ファイルレベルにおいて、ファイル完全性の検証のために使用されるMTデータを記憶するための新しいマークルツリー完全性データ (Merkle Tree Integrity Data) コンテナ (mtid) が導入される。mtidボックスは、ファイルヘッダ (File Header) (flhd) の直後に、又は、更新が容易なようにファイルの末尾に配置され得る。

その最小形態では、mtidボックスは、ルート全体MTR__MPEGG上の署名と、ファイル所有者又は管理者であり得る署名者のIDとを含んでいる。

署名は関数Sign (PrivK, MTR__MPEGG) を用いて生成され、ここで、PrivKは、識別情報の証拠として認証局 (CA) によって署名された関連する公開鍵証明書をもつ署名者の秘密鍵である。

【0091】

この実施形態では、MTの5つのレベルが使用されるが、レベルの数は5つに限定されない。レベルの数は、MPEG-G規格中に導入されるコンテナの追加のレベルに適應するために増加させられるか、又はコンテナのレベルがより少ないファイルフォーマットのために減少させられ得る。

【0092】

リーフからルートを計算するためにバイナリマークルツリーを使用すると仮定して、リーフを検証するためのやり方が、米国特許第4309569号、Method of providing digital signaturesに記載されている。

【0093】

このプロセスは、全体的完全性について保護されている、それぞれ、4つのデータ要素D0、D1、D2及びD3上でのハッシングによって生成される4つのリーフL0、L1、L2及びL3をもつ例示的なマークルツリーとともに、図4に示されている。ツリーは、最も高いレベル410と、2番目に最も高いレベル420と、リーフレベル430と、データ440とを有する。

【0094】

例えば、D0は、IDをもつAUであり、D1はメタデータであり、D2～D3はゲノムデータブロックである。トランスポートモードにおいて、点描されているブロックのみを送るためには、ルートノードの再計算を可能にするために、他の点描されているノードも送られることがあることに留意されたい。ルートノード、又はルートノード上の署名N0-3も、再計算されたルート値の検証を可能にするために送られることがある。

【0095】

ツリーは、2つの内部ノードN0-1及びN2-3と、ルートN0-3とを含む。例えばデータ要素D1を検証するために、リーフレベルL0と、中間ノードN2-3とが開示されている。データ要素D1が与えられると、L1をハッシュ (D1) として計算することが可

10

20

30

40

50

能になる。L 0 及び L 1 を用いると、N 0 - 1 をハッシュ (L 0 | L 1) として計算することが可能になり、N 0 - 1 及び N 2 - 3 を用いると、パブリックルート N 0 - 3 をハッシュ (N 0 - 1 | N 2 - 3) として計算することが可能になる。

【0096】

M P E G - G 中のデータ構造に対応する個々のマークルツリーのレベルと、バイナリマークルツリー内の内部ノードのレベルとの間に差があることに留意されたい。外部マークルツリーレベルは、アクセスユニット又はディスクリプタストリームに関連付けられた最も低いレベルから、データセットグループなど、最上位レベルデータコンテナに関連付けられた最も高いレベルまでの範囲にわたる。2ⁿ個のデータ要素をもつバイナリマークルツリーの場合、最も低いレベルには、2ⁿ個のデータ要素上での個々のハッシングによって生成された2ⁿ個のリーフがあり、その場合、中間ノードのn個のレベルがある。l = 1、…、nである、各レベルlにおいて、ノードの数は2^(n-l)によって与えられる。バイナリマークルツリーのルートノードはレベルnにある。

10

【0097】

バイナリマークルツリーを構築するときに、特定のレベルに奇数個のリーフ又はノードがある場合、最後のリーフ又はノードをそれ自体と連結することができる。リーフA、B、C、D、及びEをもつ5ノードの例について考える。プロセスは以下のとおりである。

リーフレベル：5つのリーフがあり、A及びBはノードABにつながり、C及びDはノードCDにつながる。Eは単独であり、したがってノードEEにつながる。

レベル1：3つのノードがあり、AB及びCDはノードABCDにつながり、EEは単体であり、したがってEEEEになる。

20

レベル2：最後に、ルートノードABCDEEEEEを与える2つのノードABCD及びEEEEがある。

【0098】

実施形態2：

この実施形態は問題2に対処する。それは、実施形態1においてレベル0を導入することを含み、レベル0はレベル1よりも低い。これは以下のように行われる。

各アクセスユニット又はディスクリプタストリームのためのマークルツリー中のリーフとして使用されるHash (Block) は、そのブロックのための(最大)3つの値、

Hash (ECD)、圧縮及び暗号化ブロックのハッシュ

30

Hash (PCD)、圧縮ブロックのハッシュ

Hash (PDD)、圧縮解除した後のブロックのハッシュ

から計算されたブロックMTR__blockのMTRルートに置き換えられる。

【0099】

3つの値すべてが常に必要とされずとは限らない。含められ、検証される値に応じて、完全性欠如の理由をより高い又は低い精度で検査することが可能になる。

【0100】

上記のハッシュ値はそれぞれリーフに割り当てられる。これにより、エラーを、例えば圧縮のエラーなどとしてピンポイントで特定することが可能になる。完全性は異なる場所において検査され得ることに留意されたい。サーバは、それが復号鍵へのアクセスを有しない場合、復号されたデータを検査することができないが、サーバは、暗号化されたデータを検査することができる。

40

【0101】

さらに、一例では、マークルツリーのルートは、

MTR__Block = Hash (Hash (Hash (ECD) | Hash (PCD)) | Hash (PDD)) である。この場合、例えば3進ツリー構造に従うルートのための異なる計算、例えばHash (Hash (ECD) | Hash (PCD) | Hash (PDD)) も妥当であることに留意されたい。

これにより、復号及び圧縮解除プロセスの異なる段階において、また、規格の分散型実装における異なるロケーション(例えば、クライアントとクラウド)においてデータの完

50

全性を検査することが可能になる。

【0102】

例えば、MPEG-Gクライアントが、クラウドに記憶されたMPEG-Gファイルからのデータを要求すると仮定する。クラウドは、暗号化及び圧縮されたデータが変更されていないことを検査するために、ユーザの認証情報 (credential) を検査し、マークルツリー中の情報、すなわち Hash (ECD) を使用し得る。クラウドは、次いで、(i) 帯域幅を節約するために圧縮され、(ii) クラウド自体はユーザデータへのアクセスを有しないので暗号化された、暗号化及び圧縮されたデータをMPEG-Gクライアントに送る。MPEG-Gクライアントがデータを受信したとき、MPEG-Gクライアントは、最初にデータの完全性を検査するために、Hash (ECD) を使用することができる。次いで、MPEG-Gクライアントは、データを復号し、復号及び圧縮されたデータの完全性を検査するために Hash (PCD) を使用することができる。最後に、MPEG-Gクライアントは、データを圧縮解除し、復号及び圧縮解除されたデータの完全性を検査するために Hash (PDD) を使用することができる。

10

【0103】

実施形態3：

この実施形態は問題3に対処する。この目的のために、レベル0、1、2、3又は4にあるデータ構造中のデータが追加及び／又は変更されたとき、

レベルiにあるその新しいデータ構造のための新しい／更新されたMTが計算される。

MTR__MPEGG、したがって、MTR__MPEGG上の署名がレベル5において更新されるまで、以下のプロセスが実行される。

20

新しい（又は更新された）リーフが次の上位レベルMTに追加されることになる。新しい（又は更新された）リーフの値は、現在のレベルiにあるMTのMTRの値である。

次の上位レベルにあるMTは新しい／更新されたリーフを有するので、そのルートも更新される。

【0104】

実施形態4：

この実施形態は問題4に対処する。この目的のために、1つの選択肢は、MPEG-Gファイル中の変更／更新（**）を追跡するために、実施形態1に記載されているmtidボックス中に追跡テーブルを含めることである。ファイルの作成以後、 $i = 0, 1, 2, \dots$ である、 i 番目に生成された署名に対応する、テーブル中の i 番目のエントリは以下を含んでいる。

30

効率的な検証及び記憶を可能にするための、すべての又は選択されたルート値MTR__DG、MTR__DT、MTR__AT、MTR__AG、MTR__AU、MTR__DS及びマークルツリーノード。さらなる詳細については、実施形態6～8を参照されたい。

$\text{Sign}(\text{PrivK}, i \mid \text{MTR_MPEGG} \mid \text{SIGNATURE_MTR_MPEGG}(i-1))$ として計算された $\text{SIGNATURE_MTR_MPEGG}(i)$ と示された現在のMTR__MPEGG上の署名、ここで、

$\text{SIGNATURE_MTR_MPEGG}(-1) = \text{""}$ であり、

PrivKは、識別情報の証拠として認証局(CA)によって署名された関連付けられた公開鍵証明書をもつ署名者の秘密鍵である。

40

長期的な完全性保護を目的とするとき、LMSなど、ハッシュベースの署名アルゴリズム、すなわち、例えば、RFC8554、又は耐量子アルゴリズムのうちの1つ、例えば、Falcon、SPHINCS（現在NISTによる規格化中）が好ましい。特に、いくつかのエントリにおいてプレ量子(pre-quantum)署名アルゴリズム（例えばECDSA）が使用される場合、この場合、ECDSAが、例えば量子コンピュータにより解読されるとすぐに、それらのエントリを検証することが不可能であることがある。

署名者のID

$i = 1, 2, \dots$ である、時間iにおいてファイルが変更されたとき、

50

M T R _ M P E G G は、実施形態 3 に記載されているように更新される。

m t i d 中の追跡テーブル中に新しいエントリが含まれる。追跡することを容易にするために、追跡データは以下を含み得る。

S I G N A T U R E _ M T R _ M P E G G (i)

新しい追加データ（データ追加の場合）、又は前のデータと比較した変更されたデータの差。

変更又は追加されたリーフ。

さらに、

現在のマークルツリールートまでの経路を通過することによって、リーフのデータにアクセスするときに、データのそのピースについてのファイル完全性及びすべての関係する変更を検査することができる。

m t i d 中の追跡テーブル中の特定のデータ項目を検索するときに、データのそのピースの追加／変更についてのすべての情報が見つけられる。

【0105】

実施形態 5 :

この実施形態は問題 5 に対処する。このことは、以下によって行われる。

d a t a s e t G r o u p I D を十分な長さ（例えば 256 ビット長）にし、それをランダムに生成すること。

特に、d a t a s e t G r o u p I D を M T R _ D G と置き換えること。d a t a s e t G r o u p I D が一意であることを保証するために、M T R _ D G の計算中の追加の入力、すなわち、十分に長いランダムナンス（r a n d o m n o n c e）N _ M T R _ D G が含まれる。

同様に、d a t a s e t I D を M T R _ D S と置き換えることができる。d a t a s e t G r o u p I D が一意であることを保証するために、M T R _ D G の計算中の追加の入力、すなわち、十分に長いランダムナンス N _ M T R _ D G が含まれる。

【0106】

この変更の利点は、各データセット又はデータセットグループに一意識別子が割り当てられることである。このことは、現在の M P E G - G 規格が、これらの識別子に基づいてエントリを取り出すためにファイルと A P I とをマージすることについての使用事例を含むので、利点である。識別子範囲が 8 ビットしかない場合、2 つのデータセットグループをマージするには、識別子をリネームする必要があることがある。識別子がリネームされなかった場合、A P I に対するコールは、間違った結果を戻すことがある。M T R _ D G 及び M T R _ D S を d a t a s e t G r o u p I D 及び d a t a s e t I D として使用することにより、これらの 2 つの問題が解決される。

【0107】

実施形態 6 : M T 記憶オーバーヘッド及び計算性能トレードオフ及び最適化 :

一般に、小数で存在するデータ構造の場合、大きい記憶オーバーヘッドを負うことなしに完全性検証の効率を改善するためにマークルツリー（M T）全体を記憶することができる。一方、通常、多数で存在するデータ構造の場合、記憶オーバーヘッドと計算リソースとの間のトレードオフが好ましい。2ⁿ 個のリーフをもつバイナリ M T の場合、2ⁿ 個のリーフのすべてが利用可能である場合に、M T ノード及びルートの計算は（2ⁿ - 1）回のハッシュ演算を要する。バイナリ M T 中に、すべての中間ノード及びルートを含む、合計 2ⁿ - 1 個のノードがある。

【0108】

この実施形態は、バイナリ M T のルートから数えて、1 ≤ m ≤ n である、ノードの最上位 m 個のレベルを記憶するための最適化を提案する。ノードのこのセットがすべてのリーフと一緒に記憶された場合、リーフを検証するための計算上の負担は、ノードのペア上の [2^(n-m+1) - 1 + (m-1)] 回のハッシュ演算まで低減される。リーフの記憶により、特に、データ構造が大きい場合、データ構造上のハッシングの費用がかかる演算が回避される。

10

20

30

40

50

【0109】

この最適化は、点描されている三角形が部分的なマークルツリーを表す、図5aに示されている。図5aには、最も高いレベル510、例えばレベルnにあるルートノード511が示されている。また、520にはレベルn-(m-1)、530にはレベルn-m、540にはレベル1、550にはリーフレベル（レベル0）が示されている。522にはm個のレベル、521にはn個のレベルが示されている。レベル520は $2^{n-(m-1)}$ 個のノードを含む。レベル530は 2^{n-m} 個のノードを含む。点描されている三角形部分は 2^m-1 個のノードを含む。レベル1は 2^{n-1} 個のノードを有する。レベル0は 2^n 個のノードを有する。

【0110】

三角形560は、必要とされるときにオンザフライで生成されるノードを示している。三角形560は $2^{n-m+1}-1$ 個のノードを含む。ツリー500の点描されていない部分、例えばレベル530以下は記憶されていない。最上位レベル、例えばレベル520以上は記憶されている。

【0111】

マークルツリー全体は、この例では、 2^n 個のリーフを有する。ルートを含む、最上位m個のレベルにあるリーフとノードの両方が記憶されている。最下位（n-m+1）個のレベルにある三角形560は、特定のリーフを検証する必要があるときに、オンザフライで再計算される。

【0112】

多数のリーフ、例えば 2^{32} 個ものリーフがあり得る。必要とされる場合、さらに多くのリーフがあり得る。ブロックサイズも様々であり得る。ブロックサイズが小さいほど、小さいオーバーヘッドでよりきめ細かい完全性検査が可能になるが、ハッシュツリーのサイズは増加することがある。例えば、ブロックサイズを決定するためにツリーサイズパラメータが使用される。

【0113】

記憶されているMTの部分はファイルの一部であり得る。別個のファイルとして記憶され得る。これはデータ構造の一部として記憶することができる。

【0114】

例えば、n=32、m=24である場合、32バイトのハッシュコードサイズを仮定すると、それは、ツリーの最上位部分のための（ $2^{24}-1$ ）回のハッシュ（約 $2^{24} \times 32$ バイト）と、 2^{32} 個のリーフとを記憶している。リーフを検証することは、最下位にある部分的なMTの再計算のための（ 2^9-1 ）回のハッシュ演算に加えて、ルートに達するための23回のハッシュを要する。（ 2^9+22 ）回の演算は非常に速い。メモリアーバーヘッドはデータ構造のリーフの数に大きく依存する。

【0115】

代替として、 2^n 個のリーフが記憶されていない場合、ただノードの最上位m個のレベルを記憶することによって、依然として、完全性検証のために計算される必要があるリーフの数を $2^{(n-m+1)}$ 個に低減することができる。この場合、記憶オーバーヘッドは（ 2^m-1 ）回のハッシュによって与えられ、マークルツリールートを再計算するための時間は、

$$T_{\text{binary_MT}} = [2^{(n-m+1)} - 1 + (m-1)] * t_{\text{node}} + 2^{(n-m+1)} * t_{\text{leaf}}$$

によって与えられ、

ここで、 t_{node} は、2つのノード上でのハッシングのための時間であり、 t_{leaf} は、データ構造上でのハッシングによってリーフを生成するための時間である。

リーフ又はノードの記憶がない場合、計算時間は、

$$T_{\text{binary_MT}} = [2^{n-1}] * t_{\text{node}} + 2^n * t_{\text{leaf}}$$

になる。

【0116】

10

20

30

40

50

ファイルがサーバに常駐し、検証がクライアントにおいて行われる、クライアントサーバ設定におけるバイナリMTの演算について、 $n = 4$ 及び $m = 2$ である、図 5 b に示されている例を使用してさらに説明する。図 5 b は、レベル 4 ~ 1 とリーフレベルとをもつハッシュツリーを示している。レベル 4 にはノード R があり、レベル 3 にはノード N 3 1 及び N 3 2 があり、レベル 2 にはノード N 2 1、N 2 2、N 2 3 及び N 2 4 があり、レベル 1 にはノード N 1 1 ~ N 1 8 があり、リーフレベルには、データ D 1 ~ D 1 6 に対応するノード L 1 ~ L 1 6 がある。

【0117】

レベル 3 及び 4 にあるすべてのノードと、すべてのリーフとが事前計算され、例えばデータブロック中に記憶されていると仮定する。(ハイライトされている) D 1 2 を検証するために、以下のステップが取られる。

1. サーバが、(ハイライトされている) ノード {N 1 5、N 1 7、N 1 8、N 2 4} を生成する。一般に、サーバにおけるハッシュ演算の回数は $\{ [2^{(n-m+1)} - 1] - (n-m+1) \}$ によって与えられる。

2. サーバがハッシュ {L 1 1、N 1 5、N 2 4、N 3 1} をルートの署名と一緒にクライアントに送る。一般に、送る必要があるハッシュの数は n である。

3. クライアントが、

データ構成要素 D 1 2 上でのハッシングによってリーフ L 1 2 を生成し、

サーバから受信されたハッシュを使用して、(ハイライトされている) ノード {N 1 6、N 2 3、N 3 2、R} を生成する。一般に、クライアントにおけるハッシュのペア上でのハッシュ演算の回数は n 回である。

4. クライアントが、関連付けられた公開鍵を使用して、受信された署名を復号し、復号されたハッシュコードを生成されたルート R と比較する。それらが互いに一致すれば、検証は成功である。

【0118】

この手法のコストを、完全性について保護されているデータ構造ごとに 1 つのデジタル署名を必要とする現在の M P E G - G と比較すると、この手法では、

2 つのデータ構造を検証する必要があるとすぐに、計算オーバーヘッドが減少する。その理由は、デジタル署名検証のコストがハッシュ計算よりもはるかに費用がかかるためである。MT 手法はファイルの MT ルートのための単一の署名検証を伴う。

同じデータ構造が完全性保護付きであると仮定すると、記憶オーバーヘッドが減少する。その理由は、現在の M P E G - G がデータ構造ごとにデジタル署名を必要とし、また、特に耐量子署名が使用される場合、署名サイズがハッシュ値よりも大きくなるためである。

【0119】

下表は、ツリー名と、リーフの最大数と、ストレージのニーズと、リーフを検証するためのハッシュ演算の回数と、バイナリツリー中の内部レベルの数とを含む、マークルツリーの異なるレベルの特性を要約している。

【0120】

10

20

30

40

50

【表 1】

レベル	データ構造の説明	ツリー名	リーフの最大数	ストレージ (ハッシュコードの数)	検証のためのハッシングの回数	バイナリツリー中のレベルの最大数	
0	データブロックのサイズは、32 ビット符号なし整数、例えば $u(32)$ によって指定される。このことは、それが最大 4GB になり得ることを意味する。	MT_Block (実施形態 2)	3	3 (ルート及び第 1 レベル) + 3 (リーフ)	2	2	10
1	アクセスユニット(AU)中のブロックの数は、データタイプ $u(8)$ で最大 256 個である。	MT_AU	256	$2^8 - 1$ (すべての MT ノード) + 256 (リーフ)	8	8	
1	ディスクリプタストリーム (DS) 中のブロックの数は、データタイプ $u(32)$ で最大 4.295 M 個である。	MT_DS	2^{32}	$2^{24} - 1$ (ルートを含む最上位 24 個のレベル) + 最大 2^{32} (リーフ)	$2^8 + 24 - 1$	32	20
3	データセット中の AU の数は、データタイプ $u(32)$ で最大 4.295 M 個である。	MT_DT	2^{32}	$2^{24} - 1$ (ルートを含む最上位 24 個のレベル) + 最大 2^{32} (リーフ)	$2^8 + 24 - 1$	32	
3	データセット中の DS の数は、データタイプ $u(7)$ で最大 128 個である。	MT_DT	128	$2^7 - 1$ (すべての MT ノード) + 128 (リーフ)	7	7	30
4	データセットグループ中のデータセットの数は、データタイプ $u(16)$ で最大 65,536 個である。	MT_DSG	65,536	$2^{16} - 1$ (すべての MT ノード) + 2^{16} (リーフ)	16	16	

【0 1 2 1】

実施形態 7：暗号化データコンテナのためのハッシュコードに代わる認証タグの使用

MPEG-G は、GCM モードにおける AES を使用することによって、対称鍵を用いて異なるコンテナを保護することを可能にすることに留意されたい。このことは、対応する対称鍵を使用することによって、データコンテナを暗号化し、認証することもできることを意味する。認証は、データコンテナ全体に依存する記憶された認証タグを検査することによって実行される。既存の MPEG-G ソリューション中の認証タグは、CPU 及びメモリ要件を低減するために、データコンテナのフィンガープリントとして再利用される。特に、AES-CCM 認証タグは、例えば、実施形態 1 のレベル 1 にある Hash (block)、又は、実施形態 2 における Hash (ECD) に取って代わることができる。認証タグがフィンガープリントとして再利用される場合、データコンテナ (MT リー

10

20

30

40

50

フ) のハッシュは再計算又は記憶する必要がない。

【0122】

実施形態8：選択的な完全性保護

一実施形態では、すべてのデータ構造、又はコンテナが保護される。しかしながら、いくつかの状況では、ユーザは部分的な保護を選択し得る。この実施形態では、データ構造の選択的な保護をどのように実現することができるかについて説明する。

【0123】

この目的のために、各コンテナタイプは、そのコンテナタイプのデータ構造が一般にどのように選択されるかを示すためのフィールドを有する。例えば、フィールドの値は以下であり得る。

0－いずれも選択されていない（マークルツリーは、このコンテナレベル以下にあるすべてのデータ構造を除外する）

1－すべてが選択されている

2－IDによってデータ構造を選択する

3－例えば、上位レベルコンテナ中のそれらの位置によってデータ構造の選択を示すために、ビットのシーケンスを使用して、フラグによってデータ構造を選択する

【0124】

以下は、MT検証中にそれらを含めるためのデータセットグループコンテナ、データセットコンテナ、注釈テーブルコンテナ、アクセスユニットコンテナ及びブロックコンテナの一般的な選択モードのための例示的な設定である。

mt__select__dg=1（すべてのデータセットグループを選択する）

mt__select__dt=2（IDによってデータセットを選択する）

mt__select__at=2（IDによって注釈テーブルを選択する）

mt__select__au=2（IDによってアクセスユニットを選択する）

mt__select__bl=1（すべてのペイロードブロックを選択する）

【0125】

各コンテナタイプのためのこれらの一般的な選択設定は個々のコンテナごとにオーバーライドすることができる。コンテナが除外されると、すべてのその下位データ構造は、それらの選択設定にかかわらず除外されることに留意されたい。

【0126】

実施形態9：完全性検証の速度を向上させるためのハッシュコードの選択的な記憶

ハッシュコードの記憶は署名検証の速度を改善することができるが、記憶オーバーヘッドが犠牲になる。実施形態6に記載されているように、コンテナレベルがルートに近くなる（ハッシュコードの数が少なくなる）ほど、記憶コストは軽くなり、計算速度を改善することの利益が大きくなる（各ハッシュコードはデータのより大きい塊を表す）。この実施形態では、MTハッシュコード（又はノード）をどのように選択的に記憶することができるかについて説明する。

【0127】

この目的のために、各コンテナタイプは、バイナリツリー中のリーフと他のノードの両方（実施形態6参照）を含む、その対応するMTハッシュコードが一般に記憶のためにどのように選択されるべきであるかを示すためのフィールドを有する。例えば、フィールドの値は以下であり得る。

0－いずれのノードもない

1－すべてのリーフと中間ノードとを含む、バイナリMT中のすべてのノード

2－すべてのリーフのみ

3－バイナリMTの最上位m個のレベルにあるノード

4－最初のn個のリーフ

5－バイナリMTの最上位m個のレベルにあるすべてのリーフ及びノード

6－バイナリMTの最上位m個のレベルにある最初のn個のリーフ及びノード

【0128】

10

20

30

40

50

以下は、ファイル（MPEGG）レベル、データセットグループ（DG）レベル、データセット（DT）レベル、注釈テーブル（AT）レベル及びアクセスユニット（AU）レベルにおけるMTの一般的な記憶モードのための例示的な設定である。

$mt_store_mpeg = 1$ （ファイルレベル全体にあるすべてのMTノードを、MTR_DGであるリーフとともに記憶する）

$mt_store_dg = 1$ （データセットグループレベルにあるすべてのMTノードを、MTR_DTであるリーフとともに記憶する）

$mt_store_dt = 1$ （データセットレベルにあるすべてのMTノードを、MTR_ATであるリーフとともに記憶する）

$mt_store_at = 1$ （注釈テーブルレベルにあるすべてのMTノードを、MTR_AUであるリーフとともに記憶する）

$mt_store_au = 0$ （アクセスユニットレベルにあるMTノードは記憶されない）

【0129】

記憶モード3～6の場合、記憶されるべきバイナリMT中の最上位レベルの数（m）及びリーフの数（n）を指定するために、追加のフィールドが必要とされる。各コンテナタイプのためのこれらの一般的なMT記憶設定は個々のコンテナごとにオーバーライドすることができる。

【0130】

実施形態10：全面的な又は的を絞った完全性検証

特定の構成要素の完全性を検証するためにマークルツリー上のハッシュコードの一部又は全部が記憶されている場合は、問題の構成要素を対象とするハッシュコードのみを再生成するだけでよい。次いで、問題の構成要素からツリーのルートまで遡るすべてのハッシュコードが検証され、最後に署名が検査される。

【0131】

実施形態11：柔軟なマークルツリー編成－バイナリツリー対K進ツリー、単一のツリー階層対複数のツリー階層

上記実施形態中の説明は、MPEG-Gファイルデータ構造の場合と同様に編成されたマークルツリーの階層に焦点を当てている。各保護されているデータ構造は独立したマークルツリーに関連付けられる。実施形態は一般にバイナリマークルツリーを仮定しており、例えば、ツリー中の各ノードは高々2つの子ノードを有する。したがって、MPEG-Gファイルなど、データファイルは、各々がデータコンテナに対応するバイナリマークルツリーの階層を有する。バイナリツリー構成の代替として、K進マークルツリーは、それぞれがk個の下位ノード上での同時のハッシングによって計算されたノードを含む。極端な場合、kはリーフの総数に等しく、それによりツリー深さは1になり得る。言い換えれば、ルートは、N個のリーフすべての連結上での直接ハッシングによって計算される。ハッシュ関数がO(N)の線形時間計算量（linear time complexity）を有し、バイナリモデルとの比較の容易さのために、 $N = 2^n$ であると仮定すると、N進マークルツリーのルートを再計算するための時間は、

$$T_{N\text{-ary_MT}} = 2^{(n-1)} * t_{\text{node}} + 2^n * t_{\text{leaf}}$$

によって与えられ、ここで、 t_{node} は、2つのノード上でのハッシングのための時間であり、 t_{leaf} は、データ構造上でのハッシングによってリーフを生成するための時間である。

【0132】

実施形態6に記載されているように、ノードの最上位m個のレベルが記憶されている、バイナリマークルツリーのルートを再計算するための時間は、

$$T_{\text{binary_MT}} = [2^{(n-m+1)} - 1 + (m-1)] * t_{\text{node}} + 2^{(n-m+1)} * t_{\text{leaf}}$$

によって与えられる。

【0133】

10

20

30

40

50

$1 \leq m \leq n$ である、 $(2^m - 1)$ 個のハッシュコードの記憶オーバーヘッドを用いて、2つの手法を比較すると、(1) $t_{leaf} \gg t_{node}$ であり、(2) 例えば、リーフの最大数が256個であるMT_AUの場合に、リーフの数が比較的小さく、(3) バイナリマークルツリー手法のためのノードとリーフとの混合記憶がないと仮定すると、

$T_{N-ary_MT} = (2^n - 2^m + 1) * t_{leaf}$
 $T_{Binary_MT} = 2^{(n-m+1)} * t_{leaf}$
 $m = n$ であるとき、 $T_{N-ary_MT} = t_{leaf}$ であるが、 $T_{Binary_MT} = 2 * t_{leaf}$ である。

$m = n - l$ であり、 $1 \leq l \leq (n - 1)$ であるとき、

$T_{Binary_MT} = 2^{(n-m+1)} * t_{leaf} = 2^{(l+1)} * t_{leaf}$
 $T_{N-ary_MT} = [2^n - 2^{(n-l)} + 1] * t_{leaf} > 2^{(l+1)} * t_{leaf}$
 $* \{2^{[n-(l+1)]} - 2^{[n-2(l+1)]}\} > 2^{(l+1)} * t_{leaf}$
 T_{Binary_MT}
 である。

【0134】

完全性検証の速度に関する上記の計算に基づいて、リーフを生成することと比較して、ノード上でのハッシングの計算時間は無視できるという仮定の下で、以下のことを結論付けることができる。

記憶オーバーヘッドが $(2^n - 4)$ 回のハッシュ以下であるとき、バイナリMT手法はより速くなる。その理由は、 $m = (n - 1)$ 又は記憶オーバーヘッドが $[2^{(n-1)} - 1]$ であるとき、 $T_{Binary_MT} = 4 t_{leaf}$ であるためである。この計算時間は、記憶オーバーヘッドが、次のレベルをカバーするのに十分に大きくなるまで、例えば、 $m = n$ 又は記憶オーバーヘッドが $(2^n - 1)$ になるまで同じままである。一方、N進手法の場合、記憶オーバーヘッドが $(2^n - 4)$ 回のハッシュ以下であるとき、 $T_{N-ary_MT} \geq 4 t_{leaf}$ である。

N進ツリー手法は、記憶オーバーヘッドが $(2^n - 3)$ 回のハッシュと 2^n 回のハッシュとの間にあるときにより速くなる。

バイナリMT手法では、記憶オーバーヘッドが 2^n 回のハッシュを超えると、 2^n 個のリーフが最初に記憶され得、残りのオーバーヘッドは、内部MTノードを記憶するために使用され得、それにより検証速度のさらなる向上につながる。

【0135】

ノード上でのハッシングの計算時間のみを考えると、

$T_{Binary_MT_nodes_only} = [2^{(n-m+1)} - 1 + (m - 1)] * t_{node} < 2^{(n-1)} * t_{node} < T_{N-ary_MT_nodes_only}$
 であるので、バイナリ手法は、 $m > 1$ であるとき、常にN進手法よりも優れている。

【0136】

したがって、全体として、バイナリMT手法は、ノードの最上位 $m \geq (n - 1)$ 個のレベルのみを記憶する事例、又はすべてのリーフ+ルートを記憶する事例($m = 1$)に対応する、 $(2^n - 3)$ 回のハッシュから $(2^n + 2)$ 回のハッシュまでの記憶オーバーヘッドの狭い範囲内を除いて、N進手法よりも良い性能を有する。そのような特定の記憶オーバーヘッドの制約の下では、検証速度における明らかな勝者はいない。

【0137】

これらの分析結果は、選択されたリーフの数(実施形態8)と、記憶モード設定(実施形態9)とに基づいて、バイナリマークルツリー手法かN進マークルツリー手法かの選定のための指針を与えることができる。ファイルがサーバに常駐し、検証がクライアントにおいて行われる、クライアントサーバ設定においては、バイナリ手法では n 個のノードのみを受信機に送り、一方、N進手法では、 $(2^n - 1)$ 個のリーフノードを送る必要があることに留意されたい。

【0138】

また、より柔軟なマークルツリー編成を可能にすることに利点があり得る。1つの柔軟

10

20

30

40

50

性は、ファイルごとに複数のマークルツリーを有すること、例えば、各マークルツリーが、独立して完全な全体として保護されるべき完全性をもつ同じコホート (c o h o r t) に属するデータを含むか、又は、各マークルツリーが特定の完全性保護要件のためのパラメータの異なるセットをもつ、 MT_1 、 MT_2 、…を有することを可能にすることである。また、例えば、 k 個のマークルツリーをマージして、1 レベル深い k 個のマークルツリーのルートのハッシュとして計算された MT 全体のルート、 $MTR_Overall = Hash (MTR_1 | MTR_2 | \dots | MTR_k)$ をもつ 1 つの大きいツリーにすることによって、ネストされたマークルツリーをサポートすることが可能である。

【0139】

実施形態 12：関数構成要素とデータ構成要素とのためのマークルツリーへの分割

10

実施形態 10 に関係する代替実施形態は、関数構成要素とデータ構成要素とを別個のマークルツリー階層に編成し、次いで、それらのマークルツリー階層を結合して新しいルート全体を形成することである。関数構成要素は、異なるコンテナレベルにある、ヘッダ構造と、メタデータ構造と、保護構造とを含み、データ構成要素は、ペイロードブロックデータを含んでいる構造に対応する。関数構成要素は、一般に重要であり、サイズが小さいので、データ構成要素の選択的な保護を可能にしながら、すべての機能構成要素が、選択なしにマークルツリーによって保護されるようにすることが妥当であると思われる。そのような構成の 1 つの潜在的な利点は関数構成要素のより速い検証である。例えば、データセットのメタデータを検証するために、データセット中の個々のアクセスユニットのハッシュコードは検証に関与しない。データ構成要素のためのマークルツリーのルートにあるハッシュコードのみが検証のために使用される。以下の例は、関数構成要素とデータ構成要素とを 2 つの別個のマークルツリーに分割し、次いで、それらを結合して単一のルート全体にするアイデアを示している。

20

MTR_MPEGG

$MTR_Functional$

$MTR_Functional_DG_1 = Hash (Header_DG_1 | Metadata_DG_1 | Protection_DG_1 | MT_Functional_DT_11 | MT_Functional_DT_12 | \dots)$

$MTR_Functional_DG_2 = Hash (Header_DG_2 | Metadata_DG_2 | Protection_DG_2 | MT_Functional_DT_21 | MT_Functional_DT_22 | \dots)$

30

...

MTR_Data

$MTR_Data_DG_1$

$MTR_Data_DG_2$

...

【0140】

この例では、ルート全体 MTR_MPEGG は $Hash (MTR_Functional | MTR_Data)$ としての関数マークルツリーとデータマークルツリーとのルートに依存する。 $MTR_Functional$ は、 $Dataset$ $Group$ レベルにある関数 MT ルートの連結上でのハッシングによって取得される。 $MTR_Functional_DG$ によってプレフィックスされているそのようなルートは、 $Dataset$ $Group$ $Header$ と、 $Metadata$ と、 $Protection$ と、 $Dataset$ レベルにある下位関数 MT ルートとの連結上でのハッシングによって生成される。ルート値を生成する際に、ハッシュ関数は、複数の入力と同じレベルにあるデータ構成要素を取るの、これは K 進マークルツリーに似ている。

40

【0141】

実施形態 13：タイムスタンプ

マークルツリーによる完全性保護の鮮度を示すために各マークルツリーデータ構造中にタイムスタンプを含めることができる。タイムスタンプは、最上位マークルツリーのル

50

トMTR__MP EGGとともに記憶され、署名される。さらに、MTデータの期限切れ前に、更新されたタイムスタンプを用いてMTルート全体上の署名が自動的に再生成されるように、鮮度周期を課することができる。期限切れタイムスタンプは、ファイルが最新でないことがあり、現行バージョンになりすまし、最近の変更を取り消すために侵入者によって使用されたより古いバージョンであり得ることを示す。

【0142】

代替として、いくつかのデータ構造が特定の日付によって特徴付けられているとき、その日付はそれらのデータ構造中の入力として（例えば追加のリーフとして）使用される。例えば、ユーザが数日の間に患者のゲノムデータに関するいくつかの分析を実行したと仮定する。時間の異なる瞬間（異なる時、異なる日）に変更された複数のデータ構造中に分析が反映されている場合、データ構造の各々は、マークルツリーの新しい階層全体を構築するときに異なるタイムスタンプを有することがある。ユーザは、MTR__MP EGGに署名するときにタイムスタンプを含め得る。署名のタイムスタンプは、好ましくは最新のタイムスタンプである。

【0143】

実施形態14：トランスポートモードのための使用

MP EGGはデータの記憶及びトランスポートのための選択肢を定義している。一実施形態では、データをトランスポートするとき、メッセージも完全性保護することができる。

【0144】

データ構造、例えばデータのブロックが送られるときに、メッセージは以下から始まり得る。

データ構造／データのブロックの完全性を検証するために必要とされるマークルツリー中のノード、及び

ファイル署名。

【0145】

これにより、最初にファイル全体を受信する必要なしに、ファイルの一部である、データのブロックの完全性とデータのブロックとを検証することが可能になる。

【0146】

メッセージは、例えば、これが最初のメッセージである場合、MTR__MP EGG上の署名を含み得る。より後のメッセージは、再びこの署名を含むことは冗長であるので、その必要がない。この署名はまた、最初のメッセージについて検証されるだけでよい。このことは、データのn個のブロックがクライアントによってサーバから取り出される、図6に示されている。データのこのn個のブロックはn個のメッセージ中で交換される。最初のメッセージは、ファイル署名と、（受信機においてまだ利用可能でない場合）署名者の公開鍵と、データの最初のブロックを検証するために必要とされるマークルツリー経路に関与するノードと、データのブロック自体とを含む。クライアントがこの情報を受信すると、クライアントは、受信されたデータ（ブロック1）上のハッシュを計算し、生成されたハッシュと経路中の必要とされるノードとを使用してツリーのルートを計算し、生成されたルートと署名者の公開鍵とを使用して署名を検証する。

【0147】

図6にはクライアント610とサーバ620とが示されている。メッセージ1（630）は、署名（631）と、ブロック1を検証するために必要とされるマークルツリー経路に関与するノード（631）と、ブロック1（633）とを含む。メッセージ2（640）は、ブロック2を検証するために必要とされるマークルツリー経路に関与するノード（641）と、ブロック2（642）とを含む。メッセージ3（650）は、ブロック3を検証するために必要とされるマークルツリー経路に関与するノード（651）と、ブロック3（652）とを含む。

【0148】

後で受信されたメッセージの処理は同様であるが、署名はファイル全体について一意で

10

20

30

40

50

あるので、それ以上署名を含める／検査する必要がない点が異なる。

【0149】

代替ソリューションベースとの比較

上記の実施形態は、MPEG-Gファイルの長期的な完全性保護を達成するために、これらのデータ構造中のマークルツリーを使用する。代替ソリューションは、23092-3:2019-3、7.4中の現在のMPEG-Gソリューションを変更／拡張することを含む。

【0150】

例えば、代替ソリューションは、長期的な署名アルゴリズムの使用、及び署名されたコンテナを結合することを可能にするプロシージャを定義することという2つの変更を含み得る。

10

【0151】

第1の点は、ECDSAの代わりに、耐量子アルゴリズム、例えば、LMS、XMSS又はSPHINCSなど、ハッシュベースの署名アルゴリズムを使用することによって対処することができる。

【0152】

第2の点に関しては、異なる手法を検討することができる。以下では、3つの選択肢について説明する。

1. 第1の手法は、ファイルヘッダ中に、一意のファイル識別子、例えばランダムに生成された256ビット長の識別子を含めることを含む。このファイルヘッダはシステム管理者によって署名される。ファイル中のすべてのデータコンテナを結び付けるために、MPEG-G仕様は、コンテナが署名されると、この署名されたコンテナが一意のファイル識別子を含むようなやり方で更新される。これが行われた場合、ユーザがファイルからデータコンテナを取り出すときに、ユーザは以下を行う必要がある。

20

署名メインファイルヘッダを取り出し、検証し、ファイル識別子ID__fileを抽出する。

データコンテナを取り出し、その署名を検証し、データコンテナ中のIDがID__fileに等しいことを検査する。

2. 前の手法は、異なるMPEG-Gファイル中のデータがマージされ得るものである。2つのファイルがマージされると、新しいファイル識別子を含む新しいファイルヘッダが作成される。ファイルヘッダは、前のファイルの識別子を含むマージ履歴を含み得る。これが行われた場合、ユーザがファイルからデータコンテナを取り出すときに、ユーザは以下を行う必要がある。

30

署名メインファイルヘッダを取り出し、検証し、ファイル識別子ID__file、並びに前のマージされていないファイルからの前のファイル識別子を抽出する。

データコンテナを取り出し、その署名を検証し、データコンテナ中のIDが、ID__file、又は、前のマージされていないファイルからのID__fileのうちの1つに等しいことを検査する。

3. 別の代替手法は、各コンテナ中に一意の長い識別子（例えば、ランダムに生成された256ビット長の識別子）を有することを含む。コンテナが署名されたとき、署名は親コンテナの識別子を含み得る。コンテナの署名が検証されるべきである場合、以下のプロセスに従う。

40

コンテナの署名を検査し、親コンテナのIDを取り出す

親コンテナのIDがMPEG-Gヘッダファイルに対応するまで、以下を繰り返す

署名を検査する

親コンテナのIDを取り出す

4. 別の選択肢は、セクション7.4中のMPEG-G Part 3において指定されている保護ボックスを使用し、それらを結合するプロセスを定義することである。例えば、アクセスユニット中のデータのブロックが検証されるべきであるとき、以下のステップのうちの1つ又は複数或いはすべてが実行され得る。

50

1. 所与のブロックのためのアクセスユニット保護中にデジタル署名が存在しなければならない。
2. この第1の署名の検証が成功しなければならない。
3. データセット保護ボックス中のデータセットレベルにある所与のアクセスユニットのためのアクセスユニット保護の署名がなければならない。
4. この第2の署名の検証が成功しなければならない。
5. データセットグループ保護ボックス中のデータセットグループレベルにある所与のデータセットのためのデータセット保護ボックスの署名がなければならない。
6. この第3の署名の検証が成功しなければならない。

【0153】

ステップのいずれかが失敗した場合、完全性検証プロセスは失敗する。ステップが成功した場合、次のステップが評価される。

【0154】

すべての上記のステップが個々に実行される場合でも、現在のMPEG-G規格には、それらを結合するこの検証プロセスが欠けている。さらに、すべてのデータセットグループを結合するファイル全体上の署名の定義はない。

【0155】

上記の方法を、マークルツリーを使用する実施形態と比較すると、マークルツリーを使用する実施形態がより効率的で包括的である。それにはいくつかの理由がある。

- a) MTベースのソリューションにより、個々の構成要素の完全性だけでなく、それらがMPEG-Gファイル全体の一部として一緒に属することをも検証することが可能になる。それはファイル中のデータ構造の不正な追加、削除又は並べ替えを防ぐ。
- b) これの第1の理由は、MPEG-G手法が、コンテナごとに署名を記憶することを必要とし、ハッシュベースの署名（例えば、SPHINCS）が大きいためである。例えば、Bernsteinらによる論文「SPHINCS: practical stateless hash-based signatures」を参照されたい。
- c) 各コンテナのためのハッシュベースの署名は単独で検証される必要があるので、計算オーバーヘッドははるかに大きくなる。

【0156】

GA4GHファイル中の完全性保護を保証するための適用例

Global Alliance for Genomics and Health (GA4GH) は、その論文（「GA4GH File Encryption Standard」、2019年10月21日）において、その後に変換される64キロバイトの個々のブロックをどのように暗号化し、完全性保護するかについて説明している。ブロックの各々が暗号化され、メッセージ認証コード（MAC）が追加される。しかしながら、そのソリューションは、攻撃者が通信中にブロック全体を挿入し、削除し、又は並べ替えることを防がない（このことはCrypt4GH、セクション1.1に述べられている）。

【0157】

この問題を処理するための手法は、64キロバイトのそれらのブロックの各々がリーフである、マークルツリーを定義することを含む。これは実施形態1中のレベル1と等価である。データのブロックはまた、実施形態2の場合と同様に、個々の小さいマークルツリーであり得る。

【0158】

マークルツリーのルートは署名され得るか、又は代替的に、Crypt4GHの場合と同様に、同じ鍵とMACアルゴリズムとを使用してMACを計算することができる。

【0159】

64キロバイトのn個のブロックがあると仮定する。データのブロックが送信されるべきであるとき、以下の情報が送られるべきである。

- a) マークルツリールート上の署名（又はMAC）、及び

- b) データのブロックを検査することを可能にする、バイナリマークルツリー中の対応する $\log(n)$ 個のノード（これはブロックインデックスとマークルツリーのノード ID とを含む）、
- c) データの暗号化され、MAC 保護されたブロック（Crypt 4 gh によれば、最大 64 キロバイトのサイズ）。

【0160】

データをトランスポートするときの潜在的な断片化問題により、ブロックサイズが $B = 64$ キロバイトになるように選定される場合、上記のポイント a) 及び b) 中の追加の完全性データが考慮され得ることに留意されたい。このことは、ブロックの合計サイズが最大で $B = 64$ キロバイトになり得、データブロックは、a) 及び b) によって生じるオーバーヘッドにより幾分小さくなり得ることを意味する。

【0161】

メッセージを受信すると、受信当事者は、マークルツリー中の受信された $\log(n)$ 個のノードを使用して、マークルツリーのルートを再計算し、データのブロックの位置を特定する。次いで、受信当事者はマークルツリー上の署名を検査する。最後に、受信当事者はデータの受信されたブロックを検査する。ルート署名は最初のメッセージについて検証するだけでよいことに留意されたい。これは実施形態 14 の場合と同様である。

【0162】

Crypt 4 GH、セクション 1.1 中の記述にもかかわらず、Crypt 4 GH の現在の規格はまた、例えば、データのブロック中にインデックスが含まれる場合に、データブロックの不正な挿入、削除又は並べ替えを防ぐために、何らかの完全性保護を提供し得ることを確認されたい。この特徴は独立して重要である。このインデックスはブロックの相対位置を特定する。このインデックスが Crypt 4 GH における各交換ブロック中に含まれた場合、それ以上ブロックを並べ替えることは不可能になる。受信機は、このインデックスに基づいて重複がないことを検査することもできる。受信機が、インデックス k をもつブロックを受信した場合、受信機は、それが、ブロックインデックス 1、2、…、 $(k-1)$ をもつすべてのブロックを受信したかどうかをも検査し得る。

【0163】

この最後の「インデックスベースの」手法は MT ベースの手法とは異なる。1 つの利点は、「インデックスベースの」手法は、データがどのように送られるかについてのものであるが、MT 手法は、データがどのように記憶されるかについての情報をも与えることである。このことは、攻撃者が、パケットを送るプロセスに影響を及ぼすことができる場合、攻撃者は、インデックスを正しい順序で配置しても、ブロックを入れ替え得ることを意味する。受信当事者は、その場合、間違った順序でファイルをアSEMBルすることがある。MT は、ブロックがどのように編成され、ファイル中に記憶されるかについての情報を与えるので、MT ベースの手法ではこのことは実行不可能である。

【0164】

図 7 a は、デジタルデータ構造中のデータを符号化するための符号化方法 700 の一実施形態の一例を概略的に示している。本符号化方法は、

データを複数のデータブロックとして取得するステップ (705) と、

複数のデータブロックにハッシュ関数を適用することによって、複数のデータブロックのための複数のハッシュ値を計算するステップ (710) と、

複数のハッシュ値についての第 1 のハッシュツリーを計算するステップ (715) であって、複数のハッシュ値が第 1 のハッシュツリーのリーフに割り当てられ、第 1 のハッシュツリーの 1 つ又は複数のより高いレベルが生成される、第 1 のハッシュツリーを計算するステップ (715) と、

データ構造中に複数のデータブロックと第 1 のハッシュツリーの一部とを含めるステップ (720) であって、第 1 のハッシュツリーの前記一部が第 1 のハッシュツリーのリーフの少なくとも一部を含まない、データ構造中に複数のデータブロックと第 1 のハッシュツリーの一部とを含めるステップ (720) と

10

20

30

40

50

を有する。

複数のゲノムデータブロックを取得するステップは、ゲノムデータブロックを直接受信するステップを有し得る。

【0165】

図7bは、デジタルデータ構造中の選択されたゲノムデータを検証するための検証方法750の一実施形態の一例を概略的に示している。検証方法750は、

データ構造の少なくとも一部を受信するステップ(755)であって、データ構造が、複数のデータブロックとハッシュツリーの一部とを含み、第1のハッシュツリーのリーフの少なくとも一部を含まない、データ構造の少なくとも一部を受信するステップ(755)と、

データ完全性検証のために選択されたデータ構造中のデータブロックにハッシュ関数を適用することによって、選択されたデータブロックのための複数のハッシュ値を計算するステップ(760)と、

検証のために選択されたデータブロックから始まる、対応するハッシュツリーのルートまでの経路を特定するステップ(765)と、

データ構造中で利用可能な場合に、経路に沿ったハッシュツリーのためのハッシュ値を取り出すステップ(770)、又はそのようなハッシュ値を計算するステップと、

少なくとも計算された複数のハッシュ値からハッシュツリーのルートを検証するステップ(775)と

を有する。

【0166】

当業者に明らかになるように、本方法を実行する多くの異なるやり方が可能である。例えば、ステップの順序は、示されている順序で実行することができるが、ステップの順序は変更することができるか、又は、いくつかのステップは並行して実行され得る。その上、ステップ間に他の方法のステップが挿入され得る。挿入されたステップは、本明細書で説明するような本方法の改良を表すこともあり、本方法に関係しないこともある。例えば、いくつかのステップは、少なくとも部分的に並行して実行され得る。その上、所与のステップは、次のステップが開始される前に完全に完了していないことがある。

【0167】

本方法の実施形態は、プロセッサシステムに方法700及び／又は750を実行させるための命令を含む、ソフトウェアを使用して実行され得る。ソフトウェアは、システムの特定のサブエンティティによって取られるステップのみを含み得る。ソフトウェアは、ハードディスク、フロッピー、メモリ、光ディスクなど、好適な記憶媒体に記憶され得る。ソフトウェアは、ワイヤを通して、又はワイヤレスで、又はデータネットワーク、例えばインターネットを使用して、信号として送られ得る。ソフトウェアは、ダウンロードのために及び／又はサーバ上での遠隔使用のために利用可能にされ得る。本方法の実施形態は、本方法を実行するために、プログラマブル論理、例えばフィールドプログラマブルゲートアレイ(FPGA)を構成するように構成されたビットストリームを使用して実行され得る。

【0168】

現在開示している主題はまた、コンピュータプログラム、特に、開示されている本主題を実践するために適応させられた、キャリア上の又はキャリア中のコンピュータプログラムにまで及ぶことが諒解されよう。プログラムは、ソースコード、オブジェクトコード、コード中間ソース、及び部分的にコンパイルされた形態などのオブジェクトコードの形態であるか、又は、本方法の一実施形態の実装において使用するために好適な任意の他の形態であり得る。コンピュータプログラム製品に関する一実施形態は、記載されている方法のうちの少なくとも1つの処理ステップの各々に対応するコンピュータ実行可能命令を含む。これらの命令は、サブルーチンに再区分され、及び／又は、静的に若しくは動的にリンクされる1つ若しくは複数のファイル中に記憶され得る。コンピュータプログラム製品に関する別の実施形態は、記載されているシステム及び／又は製品のうちの少なくとも1

10

20

30

40

50

つのデバイス、ユニット及び／又は部品の各々に対応するコンピュータ実行可能命令を含む。

【0169】

図8aは、書込み可能な部分1010を有するコンピュータ可読媒体1000と、同じく書込み可能な部分を有するコンピュータ可読媒体1001とを示している。コンピュータ可読媒体1000は光可読媒体の形態で示されている。コンピュータ可読媒体1001は、電子メモリ、この場合はメモリカードの形態で示されている。コンピュータ可読媒体1000及び1001はデータ1020を記憶し、そのデータは、プロセッサシステムによって実行されたときに、プロセッサシステムに、一実施形態による、ゲノムデータを符号化又は検証するための方法の一実施形態を実行させる命令を示す。コンピュータプログラム1020は、コンピュータ可読媒体1000の物理的マークとして、又はコンピュータ可読媒体1000の磁化によって、コンピュータ可読媒体1000上で実施される。しかしながら、任意の他の好適な実施形態も想到できる。さらに、コンピュータ可読媒体1000は本明細書では光ディスクとして示されているが、コンピュータ可読媒体1000は、ハードディスク、ソリッドステートメモリ、フラッシュメモリなど、任意の好適なコンピュータ可読媒体であり得、記録不可能又は記録可能であり得ることが諒解されよう。コンピュータプログラム1020は、データを符号化及び／又は検証する前記方法をプロセッサシステムに実行させるための命令を含む。

10

【0170】

図8bは、符号化システム及び／又は検証システムの一実施形態による、プロセッサシステム1140の概略表現を示している。プロセッサシステムは1つ又は複数の集積回路1110を備える。図8bには、1つ又は複数の集積回路1110のアーキテクチャが概略的に示されている。回路1110は、一実施形態による方法を実行するための及び／又はそのモジュール又はユニットを実装するためのコンピュータプログラム構成要素を動作させるための処理ユニット1120、例えば、CPUを備える。回路1110は、プログラミングコード、データなどを記憶するためのメモリ1122を備える。メモリ1122の一部は読取り専用であり得る。回路1110は、通信要素1126、例えば、アンテナ、コネクタ又は両方などを備える。回路1110は、本方法において定義する処理の一部又は全部を実行するための専用集積回路1124を備える。プロセッサ1120、メモリ1122、専用IC1124及び通信要素1126は相互接続1130、例えばバスを介して互いに接続される。プロセッサシステム1110は、それぞれアンテナ及び／又はコネクタを使用する、接触型及び／又は非接触型通信のために構成され得る。

20

30

【0171】

例えば、一実施形態では、プロセッサシステム1140、例えば符号化又は検証システムは、プロセッサ回路とメモリ回路とを備え、プロセッサは、メモリ回路に記憶されたソフトウェアを実行するように構成される。メモリ回路は、ROM回路、又は不揮発性メモリ、例えばフラッシュメモリであり得る。メモリ回路は、揮発性メモリ、例えばSRAMメモリであり得る。後者の場合、デバイスは、ソフトウェアを与えるために構成された、不揮発性ソフトウェアインターフェース、例えば、ハードドライブ、ネットワークインターフェースなどを備え得る。

40

【0172】

デバイス1140は、各説明された構成要素のうちの1つを含むものとして示されているが、様々な構成要素は様々な実施形態において重複し得る。例えば、プロセッサは、本明細書で説明する方法を単独で実行するように構成された複数のマイクロプロセッサを含むか、或いは、複数のプロセッサが本明細書で説明する機能を達成するように協働するように、本明細書で説明する方法のステップ又はサブルーチンを実行するように構成された複数のマイクロプロセッサを含む。さらに、デバイス1140がクラウドコンピューティングシステム中に実装される場合、様々なハードウェア構成要素は別個の物理システムに属し得る。例えば、プロセッサは、第1のサーバ中の第1のプロセッサと、第2のサーバ中の第2のプロセッサとを含む。

50

【0173】

本発明は以下のさらなる実施形態を含む。

【0174】

実施形態1．データ構造中のデータを符号化するための符号化システムであって、本符号化システムは、

データを受信するための入力インターフェースと、

プロセッサシステムと

を備え、プロセッサシステムが、

データを複数のデータブロックとして取得することと、

複数のデータブロックにハッシュ関数を適用することによって、複数のデータブロックのための複数のハッシュ値を計算することと、

複数のハッシュ値についての第1のハッシュツリーを計算することであって、複数のハッシュ値が第1のハッシュツリーのリーフに割り当てられ、第1のハッシュツリーの1つ又は複数のより高いレベルが生成される、第1のハッシュツリーを計算することと、

データ構造中に複数のデータブロックと第1のハッシュツリーの全部又は一部とを含めることであって、第1のハッシュツリーの前記一部が第1のハッシュツリーのリーフの少なくとも一部を含まない、データ構造中に複数のデータブロックと第1のハッシュツリーの全部又は一部とを含めることと

を行う、符号化システム。

【0175】

実施形態2．プロセッサシステムが、第2のハッシュツリーを計算することであって、第1のハッシュツリーのルートが第2のハッシュツリーのリーフに割り当てられ、複数のさらなるハッシュ値が第2のハッシュツリーの複数のさらなるリーフに割り当てられ、さらなるハッシュ値が、さらなるハッシュツリーのルートとして生成される及び／又はさらなるデータブロック上でのハッシングによって生成される、第2のハッシュツリーを計算することと、データ構造中に第2のハッシュツリーの少なくともルートを含めることとを行う、実施形態1に記載の符号化システム。

【0176】

実施形態3．データ構造が、コンテナの階層中に編成されたデータブロックを含み、プロセッサシステムが階層中の各コンテナのためのハッシュツリーを計算し、ハッシュツリーの各リーフが、コンテナ内のデータブロックのハッシュ値、又は下位のコンテナに対応するハッシュツリーのルートのいずれかである、実施形態1又は2に記載の符号化システム。

【0177】

実施形態4．プロセッサシステムが、ハッシュツリーのルート、特に、ハッシュツリーのリーフの間のハッシュツリールートに有するハッシュツリーのルートにわたるデジタル署名を計算することと、デジタル署名をデータ構造中に含めることとを行う、実施形態1から3のいずれか1つに記載の符号化システム。

【0178】

実施形態5．プロセッサシステムが、データ構造を記憶すること及び／又はデータ構造若しくはデータ構造の一部をストリーミングすることを行い、データ構造の前記一部が、データブロックの少なくとも一部と、前記データブロックに対応するハッシュツリーの少なくとも一部とを含む、実施形態1から4のいずれか1つに記載の符号化システム。

【0179】

実施形態6．複数のデータブロック及び／又はデータコンテナのサブセットが完全性保護付きと標示され、複数のデータブロック及び／又はデータコンテナの残りが、完全性保護なしと標示され、完全性保護付きと標示された部分のみがハッシュツリー中に含まれる、実施形態1から5のいずれか1つに記載の符号化システム。

【0180】

実施形態7．入力インターフェースが1つ又は複数のツリーパラメータを受信し、プロ

10

20

30

40

50

セッサシステムが、1つ又は複数のツリーパラメータに応じて、1つのハッシュツリー又はハッシュツリーの階層から選択されたノードのセットをデータ構造中に含める、実施形態1から6のいずれか1つに記載の符号化システム。

【0181】

実施形態8．入力インターフェースがデータに対する修正を受信し、修正が、追加、削除、及び／又は変更のうちの1つ又は複数を含み、プロセッサシステムが、修正を適用することと、データの修正された一部に対応するハッシュツリーの一部を選択的に再計算し、更新することとを行う、実施形態1から7のいずれか1つに記載の符号化システム。

【0182】

実施形態9．ハッシュツリーのリーフが、非圧縮形態のデータブロックのハッシュと、圧縮形態のデータブロックのハッシュとをさらに含み、データブロックが圧縮形態でデータ構造中に含まれるか、又は、

10

ハッシュツリーのリーフが、非圧縮及び非暗号化形態のデータブロックのハッシュと、圧縮及び非暗号化形態のデータブロックのハッシュと、圧縮及び暗号化形態のデータブロックのハッシュとをさらに含み、データブロックが圧縮及び暗号化形態でデータ構造中に含まれる、実施形態1から8のいずれか1つに記載の符号化システム。

【0183】

実施形態10．データ構造中の選択されたデータを検証するための検証システムであって、本検証システムは、

データ構造の少なくとも一部を受信するための入力インターフェースであって、データ構造が、複数のデータブロックとハッシュツリーの一部とを含み、第1のハッシュツリーのリーフの少なくとも一部を含まない、入力インターフェースと、

20

プロセッサシステムと

を備え、プロセッサシステムが、

データ完全性検証のために選択されたデータ構造中のデータブロックにハッシュ関数を適用することによって、選択されたデータブロックのための複数のハッシュ値を計算することと、

検証のために選択されたデータブロックから始まる、対応するハッシュツリーのルートまでの経路を特定することと、

データ構造中で利用可能な場合に、経路に沿ったハッシュツリーのためのハッシュ値を取り出すこと、又はそのようなハッシュ値を計算することと、

30

少なくとも計算された複数のハッシュ値からハッシュツリーのルートを検証することと

を行う、検証システム。

【0184】

実施形態11．データ構造がハッシュツリーの階層を含み、プロセッサシステムが、リーフから始まる、ハッシュツリーの階層のルート全体までの経路を特定する、実施形態10に記載の検証システム。

【0185】

実施形態12．符号化及び／又は検証システムがデバイスである、実施形態1から11のいずれか1つに記載の符号化及び／又は検証システム。

40

【0186】

実施形態13．複数のデータブロックがゲノムデータを含む、実施形態1から12のいずれか1つに記載の符号化及び／又は検証システム。

【0187】

実施形態14．データ構造中のデータを符号化するための符号化方法であって、本符号化方法は、

データを複数のデータブロックとして取得するステップと、

複数のデータブロックにハッシュ関数を適用することによって、複数のデータブロックのための複数のハッシュ値を計算するステップと、

50

複数のハッシュ値についての第1のハッシュツリーを計算するステップであって、複数のハッシュ値が第1のハッシュツリーのリーフに割り当てられ、第1のハッシュツリーの1つ又は複数のより高いレベルが生成される、第1のハッシュツリーを計算するステップと、

データ構造中に複数のデータブロックと第1のハッシュツリーの一部とを含めるステップであって、第1のハッシュツリーの前記一部が第1のハッシュツリーのリーフの少なくとも一部を含まない、データ構造中に複数のデータブロックと第1のハッシュツリーの一部とを含めるステップと
を有する、符号化方法。

【0188】

実施形態15．データ構造中の選択されたデータを検証するための検証方法であって、本検証方法は、

データ構造の少なくとも一部を受信するステップであって、データ構造が、複数のデータブロックとハッシュツリーの一部とを含み、第1のハッシュツリーのリーフの少なくとも一部を含まない、データ構造の少なくとも一部を受信するステップと、

データ完全性検証のために選択されたデータ構造中のデータブロックにハッシュ関数を適用することによって、選択されたデータブロックのための複数のハッシュ値を計算するステップと、

検証のために選択されたデータブロックから始まる、対応するハッシュツリーのルートまでの経路を特定するステップと、

データ構造中で利用可能な場合に、経路に沿ったハッシュツリーのためのハッシュ値を取り出すステップ、又はそのようなハッシュ値を計算するステップと、

少なくとも計算された複数のハッシュ値からハッシュツリーのルートを検証するステップと

を有する、検証方法。

【0189】

実施形態16．デジタルデータ構造中のゲノムデータを符号化するための符号化システムであって、本符号化システムは、

ゲノムデータを受信するための入力インターフェースと、

プロセッサシステムと

を備え、プロセッサシステムが、

ゲノムデータを複数のデータブロックとして取得することと、

複数のゲノムデータブロック中の少なくともゲノムデータにハッシュ関数を適用することによって、複数のゲノムブロックのための複数のハッシュ値を計算することと、

複数のハッシュ値についての第1のハッシュツリーを計算することであって、複数のハッシュ値が第1のハッシュツリーのリーフに割り当てられ、第1のハッシュツリーが少なくとも3つのレベルを有する、第1のハッシュツリーを計算することと、

データ構造中に複数のゲノムブロックと第1のハッシュツリーの一部とを含めることであって、第1のハッシュツリーの前記一部が、第1のハッシュツリーの最初の2つの最も高いレベルを含むが、第1のハッシュツリーの1つ又は複数のより低いレベルを含まない、データ構造中に複数のゲノムブロックと第1のハッシュツリーの一部とを含めることと

を行う、符号化システム。

【0190】

実施形態17．プロセッサシステムが、第2のハッシュツリーを計算することであって、第1のハッシュツリーのルートが第2のハッシュツリーのリーフに割り当てられ、複数のさらなるデータ項目が第2のハッシュツリーの複数のさらなるリーフに割り当てられる、第2のハッシュツリーを計算することと、データ構造中に第2のハッシュツリーの少なくともルートを含めることとを行う、実施形態1から16のいずれか1つに記載の符号化システム。

10

20

30

40

50

【0191】

実施形態18．データ構造が、複数のレベルを有する階層データ構造であり、プロセッサシステムが、階層データ構造の各レベルのためのハッシュツリーを計算することと、最も低いレベルの上のレベルのためのハッシュツリー中に、階層データ構造のより低いレベルのために計算されたハッシュツリーのルートを含めることとを行う、実施形態1から17のいずれか1つに記載の符号化システム。

【0192】

実施形態19．入力インターフェースがツリーパラメータを受信し、プロセッサシステムが、ツリーパラメータに応じて、データ構造中に第1のハッシュツリーのより大きい又はより小さい一部を含める、実施形態1から18のいずれか1つに記載の符号化システム。

10

【0193】

上述の実施形態は、開示されている本主題を限定するのではなく、例示するものであることと、当業者は、多くの代替実施形態を設計することが可能であろうこととに留意されたい。

【0194】

特許請求の範囲において、丸括弧に入れられた参照符号は、請求項を限定するものとして解釈されないものとする。「備える」及び「有する」という動詞並びにそれらの活用形の使用は、特許請求の範囲に記述された要素又はステップ以外の要素又はステップの存在を除外しない。単数形の要素は複数のそのような要素の存在を除外しない。「のうちの少なくとも1つ」などの表現は、要素のリストの後に付いているとき、リストからのすべての要素又は要素の任意のサブセットの選択を表す。例えば、「A、B、及びCのうちの少なくとも1つ」という表現は、Aのみ、Bのみ、Cのみ、AとBの両方、AとCの両方、BとCの両方、又はA、B、及びCのすべてを含むものとして理解されるべきである。開示されている本主題は、いくつかの別個の要素を含むハードウェアと、好適にプログラムされたコンピュータとによって実施され得る。いくつかの部分を一覧するデバイスクレームでは、これらの部分のうちのいくつかは同一のハードウェアによって実施され得る。相互に異なる従属クレームにおいていくつかの手段が具陳されているという単なる事実は、これらの手段の組合せを有利に使用することができないことを示さない。

20

【0195】

特許請求の範囲において、丸括弧内の参照符号は、例示的な実施形態の図面中の参照符号又は実施形態の公式を指し、それによりクレームの了解度が高まる。これらの参照符号は、クレームを限定するものとして解釈されないものとする。

30

40

50

【図面】
【図 1 a】

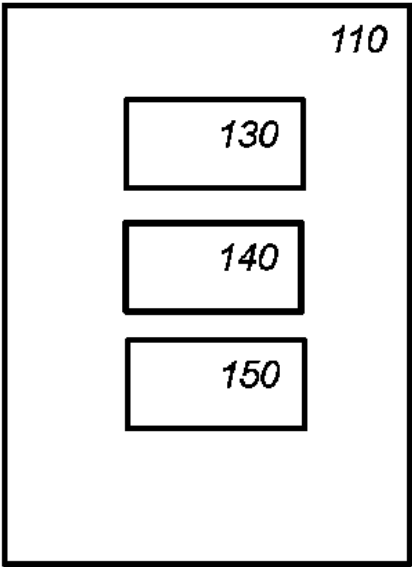


Fig. 1a

【図 1 b】

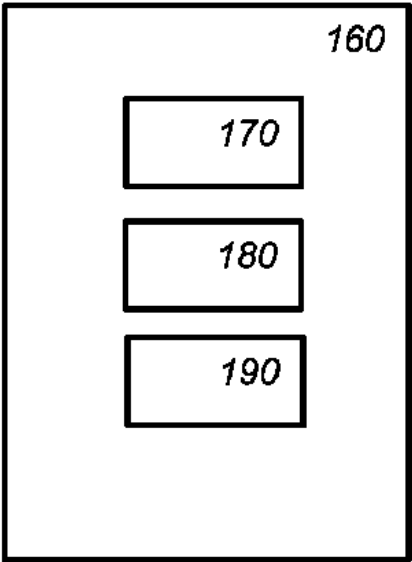


Fig. 1b

【図 2 a】

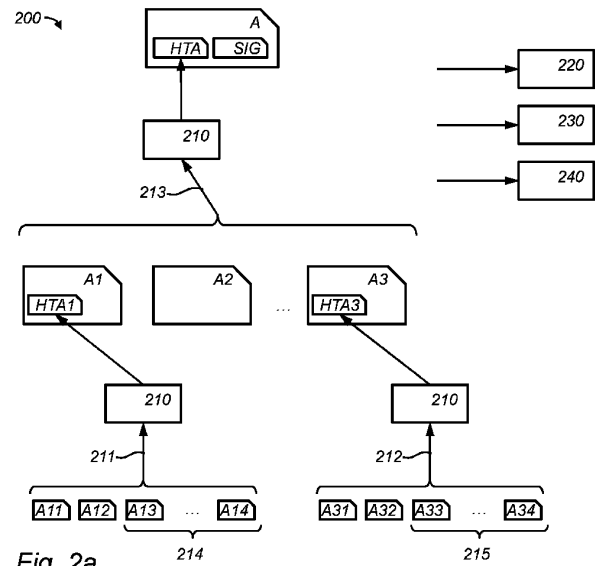


Fig. 2a

【図 2 b】

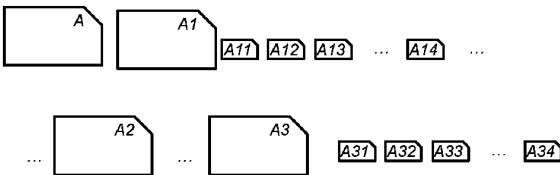


Fig. 2b

10

20

30

40

50

【 図 2 c 】

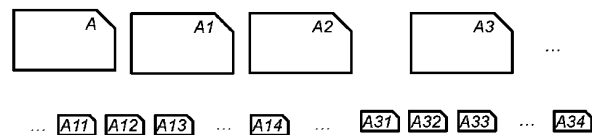


Fig. 2c

【 図 3 】

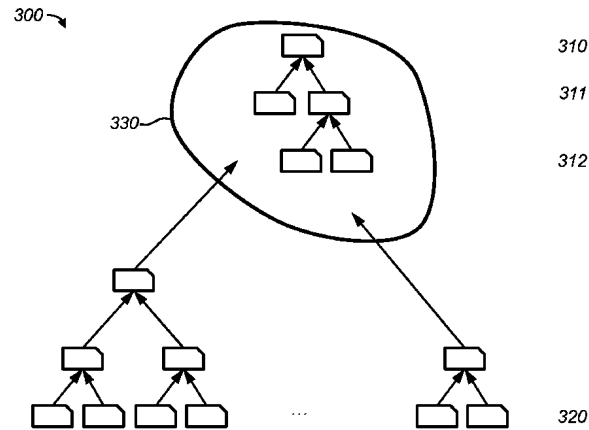


Fig. 3

【 図 4 】

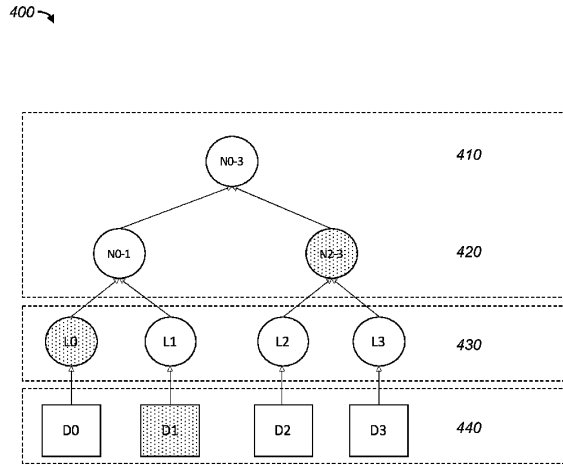


Fig. 4

【 図 5 a 】

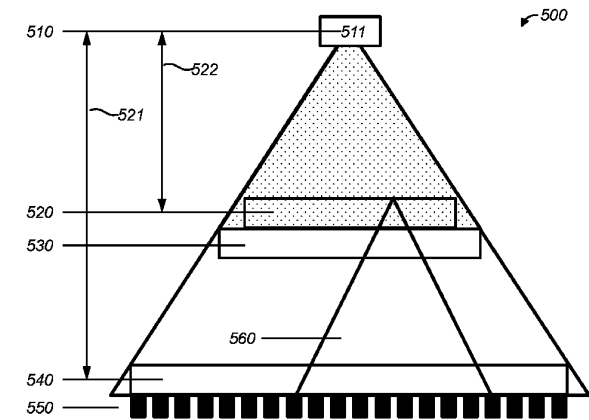


Fig. 5a

10

20

30

40

50

【図 5 b】

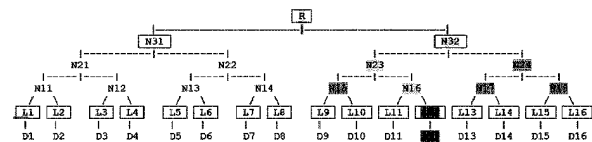


Fig. 5b

【図 6】

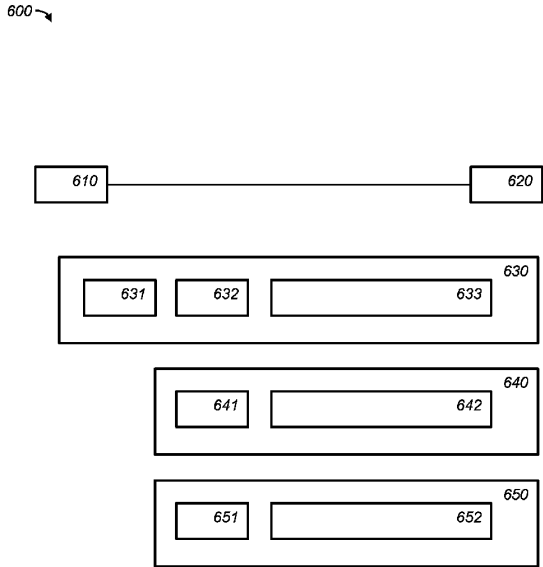


Fig. 6

【図 7 a】

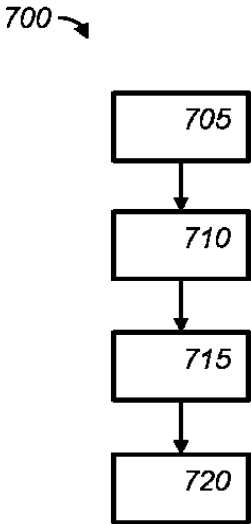


Fig. 7a

【図 7 b】

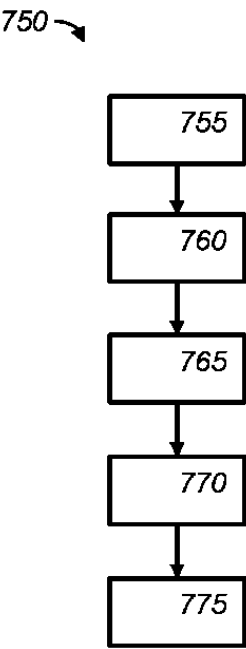


Fig. 7b

10

20

30

40

50

【 図 8 a 】

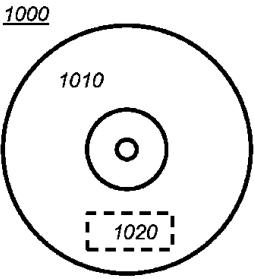


Fig. 8a

【 図 8 b 】

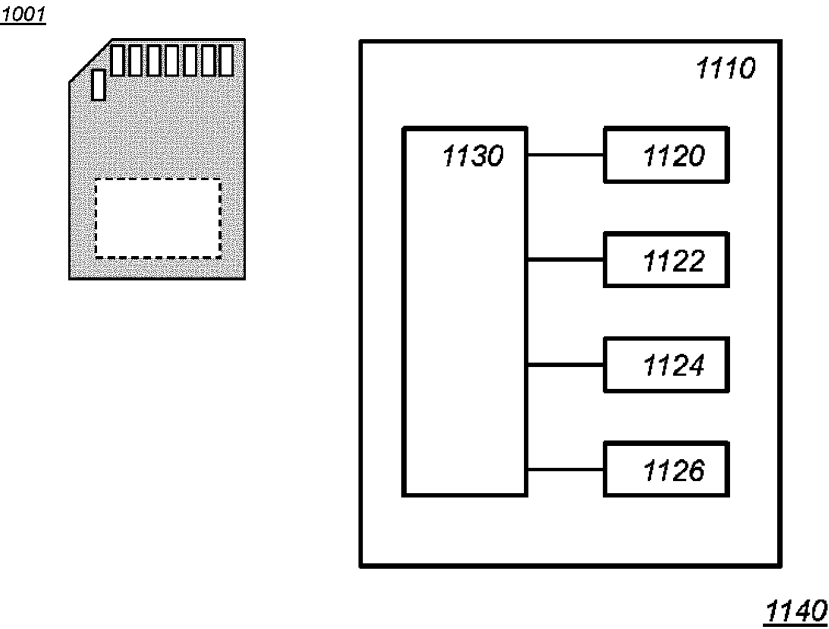


Fig. 8b

10

20

30

40

50

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2022/068316

A. CLASSIFICATION OF SUBJECT MATTER

INV. G16B50/40 H04L9/00 G16B50/30
ADD. H04W12/10 H04L9/40 H04L9/32

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G16B H04L H04W

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BUCHMANN JOHANNES ET AL: "Long-term integrity protection of Check for updates of genomic data", EURASIP JOURNAL ON INFORMATION SECURITY , vol. 2019, no. 1 29 October 2019 (2019-10-29), XP055790384, DOI: 10.1186/s13635-019-0099-x Retrieved from the Internet: URL:http://link.springer.com/article/10.1186/s13635-019-0099-x/fulltext.html the whole document ----- -/--	1-17

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

☒ See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

14 October 2022

Date of mailing of the international search report

21/10/2022

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Martínez Cebollada

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2022/068316

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	BRADLEY TATIANA ET AL: "Genomic Security (Lest We Forget)", SECURITY & PRIVACY, IEEE, vol. 15, no. 5, 2017, pages 38-46, XP011661689, ISSN: 1540-7993, DOI: 10.1109/MSP.2017.3681055 [retrieved on 2017-10-02] the whole document -----	1-17
A	US 4 309 569 A (MERKLE RALPH C) 5 January 1982 (1982-01-05) cited in the application column 2, line 16 - column 4, line 29 -----	1-17

10

20

30

40

50

INTERNATIONAL SEARCH REPORT			
Information on patent family members			International application No PCT/EP2022/068316
Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4309569	A	05-01-1982	NONE

10

20

30

40

50

フロントページの続き

,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,D
K,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),O
A(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,B
B,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD
,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IQ,IR,IS,IT,JM,JO,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,
LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,
RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,ST,SV,SY,TH,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,WS,Z
A,ZM,ZW

(特許庁注：以下のものは登録商標)

1. VERILOG

2 フィリップス インターナショナル ビー． ヴィ． インテレクトチュアル プロパティー アンド
スタンダーズ

(72)発明者 チャン イー ヒム

オランダ国 5 6 5 6 アーヘー アインドーフェン ハイ テック キャンパス 5 2 フィリップス
インターナショナル ビー． ヴィ． インテレクトチュアル プロパティー アンド スタンダーズ

Fターム (参考) 5B175 KA04 KA11