2006 年度卒業論文

提出時刻の改竄を防止する Time Stamp システム “S3” の開発

研究指導　菊池浩明　教授

東海大学　電子情報学部　情報メディア学科

3ADM1203 木澤寛厚

目次

第 1 章
はじめに

菊池研究室では、3 年生を対象に Java ゼミナールを行っている。毎週、宿題が出題されるが、期限内に出したと言い張ったり、期限後に秘密に修正するなどの不正行為の懸念があった。

そこで TimeStamp という技術がある。TimeStamp というものは、デジタルデータが特定の時間に存在していたこととその時刻以降改ざんされていないことを証明してくれる技術である(2.1 項)。

TimeStamp には、Simple Protocol と Linking Protocol の 2 種類ある。Simple Protocol は、実装しやすい、分かりやすいという特徴があり、Linking Protocol には、サーバの不正を防ぐことが出来るという特徴がある。本研究では、S. Haber らが提案した Linking Protocol を用いた Time Stamp システムを開発して、運用を行った。

第 2 章

Time Stamp

2.1 TimeStamp とは

例えば、特許を提出する際など公的な機関にデジタルな書類のデータを提出するとき、正確な時刻が必要になる。しかし、文書作成日や提出日などを PC のシステム時計を使って記入しても、これを信じることができない。システム時計は、容易に変更することができるからである。そこで、信頼のできる第三者に、それらの時刻を証明してもらう必要がある。

Time Stamp は、デジタルデータが特定時刻に存在していたことと、その時刻以降、データが改竄されていないことを、信頼できる第三者が証明してくれる技術である。以下は、よく使われている Time Stamp のプロトコルの一つである Simple Protocol の説明である。

＜モデル＞

エンティティを次のように示す

C : クライアント(依頼者)

S : サーバ(発行者)

M : Time Stamp を押したいファイル

V : Time Stamp を検証する人(検証者)

＜プロトコル＞

以下に、クライアント(依頼者)Cが、発行依頼を出し、Time Stamp を押す手順を示す

Step1 : Time Stamp を発行したいファイルのハッシュ h を計算する

$$h=H(M)$$

Step2 : ハッシュ h を、サーバ(発行者)に送信する

Step3 : サーバ(発行者)は、そのハッシュに時刻情報 T を付加し、自分の秘密鍵で署名 σ をする。それを TST と定義する

$$TST=\sigma(h,T)$$

Step4 : TST をクライアント(依頼者)に送信する

以下に、検証者 *V*が、ファイル *M*の Time Stamp を検証する手順を示す

Step1：検証者 *V*は、サーバ(発行者)の公開鍵 *pk* を入手する

Step2：*pk* から、*TST* のデジタル署名の正当性をチェックする

Step3：検証したいファイル *M*のハッシュ *h*'を計算する

$h' = (M')$

Step4：さっき計算した *h*'と、*TST* の中に入っている *h* を比較する

h'=*h* が成立すれば、時刻情報 *T* が証明される

2.2 Time Stamp の特徴

Time Stamp は、第三者に時刻を証明してもらうシステムである。ハッシュを比べることにより、ファイル *M* の完全性を証明できる。デジタル署名により、*TST* 自体の改ざんの有無を確認することが出来る。この二つにより、*TST* の中の時刻情報 *T* が信頼できるものになる。

しかし、以上で説明してきたプロトコル(Simple Protocol)だと、サーバの不正を感知できない。例えば、図 1 に示す不正を検知することが出来ない。

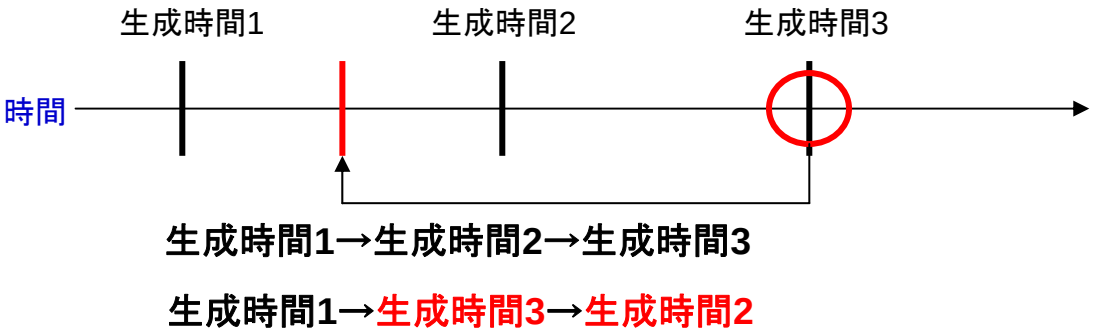


図 1：サーバの不正

2.3 Linking Protocol

2.3.1 Linking Protocol とは

2.1 項で示した一般的な Time Stamp システムでは、2.2 項で示したサーバの不正を防ぐことができない。

Linking Protocol は、2.2 項で示したようなサーバの不正を防ぐことができるプロトコルである。以下に、その概要を示す。

＜モデル＞

エンティティを次のように示す

C : クライアント(依頼者)

S : サーバ(発行者)

M : Time Stamp を押したいファイル

n : サーバ S に来た、Time Stamp 生成依頼の数

M_n : n回目のTime Stamp生成依頼のファイル

S_p : 情報公開サーバ

N : リンク情報を公開する間隔

＜プロトコル＞

以下に、クライアント(依頼者)Cが、発行依頼を出し、Time Stampを押す手順を示すStep1：Time Stampを発行したいファイルのハッシュh_nを計算する

$$h_n=H(M_n)$$

Step2：ハッシュh_nを、サーバ(発行者)Sに送信する

Step3：サーバ(発行者)Sは、リンク情報L_nを生成する

$$L_n=H(h_n,n,L_{n-1})$$

Step4：サーバ(発行者)Sは、そのリンク情報L_nに色々情報を付加し、自分の秘密鍵で署名σをする。それをTST_nと定義する

$$TST_n=\sigma(h_n,T,n,L_n)$$

Step5：サーバ(発行者)Sは、TSTをクライアント(依頼者)Cに送信する

Step6：サーバ(発行者)Sは、L_n・h_n・nを保存する。更に$n=N$のとき、L_nを、情報公開サーバS_pに公開する

以下に、検証者Vが、ファイルM_n'のTime Stampを検証する手順を示す
Step1：検証者Vは、ファイルM_n'のTime Stampが生成される前に既に公開されていたリンク情報の中で、最も新しいリンク情報L_aを、情報公開サーバS_pから得る

Step2：更に検証者 V は、計算に必要な情報を、サーバ S から得る。必要な情報を以下に示す

ハッシュ(h_{a+1}～h_n)
番号(a+1～n)

Step3：手に入れた情報からL_n'を計算する

$L_{a+1}'=H(h_{a+1},a+1,L_a)$
$L_{a+2}'=H(h_{a+2},a+2,\ L_{a+1}')$
$L_{a+3}'=H(h_{a+3},a+3,\ L_{a+2}')$
:
$L_n'=H(h_n,n,L_{n-1}')$

Step4：検証者Vは、サーバSの公開鍵pkを得て、TST_nのデジタル署名を検証する

Step5：Step3 で求めたL_n'とTST_nの中に入っているL_nを比較する。$L_n'=L_n$ならばStep6 へ、$L_n'\neq L_n$ならば、検証失敗

Step6：検証者Vは、ファイルM_n'のハッシュ値h_n'を計算する

$h_n'=H(M_n')$

Step7：Step6 で得たh_n'と、TST_nの中に入っているh_nを比較する。$h_n'=h_n$ならば検証成功、TST_nの中に入っている時刻情報Tが保障される。$h_n'\neq h_n$ならば、検証失敗となる

2.3.2 Linking Protocol の特徴

Linking Protocol の特徴は、サーバの信頼性を上げることにある。リンク情報を生成し公開することにより、サーバ S が、2.2 項で述べたような不正を行うことが出来なくなる。以下に、その理由を示す。

$$
\begin{aligned}
&M_n \rightarrow h_n \quad \rightarrow \quad L_n=H(h_n,n,L_{n-1}) \quad \rightarrow \quad \text{公開}\\
&M_{n+1} \rightarrow h_{n+1} \quad \rightarrow \quad L_{n+1}=H(h_{n+1},n+1,L_{n+0})\\
&M_{n+2} \rightarrow h_{n+2} \quad \rightarrow \quad L_{n+2}=H(h_{n+2},n+2,\ L_{n+1})\\
&M_{n+3} \rightarrow h_{n+3} \quad \rightarrow \quad L_{n+3}=H(h_{n+3},\ n+3,\ L_{n+2})\\
&\qquad\qquad\qquad\qquad :\\
&M_{2n} \rightarrow h_{2n} \quad \rightarrow \quad L_{2n}=H(h_{2n},2n,L_{2n-1}) \quad \rightarrow \quad \text{公開}
\end{aligned}
$$

もし、サーバ S が、不正にファイル M_{n+2} とファイル M_{n+3} の生成時間を入れ替えた後、M_{n+3} を検証すると

$$
\begin{aligned}
&L_{n+1}=H(h_{n+1},n+1,L_{n+0})\\
&L_{n+2}=H(h_{n+2},n+2,\ L_{n+1})\\
&L_{n+3}=H(h_{n+3},\ n+3,\ L_{n+2})
\end{aligned}
$$

このとき、L_{n+3} と TST_{n+3} の中の L_{n+3} を比較すると、必ず値が変わるため検証失敗となる。まず、L_n は、もう既に公開されているため、値を変更することはできない。また、L_{n+2} と L_{n+3} を入れ替えても L_{n+3} のハッシュ値が等しくなるようなことはありえない(一方向ハッシュの衝突耐性)。

第３章
開発システム

3.1　概要
本件級では、開発言語に JavaJDK1.5、開発環境に eclipse3.2 を用いた。

Time　Stampシステム“S3”は、TimeStampサーバS、クライアントツールC、情報公開サーバS_pからなる。

3.2　構成
本ツールは以下で構成されている。

- TimeStampClientToolMain.java
  クライアントツールのメインプログラム
  Time Stamp の生成、検証をすることができる
- TimeStampClientTool.jar
  クライアントツールのクラスファイルをまとめて、ダブルクリックで起動できるようにした
- TimeStampServerMain.java
  サーバのメインプログラム
- TimeStampKey.java
  サーバ S の秘密鍵と公開鍵を生成するプログラム
- PublicInfomationIndex.java
  情報公開サーバのプログラム
  サーブレットで動く

3.3　サーバの準備
(1)サーバプログラムを動かすフォルダの中に、以下のフォルダを作る

１．Count
　サーバSに来た、Time Stamp 生成依頼の数 n を保存するフォルダ
　ファイル名は、countn.txt　(n は自然数)
　更に、count.txt というファイルを作る
　プログラムの仕様上必要なファイルで、中には「0」と入れる

２．Hush
　Time Stampを生成する時に送られてくるハッシュh_nを保存するフォルダ
　ファイル名は、hushn.txt　(n は自然数)

３．Link

Time Stampを生成する時に生成するリンク情報L_nを保存するフォルダ

ファイル名は、linkn.txt （n は自然数）

４．VerifiesServer

リンク情報を公開する間隔 N 回目の生成の時に生成されるファイルを保存するフォルダ

情報公開サーバは、このファイルの存在を確認して公開をする

ファイル名は、verisern.txt （n は自然数）

５．Time

上記のリンク情報を生成した時刻を記録しておくファイルを保存するフォルダ

ファイル名は、timen.txt （n は自然数）

６．KeyPair

TimeStampKey.java を実行したときに生成される秘密鍵と公開鍵のペアが保存されるフォルダ

ファイル名は、TimeStampKeyPair.obj

７．Log

ログファイルを保存するフォルダ

ファイル名は、

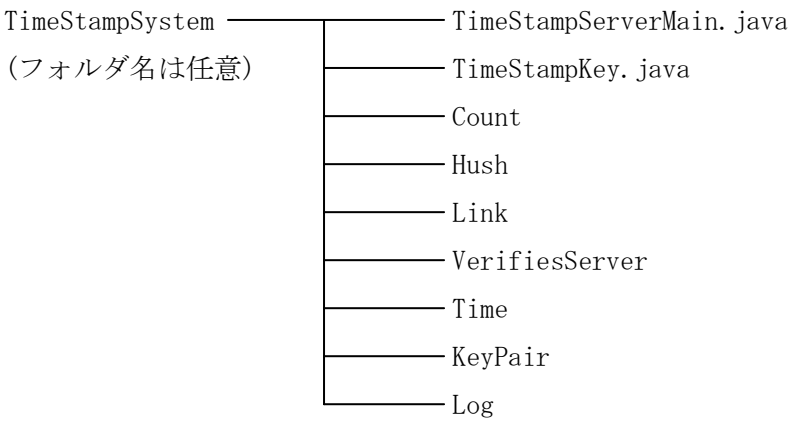
「TimeStampServerMainLog.txt」「TimeStampVerifiesServerLog.txt」

プログラムの仕様上、あらかじめ、以上の二つのファイルを生成しておく必要がある

(2) 以上の 7 つのフォルダを作ったフォルダの中に TimeStampKey.java を入れてコンパイルし実行する。すると、KeyPair フォルダに鍵が生成される

(3)(2)と同じところに TimeStampServerMain.java を入れ、コンパイルし実行する。

※まとめ



3.4　クライアントの使用方法

3.4.1　Time Stamp の生成

Time Stamp を生成する方法は、2 種類ある

１．ボタンで生成する場合

(1)TimeStampClientTool.jar をダブルクリックで実行する

すると、以下に示す図 2 のようなプログラムが起動する

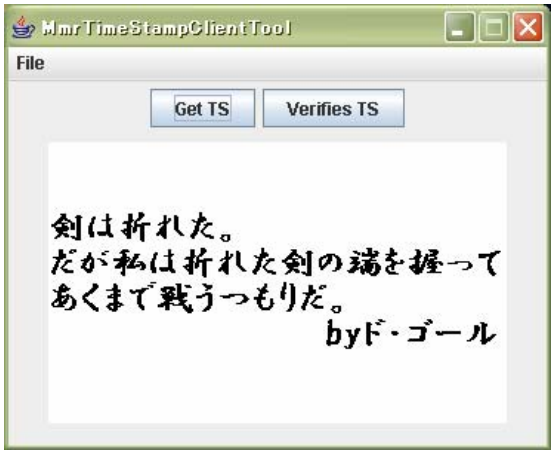


図 2：クライアントツール

(2)「Get TS」ボタンを押す(図 3)

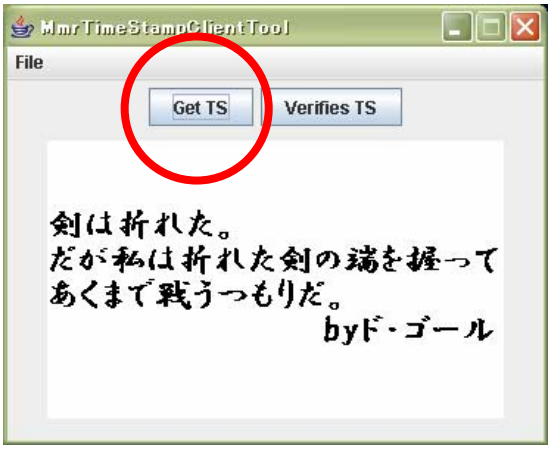


図 3：「Get TS」ボタン

(3)Time Stamp を押したいファイル *M* を選択する

(4)ファイル *M* の *TST* が生成される(図 4・図 5)

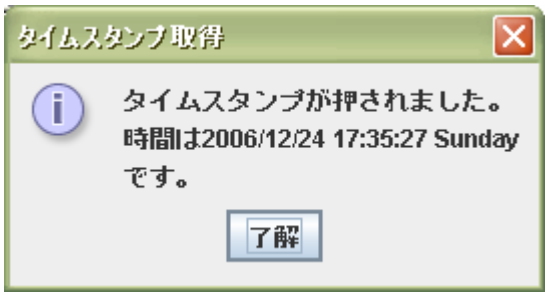


図 4：生成完了ダイアログ

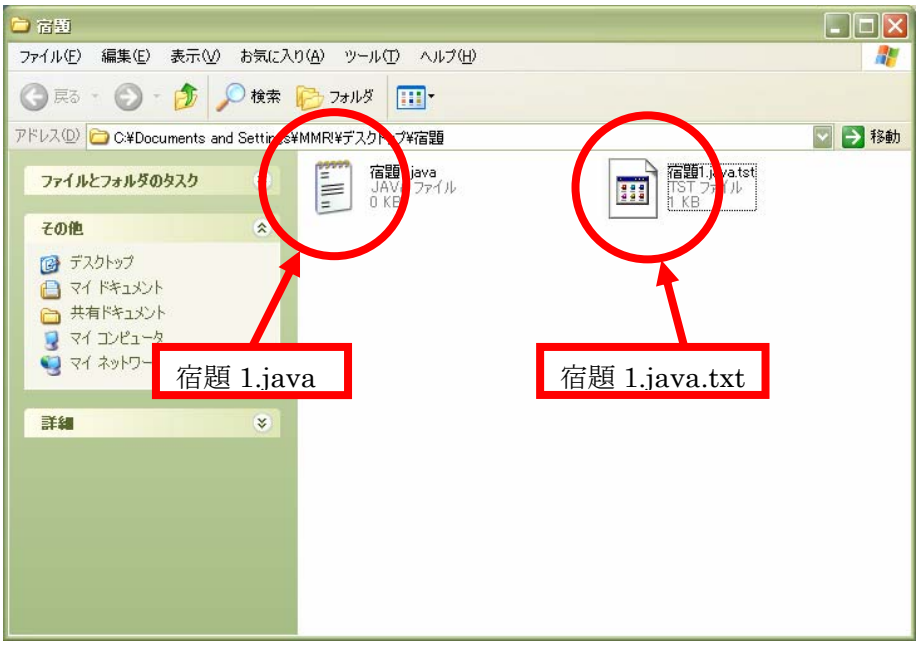


図 5：TST 生成

２．ドラッグ＆ドロップで生成する場合

(1)TimeStampClientTool.jar をダブルクリックで実行する(図 2)

(2)Time Stampを押したいファイル群(M_1～M_n)を、ドラッグ＆ドロップで、ツールの上に持っていく(図 6)

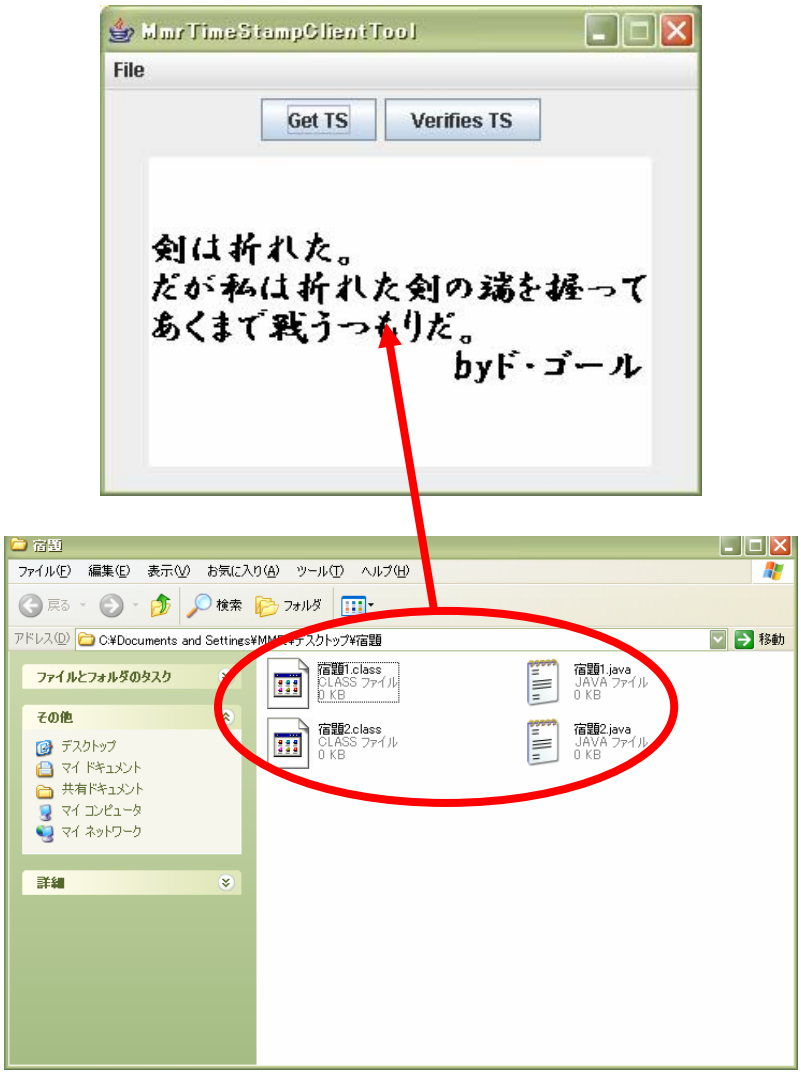


図 6：ドラッグ＆ドロップで Time Stamp 生成

(3)確認のダイアログ(図 7)が出て、TST が生成される

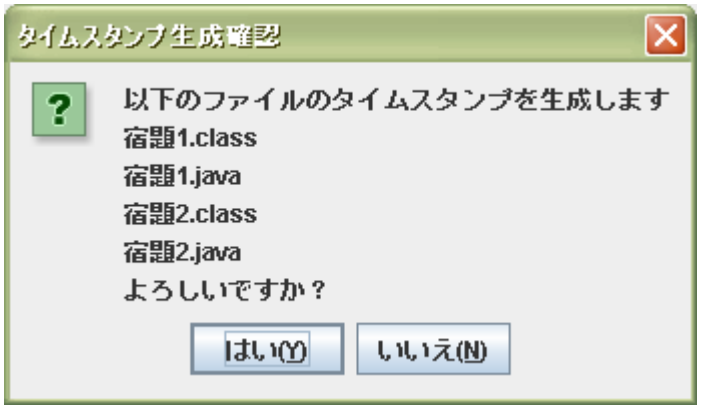


図 7：確認のダイアログ

3.4.2　Time Stamp の検証

Time Stamp を検証する方法も、2 種類ある

１．ボタンで検証する場合

(1) TimeStampClientTool.jar をダブルクリックで実行する(図 2)

(2)「Verifies TS」ボタンを押す(図 8)

図 8：「Verifies TS」ボタン

(3)検証したいファイル *M'* を選ぶ

(4)検証したいファイル *M'* の *TST* を選ぶ

(5)検証成功か失敗か、結果がダイアログで表示される(図 9・図 10)

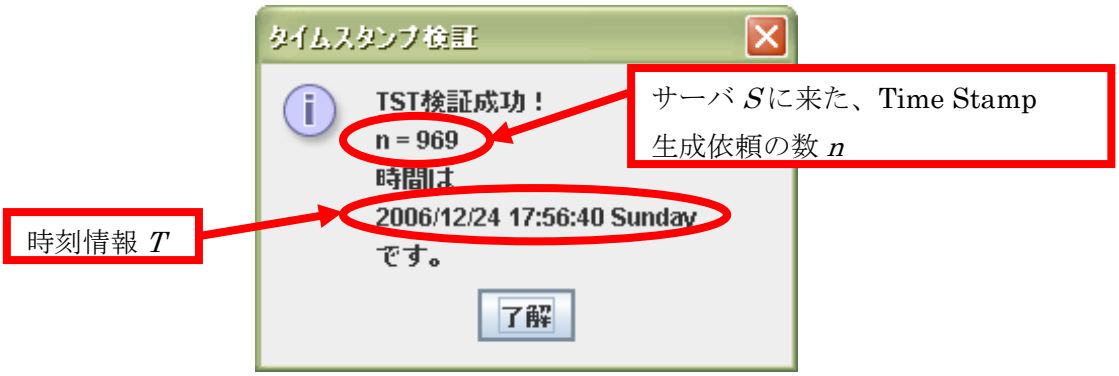


図 9：検証成功

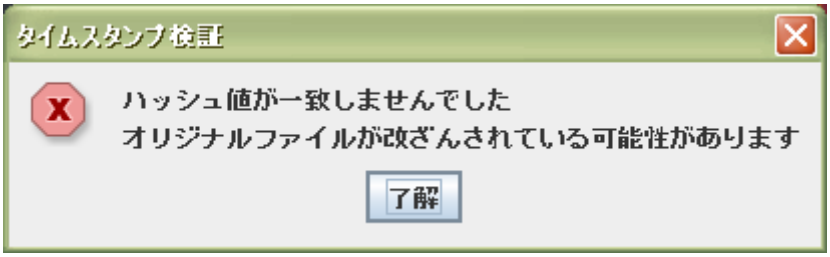


図 10：検証失敗

２．ドラッグ＆ドロップで検証する場合

(1) TimeStampClientTool.jar をダブルクリックで実行する(図 2)

(2) 検証したいファイル M’と、検証したいファイル M’の TST を、ドラッグ＆ドロップで、ツールの上に持っていく(図 11)

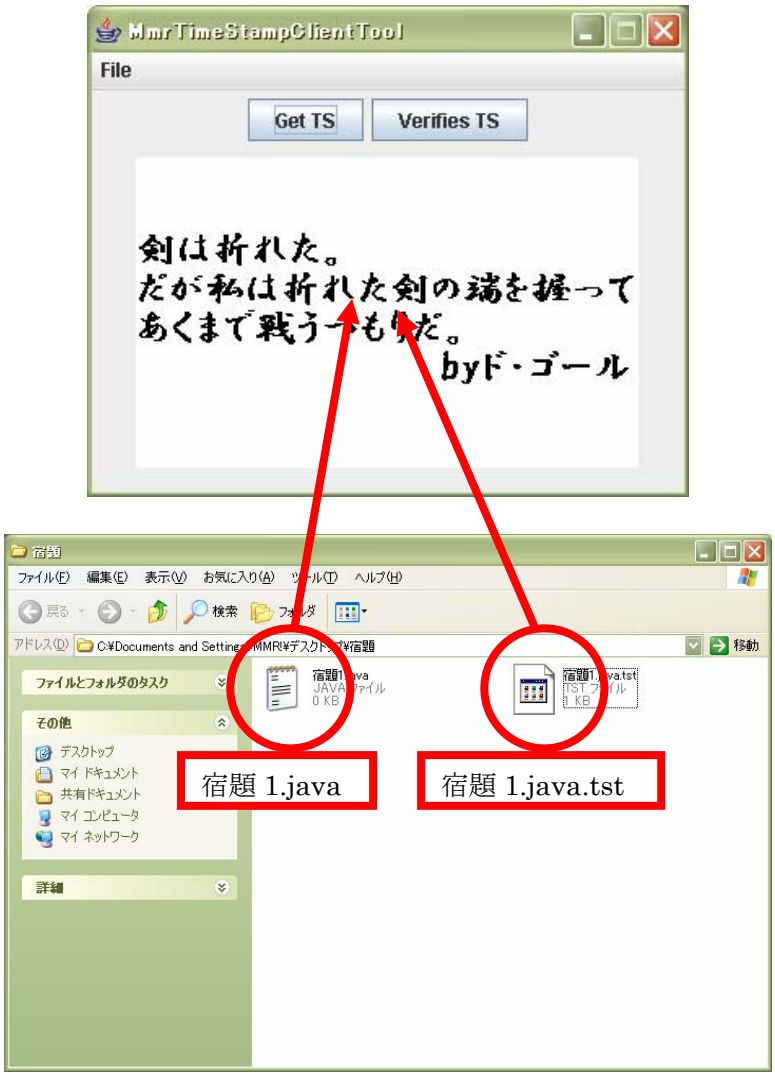


図 11：ドラッグ＆ドロップで Time Stamp 検証

(3)検証成功か失敗か、結果がダイアログで表示される(図 9・図 10)

3.4.3　公開情報の確認

情報公開サーバは、「PublicInfomationIndex.java」を置いたところにアクセスすると確認することが出来る。今回の場合は、

http://noisy.cs.dm.u-tokai.ac.jp/~nandatte/servlet/PublicInfomationIndex

で、確認することが出来る(図 12)

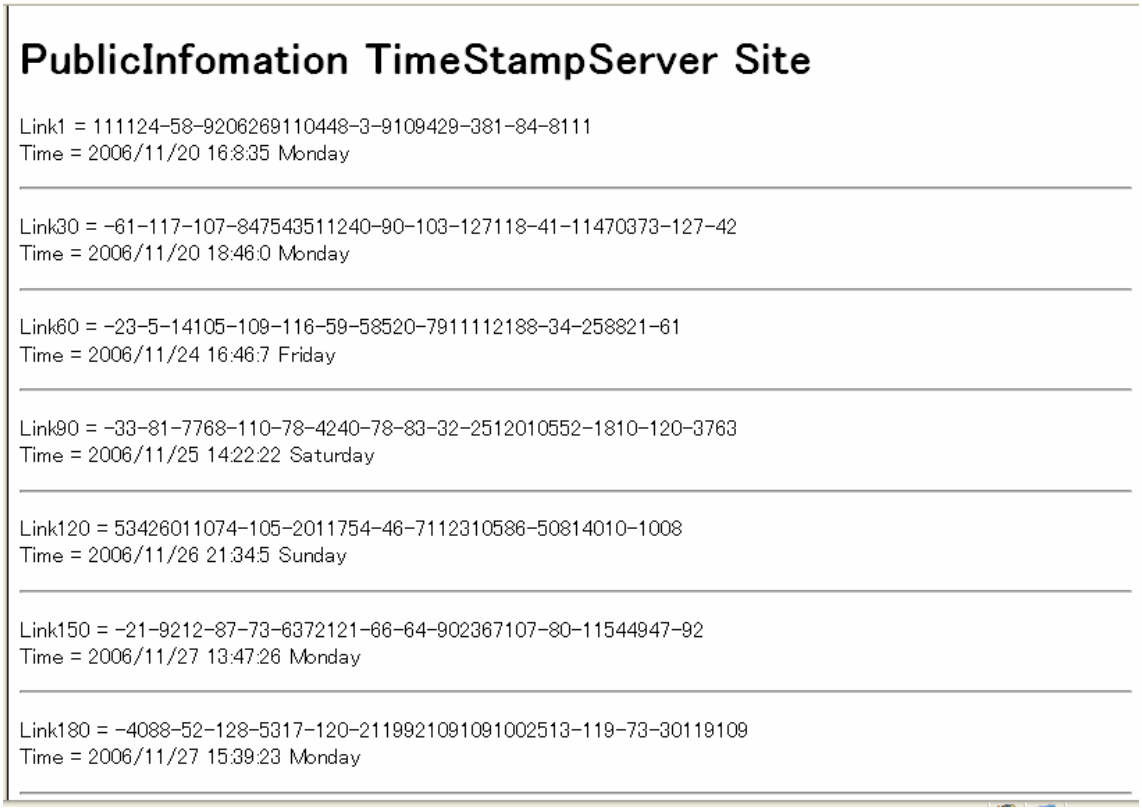


図 12：公開情報

3.5　システムの特徴

まず、Time Stamp生成・検証をドラッグ＆ドロップで行えるようにした。Time Stamp体験サイト[3]で使ったツールでは、Time Stampを押すためにファイルをいちいち選ぶのが面倒くさかった。なので、ドラッグ＆ドロップで簡単に行えるようにした。

そして、通信プロトコルは独自のものを使用している。

第４章　運用実験

4.1　実験目的

菊池研究室では、3 年生を対象に Java ゼミナールを行っている。毎週、宿題が出題されるが、期限内に出したと言い張ったり、期限後に秘密に修正したりするなどの不正行為の懸念があった。

それらを防ぐために、Time Stamp システム“S3”を使ってもらう。このシステムを使うことにより、上記した不正行為を防ぐことが出来る

4.2　実験方法

2006 年 11 月 21 日から、菊池研究室にて、3 年生 21 人に対して運用を行った。

http://www.cs.dm.u-tokai.ac.jp/Java/Java2006/TimeStamp/index.html に特設ページを設け、クライアントツールをダウンロードしてもらい、使ってもらう。

4.3　結果・評価

システム稼動中、定期的にアクセスログを取っていた。図 13 はそのまとめである。

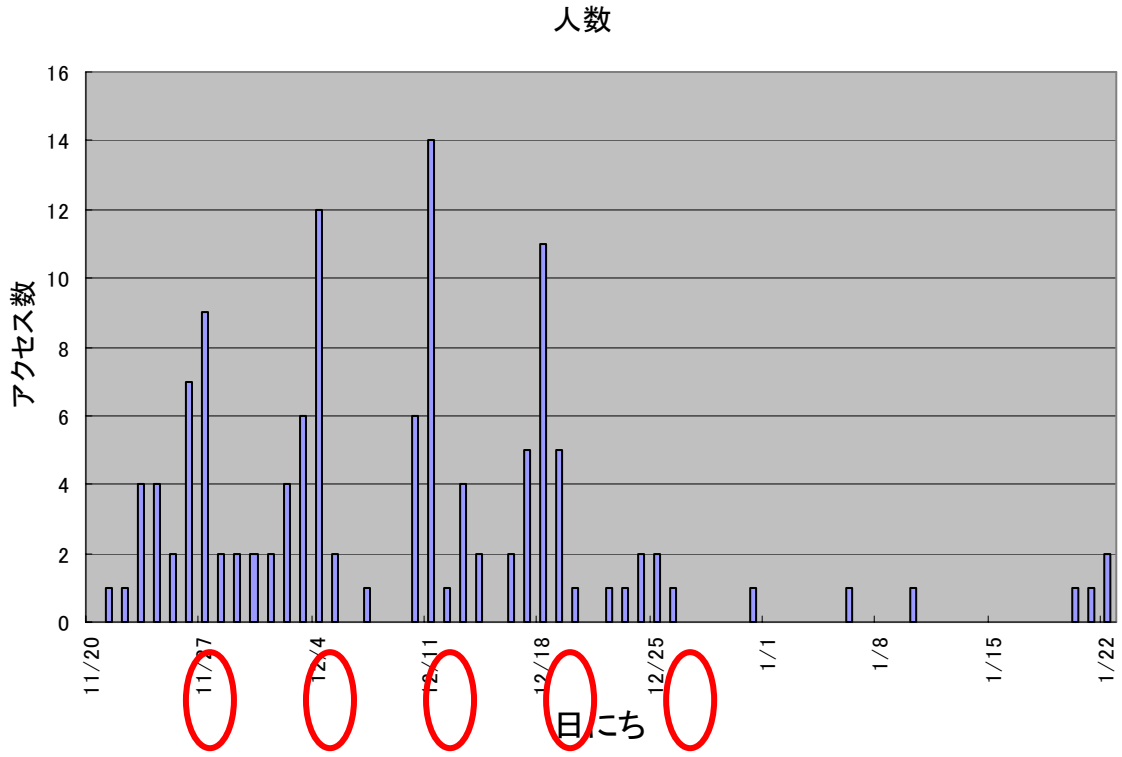


図 13：アクセス情報

赤い丸が付いている日が、宿題の提出日である。その提出日とその 1 日前にアクセスが集中していることが分かる。更に、12 月 25 日は宿題の提出期限だが授業が無かった日である。この宿題に対しては、前回の授業が終わった後やった人が 5 人もいた。

更にシステムを使用してもらった 3 年生からアンケートをとった。以下の図は、その結果である。

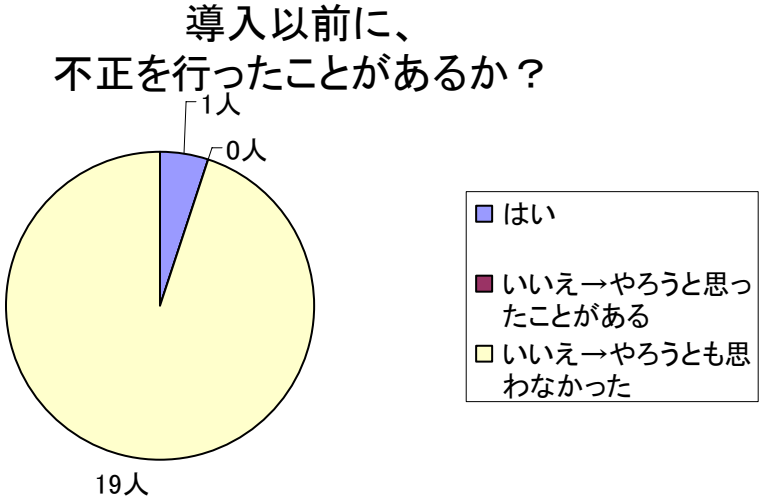


図 14：導入以前に不正を行ったことがあるか？

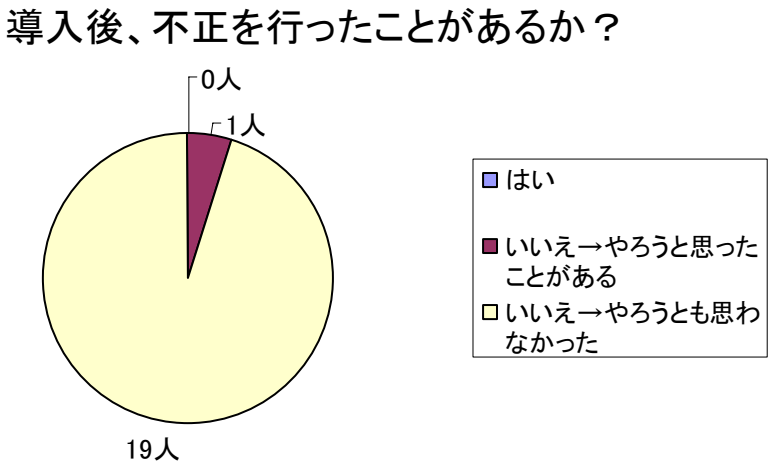


図 15：導入後、不正を行ったことがあるか？

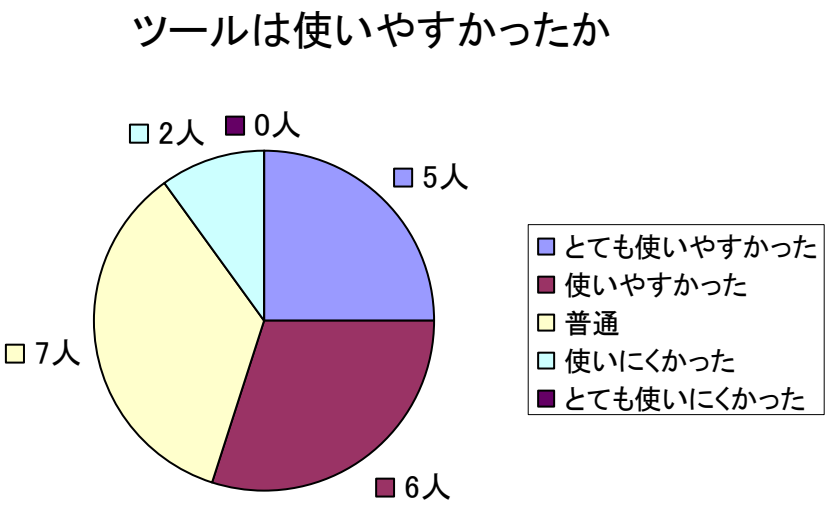


図 16：ツールは使いやすかったか

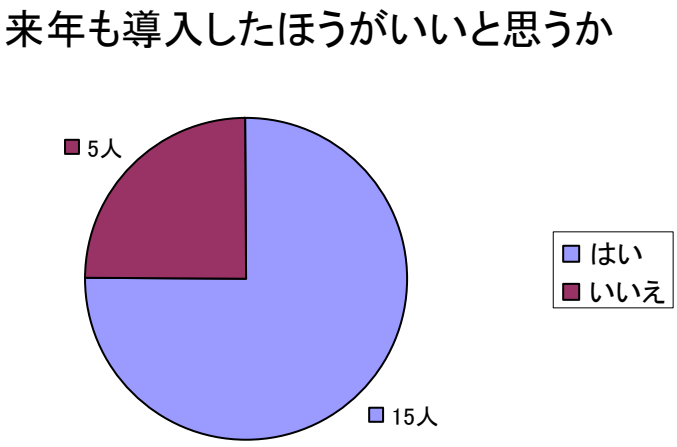


図 17：来年も導入したほうがいいと思うか

図 14 と 15 より、導入後、不正がなくなったことが分かる。図 16 では、半分以上の人が使いやすいと答えたので、ツールは使いやすかったということが分かる。逆に使いにくかったと答えた人の理由には、「サーバが落ちて使えなかったときがあった」というものがあった。運用中、2 回ほどサーバが落ちてしまった。これは今後の課題に挙げられるだろう。そして、図 17 より、このシステムの必要性はあったということが分かる。

第５章　おわりに

Java を用いて、提出時刻の改竄を防止する Time Stamp システム“S3”を開発した。
そして、宿題の提出時間の改ざんを防ぐことに成功した。
今後の課題として、検証の際にまとめて検証できるようにプログラムを変更することなどが挙げられる。

参考文献

[1] 宇根，他，デジタルタイムスタンプ技術の現状と課題，金融研究第 19 巻別冊第 1 号，pp. 105-154，2000

[2] RFC3161 http://www.ipa.go.jp/security/rfc/RFC3161JA.html

[3] SII，TimeStamp 体験サイト http://www.sii.co.jp/ni/tss/trial/index.html

謝辞

本研究を完遂するにあたり，多大なるご指導を受け賜りました東海大学電子情報学部情報メディア学科菊池浩明教授に心より感謝申し上げます.

そして，今回の実験に参加していただいた、菊池研究室の 3 年生，Java ゼミ係の皆さんに感謝の意を述べると共に，謝辞とさせて頂きます.

付録

１：RFID タグに TST を入れるプログラムのまとめ(夏休みの課題)

１．報告事項

(ア) 課題内容

① サーバにファイルのハッシュを送り、そのレスポンスとしてハッシュに時刻情報を付加(このファイルがタイムスタンプトークンとなる)して返す、簡易タイムスタンプシステムを作る(都合により、サーバはローカルで実現することとなった)。

② タイムスタンプトークンの情報を RFID に入れ、タイムスタンプトークンを視覚化する。

(イ) プログラム内容

① TimeStampClientToolMain.java

このプログラムは、クライアント側が使うツールのプログラムである。

1. GET TS ボタン

このボタンを押すと、まず、ファイルの選択を要求される。そして選択されたファイルのハッシュ値を計算し、サーバへ情報を渡す(今回はすべてローカルで実現)。すると、ハッシュに時刻情報が付加されて返ってくるので、その情報を RFID タグの中に挿入される。

2. Verifies TS ボタン

このボタンを押すと、まず、検証したいファイルの選択を要求される。そして、選択されたファイルのハッシュ値を計算する。タグをかざし、そのタグの中に書いてあるハッシュの値と、今計算したハッシュの値を比べ、等しければ検証成功、等しくなければ検証失敗のダイアログを表示する。

② TimeStampServerMain.java

TimeStampClientToolMain.java からハッシュ値を送り、そのレスポンスとして時刻情報を付加したものを返すプログラム。

都合により、発表では使用しなかった。

③ SerialPortHandler.java

RFID の基本設定が書いてあるプログラム。杉田くんから譲り受けたものなので、詳しくはよく分からない。

④ Sound.java

音を鳴らすプログラム。TimeStampClientToolMain.java にて、タイムスタ

ンプを生成したときと検証したときに音がなるようになっている。

(ウ) プログラムの実行手順

① TimeStampClientToolMain.java を実行する

② タイムスタンプを生成する

1. GET TS ボタンを押す

図 1：システム概観

2. ファイルを選択する

3. 以下のようなダイアログが出てきたあと、RFID タグをタグリーダーにかざす

図 2：ダイアログ

4. 以下のようなダイアログが出て終了

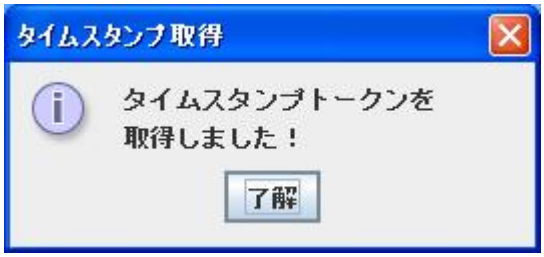


図 3：トークン取得ダイアログ

これで、RFID タグの中に、スタンプの情報が入る

③　タイムスタンプの検証

1. Verifies TS ボタンを押す

図 4：システム概観

2. 以下のようなダイアログが出て、ファイルを選ぶ

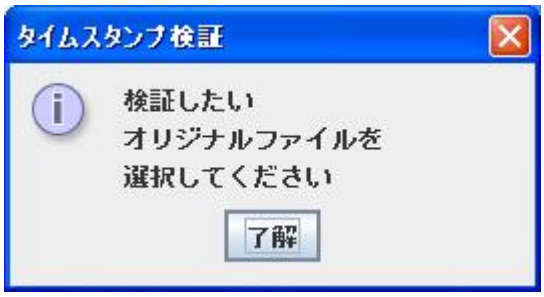


図 5：ダイアログ

3. 以下のようなダイアログが出たあと、RFID タグをリーダーにかざす

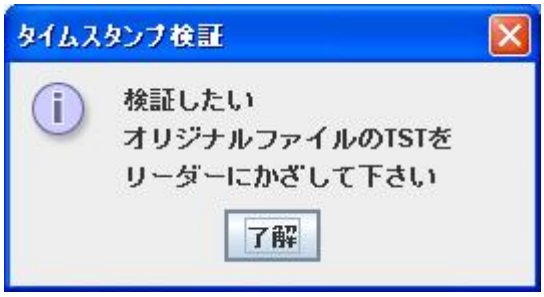


図 6：ダイアログ

4. 成功か失敗かのダイアログが表示される

成功の場合

図 7：検証成功

失敗の場合

図 8：検証失敗

④ RFID の中身

RFID の中身は以下のようになっている

ハッシュ値

-72-10-～7464,2006/10/17 Tuesday 1:0:36

時刻情報

2．結果

このプログラムにより、タイムスタンプトークンを RFID タグに入れることができ、視覚化することができた。これにより、タイムスタンプについて理解することが容易になると考えられる。

3．考察

(ア) RFID にタイムスタンプトークンの情報を入れることにより、タイムスタンプトークンを視覚化することに成功した。

(イ) そして、それによりタイムスタンプトークンについての理解が容易になったと考えられる。

(ウ) RFID タグにアグリゲイトでまとめたタイムスタンプトークンを入れて持ち歩き、タグをかざすだけで時刻を証明できるシステムは、なかなか便利そうだと思った。

２：RFC3161

RFC とは、インターネットに関する技術の標準を定める団体である IETF が正式に発行する文書である(IT 用語辞典より)。その 3161 番に、TimeStamp についての記述がある。それを調べてみた。

中には、TSA の要件や TST の細かい規定などが書いてあった。また、TST のフォーマットは、ASN.1 で記述してあった。

運用実験後、3 年生にアンケートを行った。

その結果を以下に示す。

３：アンケート結果

ここでは、TimeStmap 導入後に 3 年生を対象にしたアンケート結果の詳細を記す

１．タイムスタンプツールについて、あてはまるものに○をつけてください

① とても使いやすかった
② 使いやすかった
③ 普通
④ 使いにくかった
⑤ とても使いにくかった

①→5 人
②→6 人
③→7 人
④→2 人
⑤→0 人

２．上記の理由

１で①と答えた人

・ドラッグ＆ドロップ
・ドラッグ&ドロップで終わるから
・download だけするとすぐ使える、簡単ですから
・ドラッグ＆ドロップで実行できたのが使いやすかった
・ドラッグアンドドロップで操作することができたから

②と答えた人

・まとめてファイルを押すことが出来た
・ドラッグすればよかったため
・特に難しい操作もなく、難なく使えた
・ドラッグ＆ドロップでタイムスタンプファイルを生成できる点が直感的で使いやすかった
・D&D するだけの作業だったのでラクでした
・ファイルをドラッグするだけで良いから

③と答えた人

・毎回起動せずに D&D だけで使えればなおよかった
・普通に使えていたので
・ふつうだったから

・使いやすかったがネットワークが落ちたりと、不具合もあったりしたから
・ネットワークが落ちていたり、最初使ったときにフリーズしたりしたので
・特になし

④と答えた人
・いちいちタイムスタンプを押すためのファイルを探すのが面倒だった
・サーバが落ちたりしたので

3．タイムスタンプを導入する以前に、不正(提出期限後にコッソリ宿題の内容を変更したなど)を行ったことがありますか？
① はい
② いいえ→やろうと思ったことがある
③ いいえ→やろうとも思わなかった

①→1 人
②→0 人
③→19 人

4．上記で①・②と答えた人に質問です。何を行おうとしましたか？
・提出後に改善できる部分があったので変更した

5．タイムスタンプ導入後、不正(提出期限後にコッソリ宿題の内容を変更したなど)をおこなったことがありますか？
① はい
② いいえ→やろうと思ったことがある
③ いいえ→やろうとも思わなかった

①→0 人
②→1 人
③→19 人

6．来年以降、導入したほうがいいと思いますか？
① はい
② いいえ

①→15 人
②→5 人
③→0 人

７．意見、感想、要望等

６で①と答えた人

・サーバが落ちていたことがあったので、管理に気をつけて欲しい
・サーバがたびたび落ちて、使えなくなるのが残念でした
(仕方がないことだと思いますが)
・不正を行わせないという点ではタイムスタンプは非常に役立つと思うのですが若干手間がかかると感じました。もう少し使いやすい方がいいかと思いました
・これからも頑張ってください
・ここの良心にまかせても大丈夫だと思います
・使いやすかった
・採点がめんどくさいそうで…
・バイナリエディタか何かであれば不正が可能になりそうなのが少しだけ気がかりなので、何かしらの改良があった方がよさそうな気はする
・なし
・感想は、いつもタイムスタンプを押したか凄く気になっているのですし、
プログラムが正しく出来ていても、タイムスタンプを押さないことで点数を
失うのが凄くいたいと思いますけど。
以上です。
・宿題に関しては、このような感じで不正を防ぐ事が必要になると思うので、とても良いツールだと思います。
・検証するのが少し面倒くさかったです
・6 について、導入しなくても良いというのではなく、あまり必要性を感じなかった
・すばらしい技術だと思います
これを導入することによって学力向上・時間厳守ができなくなるので、これからもよりよいものになるように頑張ってください。
・提出するファイル数が多くなってしまって少し大変だった。

②と答えた人

・Java が使えない PC でスタンプが押せないのが少し不便でした
・ここの良心にまかせても大丈夫だと思います
・採点がめんどくさいそうで…
・6 について、導入しなくても良いというのではなく、あまり必要性を感じなかった