2019年度春学期　卒業論文

ブロックチェーンを利用したデジタル証拠の記録と真偽判別システム

立命館大学　情報理工学部　情報システム学科

W A N G Y U N C O N G

学籍番号：2600150492-3

指導教員：上原哲太郎教授

2019年 8月5 日

目录

[1. はじめに](#_Toc29872)

[2. 背景](#_Toc27637)

[2.1 いろいろなファイル認証手法](#_Toc20639)

[2.1.1 電子公証制度](#_Toc24419)

[2.1.2 タイムスタンプサービス](#_Toc26683)

[2.1.4 ブロックチェーンの利用](#_Toc11255)

[2.2　ブロックチェンの種類](#_Toc3301)

[2.3　イーサリアム](#_Toc20712)

[2.4 python kivy](#_Toc20202)

[2.5 qrcodeの利用](#_Toc15222)

[3. システムの概要](#_Toc13320)

[4.実装](#_Toc31203)

[4.1多重署名smartcontractの作成](#_Toc6925)

[4.2 python kivy で　UIを作る](#_Toc6927)

[4.3 miner serverのたつとqrcodeのscan](#_Toc32125)

[4.3.1 geth](#_Toc18437)

[4.3.2 pythonとblockchainの連携](#_Toc1895)

[4.3.3 qrcode scan](#_Toc20230)

[4.3.4 buildozerでandroid app　apkを生成](#_Toc32455)

[5.実験](#_Toc21459)

[5.1　イーサリアムのsmart contractのテスト](#_Toc32563)

[5.2スマートフォンのテスト](#_Toc14139)

[6. 考察](#_Toc29596)

[6.1 システム](#_Toc6209)

[6.2 欠点](#_Toc18281)

[6.3セキュリティ問題とプライバシー問題](#_Toc9049)

[6.4 今後の仕事](#_Toc20803)

[7. おわりに](#_Toc18563)

[８.謝辞](#_Toc19773)

[9. 参考文献](#_Toc30545)

# 1. はじめに

電子技術の進歩により、デジタルデータは広い範囲で使用されるようになった。その結果、紙で扱われてきた証拠の代わりとして、デジタルの証拠はいまよく使用されます。デジタル証拠は客観性と高科学性を持つ証拠の一つである、裁判実務においても重要な証拠として扱われることもあります。

　しかし、電子情報は簡単に変更または削除されます。電子証拠は、メディアに保存された書き換え可能なデータであり、電子証拠の保存、送信、または使用時に、電子証拠の監視、盗聴、傍受、改ざん、削除などの外部からの影響や損傷を非常に受けやすくなります。 改ざんが容易であるといる認識されます。

法律で要求されている証拠は本物でなければならず、電子的証拠の脆弱性を考慮すると、それは破壊に対して極めて脆弱であり、本物で証拠が完全であるかどうかは様々な関連要因を考慮して検討する必要があります。電子的証拠の信憑性が判断できない場合、その証拠の妥当性を確認することはできず、それを証拠として受け入れることはできません。デジタル証拠は提示した時はプリンタ、それを元のデジタルデータは一致かどうか分からない。警察とかの関係者がデータを読む時は改ざん可能性があります

　以上の理由から、デジタル証拠を改ざんされていないことを証明して、改ざんの有無を確認できるシステムの建築を求められる。

　本論文はブロックチェーンとファイルハッシュ値を利用して、多重署名をつかて、特定のファイルのHASH値をブロックチェーンに載せる。ブロックチェーンのロッグをチェックして、改ざんの有無を表示しますandroid appを作ります。電子公証制度等の既存の制度の問題点について検討を加える。その上で、ブロックチェーン技術とhash値保管システム着目し、便利性も求められますandroid appを作ります。

# 背景

## 2.1 いろいろなファイル認証手法

### 2.1.1 電子公証制度

公証制度は，契約の成立及びその内容を明確にし，また，契約等の成立時期を明らかにして，後の紛争を防止することなどを目的とし，広く国民に利用されています。そこで，電子的なデータについても，従来の書面についての公証と同様の機能を整備するため，政府は，第１４７回通常国会に，「商業登記法等の一部を改正する法律案」を提出しました。この法律案は，平成１２年４月１１日に成立し，４月１９日に公布され，これにより「公証制度に基礎を置く電子公証制度」が創設されました。

電子公証制度は，現在公証人が私署証書について行っている認証や確定日付の付与の事務に対応して，電磁的記録（電子文書）についても，電子公証事務を行う公証人である指定公証人が，電磁的記録の認証，日付情報の付与を行うものである。

電磁的記録の認証 とは、嘱託人（当事者）が，法務大臣の指定を受けた公証人（指定公証人）の面前で，電磁的記録に記録された情報に，電子署名をし，又は電子署名をしたことを自認した場合に，電磁的記録の認証を受けることができることとされています。

日付情報の付与とは、指定公証人が電磁的記録に記録された情報に日付を内容とする情報（日付情報）を付し，これに電子署名をすると、当該情報を確定日付のある証書とみなすこととされています。

電磁的記録の認証と日付情報の付与により認証を受け，又は日付情報を付された情報を保存し，その内容を証明することができることとされています。具体的には，付与された時点での電磁的記録の内容である情報を圧縮して得られた情報（ハッシュ値）が保存されることになります（なお，当該ハッシュ値から元の情報を復元することはできません)。嘱託人，その承継人又は法律上の利害関係を有する者から請求があれば，これらの者が保有する情報と認証を受け，又は日付情報を付された情報の同一性に関する証明が（嘱託人等が保有する情報のハッシュ値と保存されたハッシュ値を比較することによって）行われることとされています。

しかし、この制度はいくつ改善したい所があります。

1まずは金の問題です、手数料は政令（「公証人手数料令」）により次のように定められています。

　＜日付情報の付与＞　⇒　700円  
　＜電磁的記録の認証＞  
　　(1) 私署証書の認証　⇒　11,000円（原則）  
　　(2) 定款の認証　⇒　50,000円（印紙税40,000円は不要）  
　＜電磁的記録の保存＞　⇒　300円  
　＜情報の同一性に関する証明＞　⇒　700円  
　＜同一の情報の提供＞　⇒　700円（書面による交付の場合は、1枚につき20円を加算）

もっと安いコストが欲しいです。

2 手続きが複雑、もっと楽な認証したい。例えば

公証人との連絡

申請に当たっては公証人を指名しなければなりません。  
そのために申請前に公証人と連絡を取ってください。特に定款認証については文章に不備があると設立登記ができなくなる恐れがありますので、文章の内容についても公証人にご相談ください。  
また、嘱託・請求したものの受領の日時なども打ち合わせてください。

それはいつでも空いてわけでもない、それは時間を要します。

３いつでも電子公証制度を利用できわけがない。

電子公証システムの事務取扱時間は月曜日から金曜日（国民の祝日・休日、12月29日から1月3日の年末年始を除く。）の8：30から17：00となっています。ただし、17：00でシステムは稼働を停止しますので、間際に申請された案件は、当日中に処理できない場合があります。

そこでいつでも稼働できるシステムを求めいられる。

4 情報の有効期限があります。

電子公証制度は認証や確定日付を付与した電子文書は20年間保存します。長い保存期限が欲しい。

## 2.1.2 タイムスタンプサービス

タイムスタンプサービスとは、タイムスタンプ局と呼ばれる第三者機関が、利用者が持つデジタルデータがある時刻以前に存在したことを証明する技術であるタイムスタンプ技術を利用したサービスである。

このサービスを利用すれば、電磁的記録のハッシュ値を生成した際に、そのハッシュ値をタイムスタンプ局に送信し、タイムスタンプ要求を行えば、タイムスタンプ局において時刻証明となる情報を添付したタイムスタンプ.トークンを生成し、それにタイムスタンプ局のデジタル署名を施して返送されることから、タイムスタンプに刻印されている時刻以前にその電子文書が存在し、その時刻以降電子文書が改ざんされていないことを証明することができます。

電子公証制度と同じ、データの有効期限があります。タイムスタンプは、タイムスタンプ局の電子署名が施されていることにとってその時刻証明が保証されるが、その保証は、電子認証局がタイムスタンプ局に発行した公開鍵証明書の有効期限に縛られる、有効期限を越えたデータは検証手法がなく、その有効期限を越えて保証されるためにはタイムスタンプを付し直さなければならない。

そしてタイムスタンプが付されたデジタルデータ（電磁的記録のハッシュ値）は、利用者の手元に保管されているだけであり、タイム時刻証明となる情報を一元的かつ集中的に管理、公開するわけではない、情報の同一性を証明した電磁的記録やタイムスタンプが付された情報を公開する場所がない。つまり、電磁的記録と情報の同一性に関する証明を請求するためには、誰でも直接利用が想定されてない、例えば、タイムスタンプ制度において、申請の時は申請書を出します、その申請が認める場合は情報の同一性証明できるだが、警察官や検察官は官職証明書を利用で容易だが、民間の弁護人にっとては身分証明はづらいです。やりたいのは誰でも簡単に情報の同一性証明できるシステムを建築。

現在では専門家なくてもデジタルデータが容易に改ざんする、情報の同一性を証明できないは改ざんのリスクは高くなる、例えば、捜査機関がデジタル証拠を読むとき、改ざん、偽造のリスクがあります、証拠を提示したとき、情報の同一性の証明は非常に重要です。

### 2.1.4 ブロックチェーンの利用

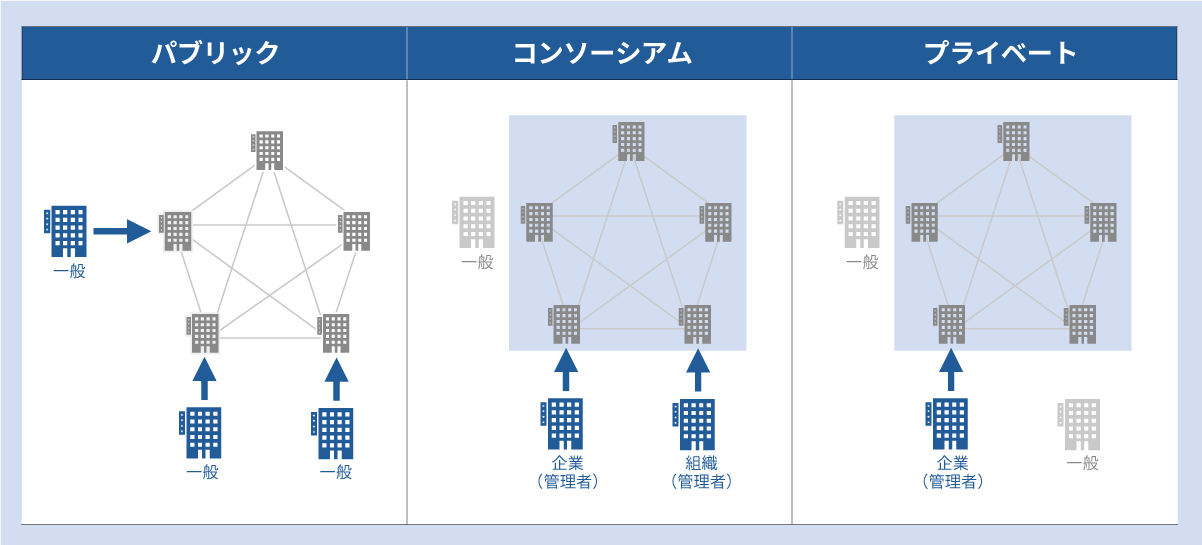
**電子公証制度とタイムスタンプサービスの問題点について、ブロックチェーンを利用するシステムを検討する。**

**ブロックチェーンの改ざんは困難です、ブロックチェーンは、一つ一つブロックが繋がるような仕組みです、ブロックは一つ前のブロック全体のハッシュ値が含まれていますので、一つノートで改ざんがあれば、ネットワークの全ノード間での整合性が保てなくなる、改ざんしたい場合なら、全ネットワークの51パーセント計算能力が必要です、それは事実不可能になります。改ざんは困難という特性はファイルの認証にはポテンシャルがある、さらに、blockchainのユーザーは自分の秘密鍵を持つから、それをつかて署名は容易に実現します。**

**ブロックチェーンは分散システムです、**分散型台帳とも呼ばれるとおり、すべての取引を台帳に記録させ、取引に参加しているすへてのユーサが同じ台帳を保持していることを条件に取引の正当性を確認する仕組みです。そのデータは誰でも読みます。変更履歴が自動的に残され、ユーザ全体で共有されるという透明性が挙げられます。そして参加者１人のデータが消失しても取引データは失われない。P2Pと言う形式ので、特定の機関に負荷が集中するリスクを避けます。そういう分散性を持つから、誰でも、いつでも、どこでも情報を確認することができ。そして、記録はほぼ半永久的に保存している、情報の有効期限は心配いらない。そして、今回は複雑なプログラムを書かない、ブロックチェーンにトランザクションを生成するコストは低い。

## 2.2　ブロックチェンの種類

ブロックチェーンには、大きく分けると3つの型に分類することができます。



パブリックチェーン：特徴は、誰でも参加できるネットワークで、**管理者が存在しない点**です。また、取引が全て公開されており**透明性が高い点**です。

コンソーシアムチェーン：コンソーシアムチェーンは、管理主体がいるブロックチェーンの1つで、**管理主体が複数の企業や組織から成る、パブリックチェーンとプライベートチェーンの中間に位置するブロックチェーン**を指します。

プライベートチェーン：管理者が存在し、ネットワークに参加するためには**管理者の許可が必要なブロックチェーン**です。

では、どのタイプのブロックチェーンの利用が適しているか。

分散性台帳に保全されたデータは、デジタル証拠のhash値や台帳に書き込まれたデータを参照、確認するニーズを有している者は弁護人や被疑者、被告人、あるいは捜査機関や検察等に限られる。そう考えると、広くシステムの利用を一般に開放する必要性はなくなた、プライベートチェーンを選択。

しかし、台帳に記録されるデータは、電磁的証拠それ自体ではなく、あくまでもそのhash値であり、その値自体には特別な意味はない。そして、実際に、利用したデジタル証拠は一般に広く応用したい。システムの利用者を捜査機関や法律関係者に限定しなければならない。実際、改ざんの有無を検証するための情報の開示は、広く一般に公開されているの方がより客観的であり、かつ信頼性も高い。そして、ブロックチェーンのネットワークにより多くのノードに参加し、ブロックが長く続いていくことによる耐改ざん性にある。

以上の点を見て、今回はパブリックチェーンを選択しました。

## 2.3　イーサリアム

パブリックブロックチェーンですが、ビットコインやイーサリアムなど、仮想通貨と呼ばれるモノのほとんどがパブリックブロックチェーンです。

今回はビットコインブロックを使わず、イーサリアムを選択理由は：

1. ビットコインと違い、イーサリアムは「分散型アプリケーション」のプラットフォームのために開発されました。このプラットフォームを実現させたのが「スマートコントラクト」技術です。
2. イーサリアムは時価総額第2位の仮想通貨であり、つまり、利用者が多いと認識さらます。スマートコントラクトの勉強に参照する資料とソースコードが多い。

smart contractとは、その名前の通り、コントラクト（契約）をスマートに行えるプロトコルのことです。つまりスマートコントラクトとは契約の自動化であり、契約の条件確認や履行までを自動的に実行させることができます。

ビットコインはブロックチェーンの技術を用いて悪意のある参加者が参加する可能性のあるP2Pネットワーク上でお金の取引を正しく動作させる環境でした。一方でEthereumは、取引だけでなくアプリケーション（処理）をこのようなP2Pのネットワーク上で正しくに動作させることを可能にする環境を提供しています。これを実現するための機能として、大きな役割を果たすのがEthereum特有の自由度の高い記述ができるスマートコントラクトです。

つまり、ブロックチェーン上で実行するプログラムをつくることができます、実装は容易になります。

イーサリアムのネットワーク上では、ブロックチェーン上に実現したい様々なスマートコントラクトを格納し、ノード間で実行、共有することが可能であるが、そのスマートコントラクトを実行するためのコストとして、Gasと呼ばれる手数料がEther(イーサリアムの暗号通貨)を利用して支払い。また、スマートコントラクトは、160ビットのアドレスで表示されるアカウントと呼ばれる概念で管理される。アカウントには、イーサリアムを利用するユーザが保持する外部所有アカウント(EOA:Externally Owned Account)とスマートコントラクトを表すコントラクトアカウント(CA)の2種類がある。外部所有アカウントは、秘密鍵にとって管理されたアカウントであり、Etherを送金するトランザクションやスマートコントラクトを実行するトランザクションを発行する際にそのトランザクションに秘密鍵で電子署名を付すことでトランザクションの発行が認証される。そして、コントラクトアカウントは、スマートコントラクトをブロックチェーンにデプロイした際に外部所有アカウントからトランザクションを介して生成されるアカウントとしてブロックチェーン上に存在し、外部所有アカウントが発行するトランザクションが発行するトランザクションをトリガーにスマートコントラクトに記述されたコードを実行する。ただし、コントラクトアカウントは秘密鍵は持たない、故に、外部アカウントがトランザクションを介して、コントラクトを使用します。

## 2.4 python kivy

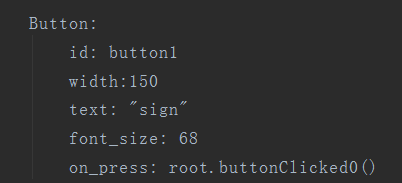
今回はスマートアプリが開発したい、スマートフォンは印鑑のようなsign機能を実現したい。紙証拠は印鑑でsignします、電子証拠は”電子印鑑”でsignしたい。そこで、スマートフォンを”電子印鑑”として機能しと思います。

Android とiosの開発は大きく違います、だから、今回はandroid appの開発を選びます。

Android appの開発は主にJavaです、でも、今回はpython kivyを開発します。Python kivyはアプリケーションを迅速に開発するためのオープンソースのPythonライブラリです、ユーザーインターフェースを利用することです。

Python kivy を選んだ理由は：

1. 簡単　例えば、一つパタンを生成したい、



このパタンの属性を全て並べます、最後はこのようなパタンを生成します

IMG_256

ただ6行程度で一つパタンが作ります。そして、構成は非常に分かり安い。

1. 通用:多くのシステムプラットフォームで支持する。Windows、Linux、MacOS、Android、iPhone. Windows上で開発プログラムは簡単にAndroidに移植できる。実際実装した時もWindowsでテスト後、Androidに移植。
2. 迅速:python言語はインタプリティブ言語。Java言語は[コンパイラー言語](http://dict.asia/jc/%E3%82%B3%E3%83%B3%E3%83%91%E3%82%A4%E3%83%A9%E3%83%BC%E8%A8%80%E8%AA%9E" \o "コンパイラー言語)で。python言語実行の速度が早い。

## 2.5 qrcodeの利用

qrcodeとは、デンソーウェーブが開発した[2次元バーコード](https://kotobank.jp/word/2%E6%AC%A1%E5%85%83%E3%83%90%E3%83%BC%E3%82%B3%E3%83%BC%E3%83%89-6899)の[一種](https://kotobank.jp/word/%E4%B8%80%E7%A8%AE-434200)で、モザイク状の四角いドットで作られている。1方向だけでしか情報を記録できない[バーコード](https://kotobank.jp/word/%E3%83%90%E3%83%BC%E3%82%B3%E3%83%BC%E3%83%89-7132)に対して、QRコード（2次元バーコード）は、縦、横2方向に情報を持つことで、記録できる情報量を飛躍的に増加させた

qrcodeを利用する理由は：

１、大容量データの収納：従来のバーコードが 20桁程度の情報量しか扱うことができなかったのに対し、QRコードは、その数十倍から数 百倍の情報量を扱う事ができます。

またQRコードはあらゆるデータを扱う事も可能です。(数字・英字・漢字・カナ・ひらがな・記号・バイナリ・ 制御コード等)。1つのコードで、最大 7,089文字(数字のみの場合)という大容量を表現できます。

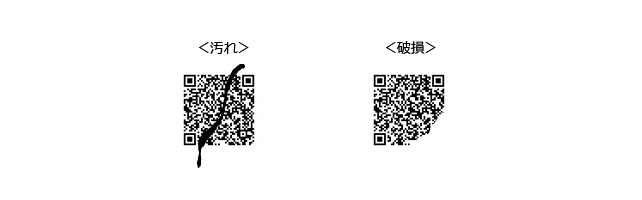
前は言たイーサリアムのアカウントのアドレスは160ビットです、そして、ファイルの認証について、ファイルのhash値を用いるから、256bitのデータ、入力は64個の16進数、例えば、123のhashはa665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3。これをユーザーに入力と非常に不便です。そこで、addressとhashをqrcodeを生成する、データをqrcodeに収納。

２、小スペースへの印字：QRコードは、縦・横両方向でデータを表現しているので、バーコードと同じ情報量であれば、10分の1程度の 大きさで表現できます。(さらに小さなスペースの表現が可能なマイクロ QRコードもサポートしています。)。例えば、hash値

0x9e69e7e29351ad837503c44a5971edebc9b7e6d8601c89c284b1b59bf37afa80(qwert)対応のqrcodeは

３、汚れ、破損に強い：QRコードは、コードの一部に汚れや破損があってもデータの復元が可能な「誤り訂正機能」をもっています。データの復元はコードワード単位で、最大約30%が訂正可能となっています。hash値のデータは汚れまたは破損が発生すると、元のデータは復元できない、例えば、hash値

0x9e69e7e29351ad837503c44a5971edebc9b7e6d8601c89c284b1b59bf37afa80はなのhash値が復元不可能。でも、QRコードの場合は、こんなたら復元可能

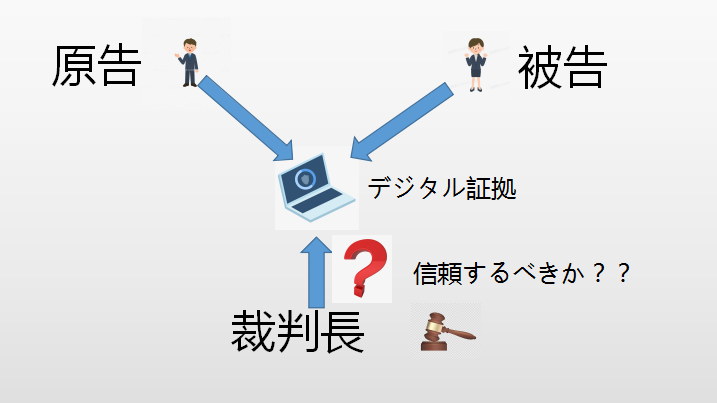


４、360°どの方向からでも読み取り可能：QRコードは、360°どの方向からでも、高速な読み取りが可能です。ユーザにとっては、スマートファンで読みとりは基本アイデアです、keyboardをつかずに高速な読み取りが可能です、そして正確性も保証されます。

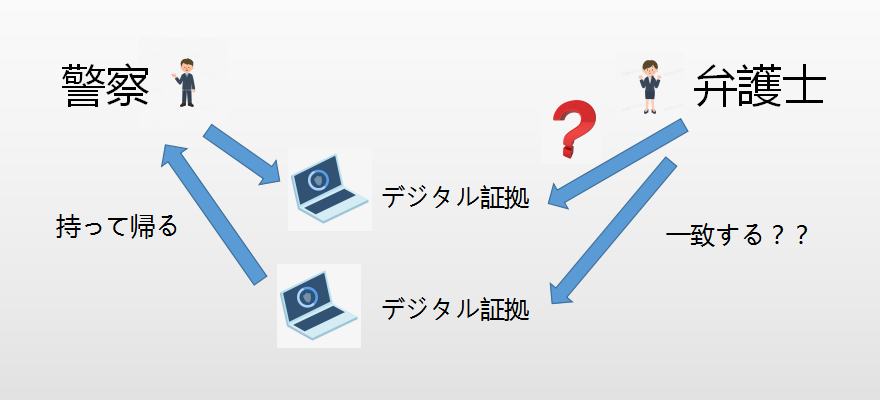
# システムの概要

実際の民事裁判の状況を分析する、システムを紹介します。

解決したい問題：

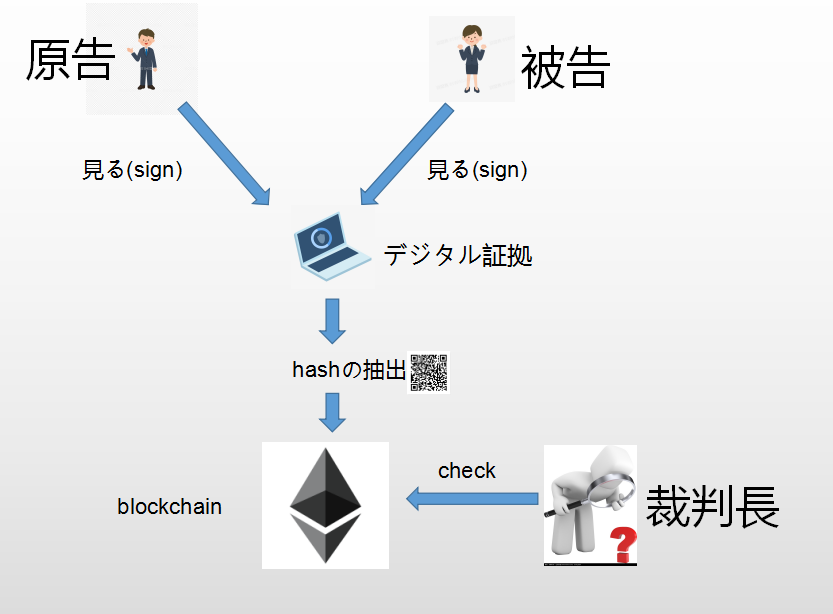


**裁判長はこのデジタル証拠は本当に信頼するべきか？**



返たデジタル証拠は元のデジタル証拠は一致するのか？

システム：



１、両方は原告と被告

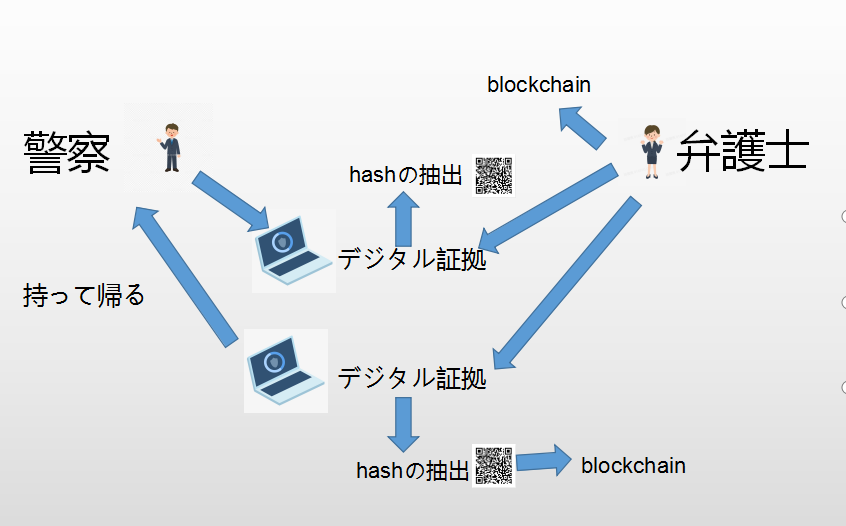
２、両方の意見が一致したら、hashを生成します。

３、hashの抽出は安全性を持つツールで生成。qrcode形式です。

４、signとは、自分を持つblockchainのaccountのaddress、秘密鍵とhashをsignします、hashの入力はqrcode scan。

５、smartcontractを介して,public blockchainに載せる。

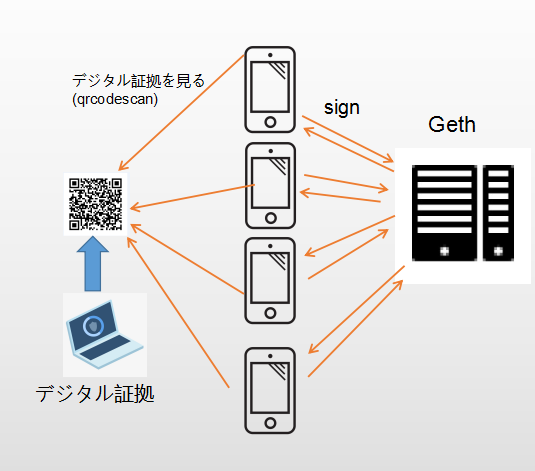
６、checkとは、hashの入力して、blockchainの記録をチェックします。



１、警察が持って帰る前に、先に弁護士とsignします、blockchainに載せる。

２、帰った証拠はhashを生成して、そしてblockchinにcheckします。

# 4.実装



１、イーサリアムのsmartcontractを作ります。そして

Gethにdeploy

すると、signなどのblockchainの機能を実現します。

２、python kivy をつかて、UIとqrcode scan

機能を作ります。

３、スマートフォンは容量の制限がある、

blockchainに参加するにはサーバが必要です

Ubuntu server18.04をつかて

サーバを作りました、中にはGethが動かす。

４、Web3.pyを使用する、

pythonとblockchainに繋がることができます。

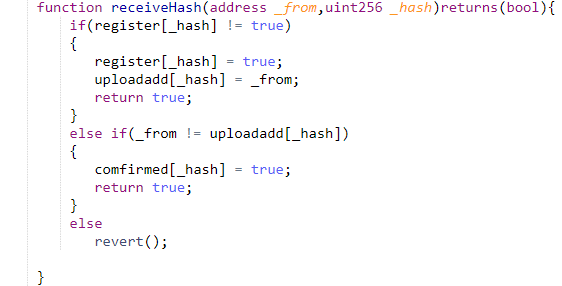
## 4.1多重署名smartcontractの作成

今回は三つの機能を有するスマートコントラクトを作ります。

1. signの機能：ユーザが自分のaddress、秘密鍵(今回はサーバで入力)、とデジタル証拠による生成したhash値を入力、signを実現。
2. checkの機能：ユーザがcheckしたいのデジタル証拠による生成したhashを入力、true or falseを返た値により、このデジタル証拠が改ざんされていないを知る。
3. deleteの機能：ユーザが一度signしたデジタル証拠の痕跡を削除（実際のトランザクション削除ではない）、このhash値にmappingするtrueはfalseに変更。

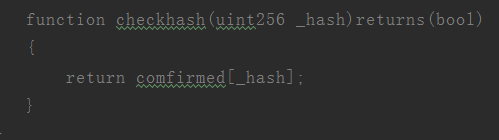
イーサリアムのスマートコントラクトを用いた言語はsolidityである。

<https://remix.ethereum.org/> remixは、soldiity開発のIDEでWeb版が提供されており、誰でも使うことができます。多重署名smartcontractのコードは三つの関数が構成されます。今回使用のはsolidity v0.4.23



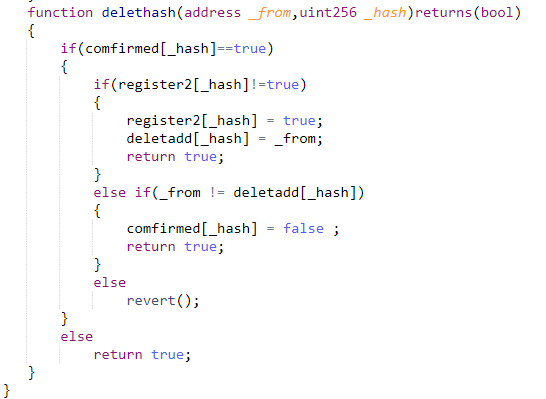
　　　　　　　　　　　receiveHash関数

1. receiveHash関数はsign機能を実現



　　　　　　　　　　Checkhash関数

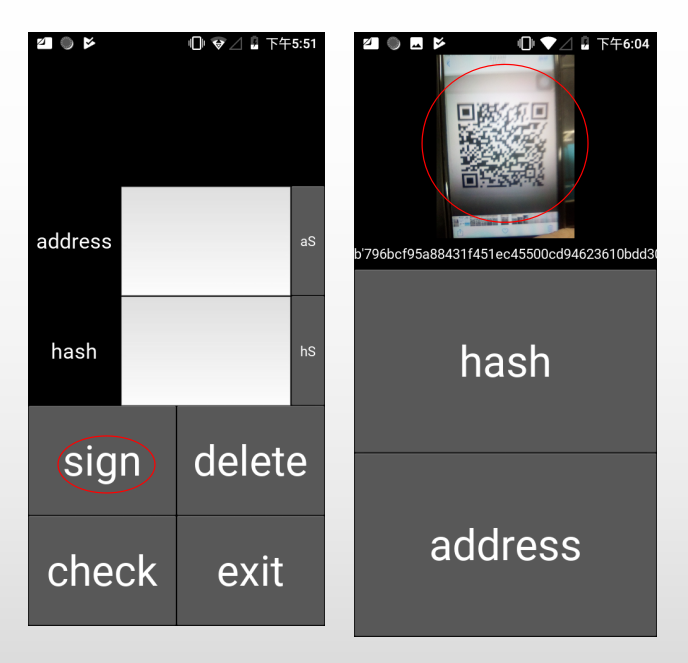
1. checkhash関数は　check機能を実現



delethash関数

1. delethash関数は、　delete機能を実現。

## 4.2 python kivy で　UIを作る



1. ユーザーはblockchainのaddressとhash値を入力した後、signパタンを押すと、signが完了。今回の実装はtestnetで動かすから、秘密鍵はサーバ側で入力した。実際publicnetで使用した時は、秘密鍵の入力が必要です。
2. addressとhashの入力はすべてqrcode scanで入力します。

## 4.3 miner serverのたつとqrcodeのscan

python kivyでUIを建築終わたあと、機能を付けます。

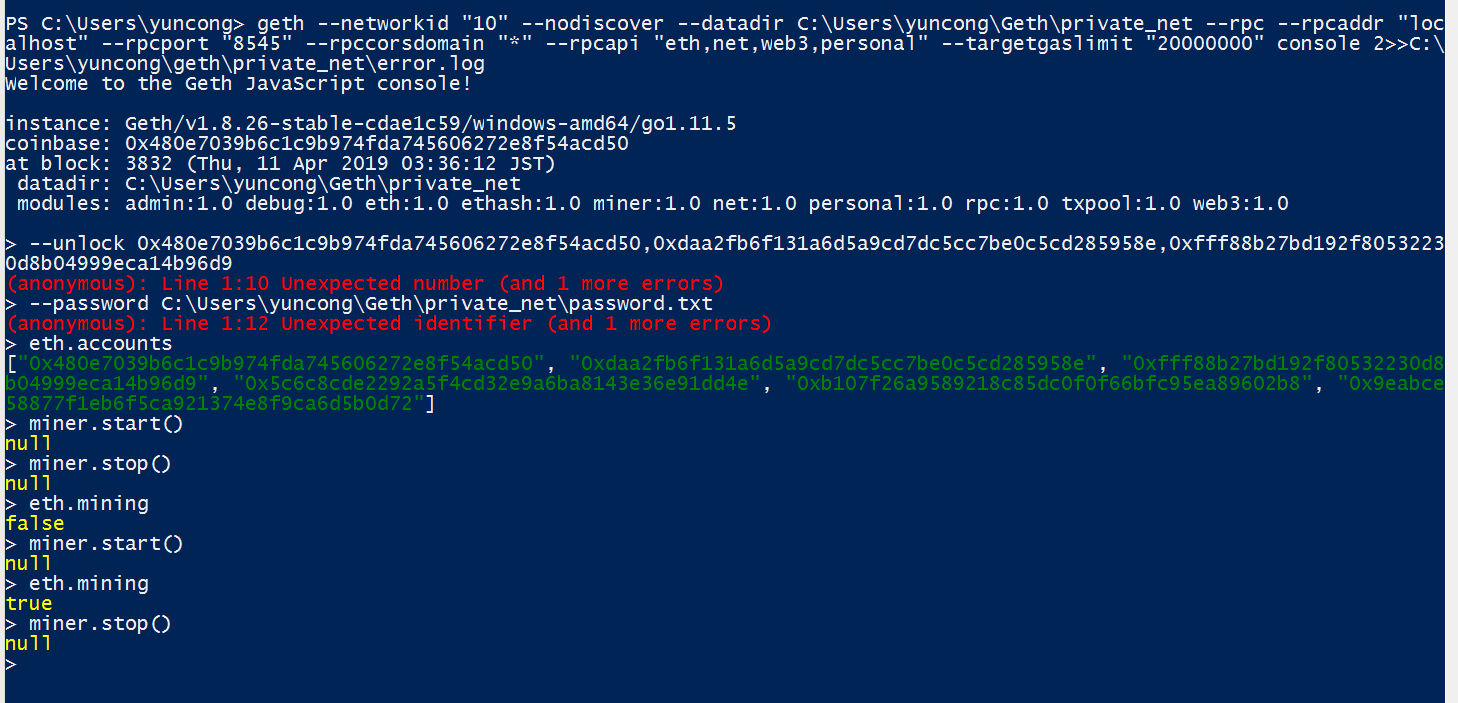
### 4.3.1 geth

実際トランザクションが生成するにはgasと呼ばれるcostがかかります、そうなると、マイニングが必要です。本論文はgethを利用します。

Ethereumを利用する場合、まずはEthereumのP2Pネットワークに参加する必要があります。ネットワークへの参加はEthereumクライアントをインストールし起動することで参加が可能になります。

Ethereumでは、Ethereumの仕様を実装した幾つかのEthereumクライアントが存在しますが、現在のところ推奨されているクライアントは「Geth」です。Gethはプログラミング言語[Go](http://golang.org/" \t "https://book.ethereum-jp.net/first_use/_blank)により実装されたCUIクライアントであり、GethをインストールすることでEthereumネットワークにフル・ノードとして参加.

Geth v1.11.5の起動



miner.start():minerの起動

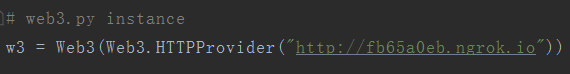
miner.stop():minerの停止

今回はtestnetで実装しますから、accountは事前に作らたものであり、実験に便利ために、秘密鍵は起動時は入力完了しました。本当の場合は秘密鍵はサーバ側は知らないのまま、ユーザーが自分の秘密鍵で入力します、ユーザーは自分の秘密鍵でトランザクションをサインします。

### 4.3.2 pythonとblockchainの連携

　　　gethが起動したあと、appとgethはどうやて繋がる。appはweb3.pyを用いた.web3.pyはEthereumと対話するためのPythonライブラリ。

今回はweb3.py v4.9.2を使用します



　　　　　　　　　　　　gethとの遠隔接続

gethを動かすサーバのipを入力すると、簡単に接続されます。

接続が成功した場合は、gethの各機能が使えます。



　　　　　　　　スマートコントラクトのデプロイ

デプロイは mutisign.constructor().transact()そしてデプロイのスマートコントラクトのアカウントのaddressを返す。

IMG_256

　　　　　　　　コントラクトのreceiveHash関数の使用

receiveHash関数はaddressとhashが必要です。

IMG_256

　　　　　　　　　　コントラクトのcheckhash関数の使用

checkHash関数はhashのパラメ—タ—が必要です。

### 4.3.3 qrcode scan

　　　今回はkivy.garden.zbarcamというライブラリを利用します。

IMG_256

　　　　　　Qrcodeの解析

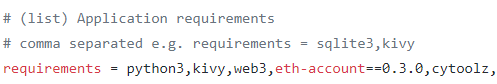
pil\_imageはqrcodeのimage。

Pyzbar.decodeはこのpil\_imageを解析のもの。

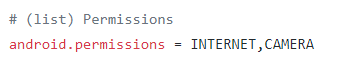
codesの値はqrcodeのデータ,すなわちブロクチェーンアカウントのaddressやファイルのhash値。

### 4.3.4 buildozerでandroid app　apkを生成

windowsで実行成功な場合は成功した後は、androidに移植が行います。kivyアプリをandroidに移植する時は、buildozerと言う自動化ツールを利用します、buildozer.specと言う設置に関するファイルがあります。



requirementsを設置します、例えば、pythonのライブラリ



Androidの権限を与える、例えば、ネットワークとカメラ権限を与える

設置終わた後、buildozer android debug deploy run 命令を押すと、apkが自動的に作ります。

# 5.実験

## 5.1　イーサリアムのsmart contractのテスト

test data:

address: 0xca35b7d915458ef540ade6068dfe2f44e8fa733c

hash: 0xa665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3 (123)

receivehash関数(sign機能)

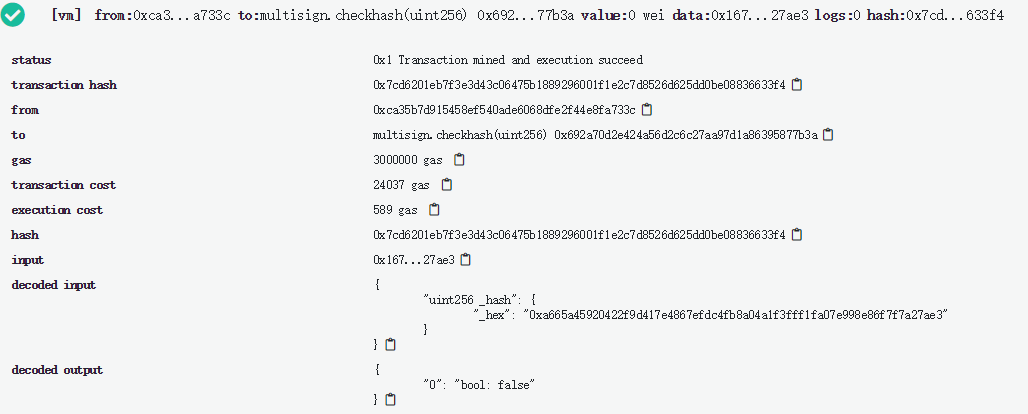


約100000gasかかる、大体0.000000000000435595eth、つまりほぼ0円です。

test data:

hash:0xa665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3 (123)

checkhash関数(check機能)



約25000gas

## 5.2スマートフォンのテスト

実際の民事裁判を模擬、テストを行います。

原告の場合：

test data:

address1:0x9b38398caace8b1321261234dd10974e21eda867

hash1:0x9e69e7e29351ad837503c44a5971edebc9b7e6d8601c89c284b1b59bf37afa80(qwert)



sign　　　　(原告)

結果はsignが成功しました。時間は10**s**

被告の場合：

test data:

address2:0x370dd0c2da65995746c9ec85b78f4925b8df11de

hash1:0x9e69e7e29351ad837503c44a5971edebc9b7e6d8601c89c284b1b59bf37afa80(qwert)

sign

　　結果はsignが成功しました。　　時間は**40s**

裁判長の場合：

test data:

hash1:0x9e69e7e29351ad837503c44a5971edebc9b7e6d8601c89c284b1b59bf37afa80(qwert)

check

結果はこのデジタル証拠は正しいと判断した。時間は**1s**

結果の分析：まず原告と被告がデジタル証拠に生成するhash値を自分を持つ秘密鍵をsignしました。

そして裁判長がそのデジタル証拠に生成するhash値をcheckします、しかし、この場合は秘密鍵とaddress一切必要ない、ただでhash入力すると、checkができました、それは、誰でもcheckです。

サーバと接続しますから、時間はネートの速度に関連して、でも、大体1min以下になります。そして、今回はtestnetで実験しますから、実際にpublicnetに時間は変わります。

でも、電子公証制度とタイムスタンプサービスに比べるは十分早い。

# 考察

## 6.1 システム

システム全体としての観点から：

1.原告と被告が審判に証拠を提出する際の信頼性テストにシステムが使用される、証拠の信頼性を保証するために相手方の文書に署名することができる。そして裁判官両当事者によって提出された文書の信憑性を検証することも可能です。

2.同時に、変更不可能な文書を他の人に渡す必要があるときは、改ざんを防ぐために、文書の真正性を検査によって検証することができる。

3.それぞれのアップロード、署名、およびチェックは、トランザクションレコードの形式でブロックチェーンに記録され、ファイルの署名レコードは、トランザクションのハッシュ値を通して追跡することができます。ブロックの不可逆的な変更により、記録は保証されます。

4.システムは、ブロックチェーンアカウントの公開鍵と秘密鍵を使用してアカウント名とパスワードとして機能し、セキュリティも保証され、同時にファイルはハッシュ値の形式でマッピングされます。ファイルも保護されており、実行はより効率的です容量を気にする必要はありません同時にブロックチェーンの公開鍵と秘密鍵はユーザー名とパスワードより匿名性があります。

5.ブロックチェーンは分散システムであるため、サーバーのシャットダウンや障害を心配することなく、いつでもどこでも誰でもネットワークに参加して認証に参加することができます。

6.公開鍵と秘密鍵はより匿名ですが冗長すぎます、ここではユーザー入力の負担を減らすためにqrcodeスキャン入力方式を使用します。

7.時間のcostは電子公証制度とタイムスタンプサービスより少ない、signと認証のすべてはsmartcontractに任せる、ファイルの電子証明の生成、認証局の証明書などの手続きを一切使用しない。自動化されたプログラムを使用して時間のコストを削減する。

8.金のcostも下がります。実際に使用した金はすべてgasのコストに関する。

本実験のcostは一回uploadとsignとcheckしたら約100000gas、大体0.000000000000435595eth、つまりほぼ0円です。

## 6.2 欠点

しかし、いくつかの問題があります。

1.複数側(三方以上)による承認が必要な文書の中には対応してない。例えば、一つのデジタル証拠は三方でsignしなければならない、両方だけがsignしました、その第三方signにかからわず、このデジタル証拠は承認されます、第三方がこのデジタル証拠をsignしてないでも、デジタル証拠は承認されます、第三方の認証は実際意味ない。

2.いつでも使用できるとは言え、でも、今回の場合はスマートファン、スマートファンにはブロックチェーンノード直接に参加する手段はない(ブロックチェーンの同期は大きいな容量が必要)、ノードに参加したサーバに介して機能します。サーバがある限り、必ず稼働時間に縛られている、そこで運用するコストもいる。

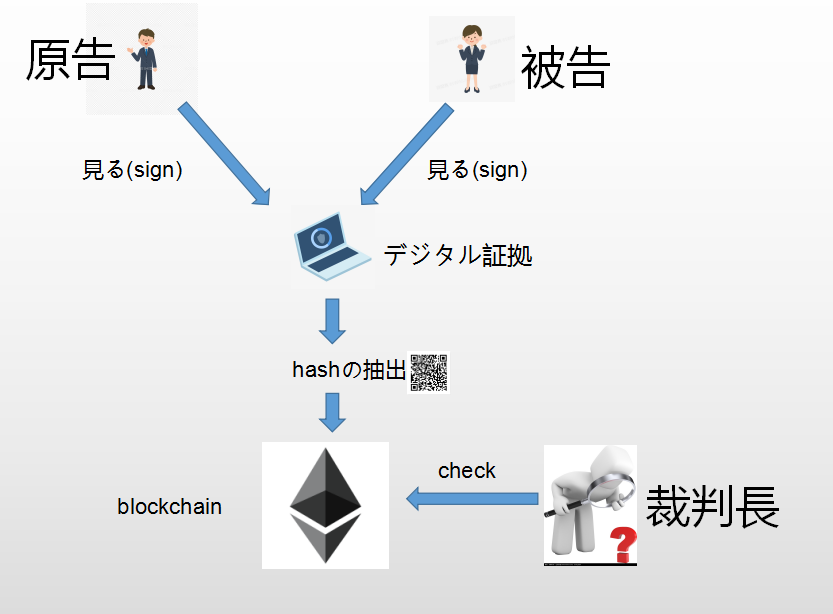
3.詳しい情報は表示されていない、今回の場合は、hash値を入力と、このhash値に対応するデジタル証拠は真実かどうかを表示します。詳しい情報のcheckが欲しい、例えば、signの時間、signの人数、signするコストとか。すべては情報はblockchainのブロックに記録されだが、専門家がない人は簡単にcheckできない。その場合、楽なcheckが求められる。

1. 責任者が見つからない。ユーザーは自分の身分を証明するのは自分だけを持つ秘密鍵である。例えば、signしたデジタル証拠は何らかの原因でトラブルが発生しました、そのsignした人に探す、その探す手段はない、知るのはただのsignしたアカウントのaddressです、もし、このsignした人はこのaddressは僕じゃないと言います、検出手段はない。だから、signをした時は慎重し。

## 6.3セキュリティ問題とプライバシー問題

1. 今回はスマートファンをサーバーに介して機能します仕組みから、ユーザーは自分アカウントのaddress,秘密鍵、デジタル証拠に生成するhash値をサーバーに送信する、その過程は、盗聴が発生する可能性はある、この送信するデータを盗聴すると、秘密鍵が漏らす、それはプライバシーの問題。もし、この秘密鍵をつかて、不正行為を行う場合、例えば、この秘密鍵をつかて、不信デジタル証拠を認証とか。これは安全性問題が発生。
2. qrcodeを利用する時の問題。実際の場合を分析します:

理想の場合はこのhashの抽出は信頼されたツールです、もし、このツールが悪ツールだたら、問題を起こります。例えば、このツールはデジタル証拠改ざんかどうかを問わず、ずっと同じhashを生成する、だたら,もしデジタル証拠が改ざんされたら、checkの時は発見できない。一方、被告は自分の不利な証拠を無効したい、悪ツールの機能は、同じデジタル証拠としても、違ったhash値を生成する、この時checkする結果はこのデジタル証拠は改ざんされている(実際改ざんされてない)、この不利な証拠を無効する。以上の理由から、このhashを生成するツールは必ず安全性を持つこと注意します。



## 6.4 今後の仕事

一つはkivyで勉強を続けたこと、いまのUI画面が非常にシンプルですから、つまり、現時点での写真は非常に単純で、将来的にはインターフェースを華麗にしたいと考えており、それを便利に使用することが私の目標です。

2つ目は、機能には多くの不完全性があることです。今後もプログラムを学び、改善し続けなければなりません。

3つ目は、自分のテストネットワークで自分のプログラムを実行するだけで、ソフトウェア実際運用してかが私の目標です。

今後も問題点について改善する。これらの問題を解決するために努力する必要があります。

# 7. おわりに

　　　本論文では、電子データ認証にいま使われる電子公証制度とタイムスタンプサービスを検討し、blockchainを利用した認証システム提案して、android上で実装します。結果として、ファイルが改ざんされかどうかは検出できる。しかし、実用性はまだ足りない、今後も不足の部分を改善するに努力する。

# ８.謝辞

本論文を書いて間に、指導教員の立命館大学サイバーセキュリティ研究室の上原哲太郎教授に、論文のテーマ決定から、資料収集、書き方、言語表現まで常に丁寧に温かくご指導いただき、心から御礼申し上げます。また、研究室の皆様ご協力いただき、ここに感謝の意を表したい。

# 9. 参考文献

1. 小坂谷聡,上原哲太郎, ブロックチェーンを利用した刑事手続におけるデジタル証拠の改ざん防止システムについての考察，2018
2. 加嵜 長門(著),篠原 航(著),丸山 弘詩 (編集) , ブロックチェーンアプリケーション開発教科書,2018
3. 菊池　浩明、上原哲太郎、ネットワークセキュリティ,2017
4. 日経Linux(編集)、Ubuntuで作る　かんたん　Linux サーバー(日経BPパソコンベストムック)、2013
5. Kivy:Cross-platform Python Framework for NUI Development

<<https://kivy.org/>>

1. Web3.py — Web3.py 4.9.2 documentation

<<https://web3py.readthedocs.io/en/stable/>>

1. kivy-garden/garden.zbarcam

<<https://github.com/kivy-garden/garden.zbarcam>>

1. Youtube sentdex Kivy Application Development

<<https://www.youtube.com/user/sentdex>>

1. ngrok - secure introspectable tunnels to localhost

<<https://ngrok.com/>>

1. kivy/buildozer: Generic Python packager for Android and iOS - GitHub, <<https://github.com/kivy/buildozer>>
2. 7-5 電子公証 | 日本公証人連合会,　<<http://www.koshonin.gr.jp/business/b07_5>>
3. タイムスタンプとは｜セイコーサイバータイム，　　<<https://www.seiko-cybertime.jp/time/>>