

Electronic Cash
CS3 Programming Methodology
Practical Exercise
Deliverable 1

by James O'Donnell
Michael Jane
Lucas Dixon
Jai Redden
Iain Reid

Contents

Requirements Documents	3
• Introduction	3
• System Model	4
• System Evolution	7
• Functional Requirements	9
• Attributes	11
• Constraints	13
• Hazards and Threats	14
Requirements Specifications	16
• Functional Requirements Specification	16
• Attribute & Measures Specification	17
• Constraints Specification	20
• Threats and Hazards Specification	21
• Glossary	23
• Reviews	24
• Appendices and EML specification	26

Introduction

(Edinburgh Leisure, our client, have commissioned this system and will be forthwith referred to as customer or client.)

Introduction to e-commerce system.

The rationale behind developing this system is that there is a need for a secure system to provide anonymous payment for services over the internet.

The system will be accessible by the user and the web based service provider.

The user will access the system to provide information concerning how much local currency the user will want to use via the system.

The web based service provider will access the system to verify a user's payment which in turn will provide a basis for the payment of the web based service provider by the e-commerce system.

The relevant parties in the normal running of the system are:

- the user
- the user's bank
- the company providing the web based service
- the owner of the e-commerce system. (our client)

Needs of the relevant parties

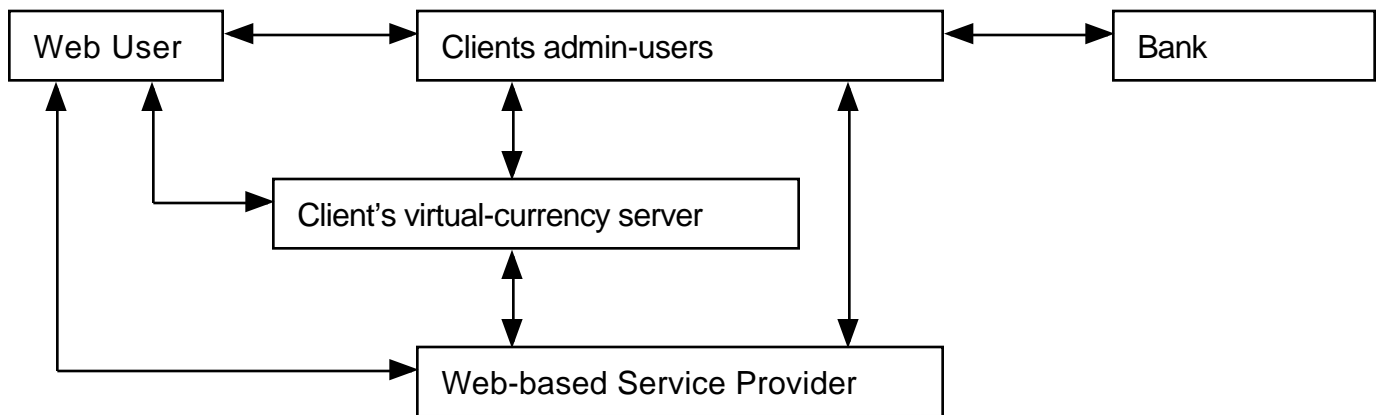
- **the user**
 - an anonymous payment system
 - assurance that their financial details will not be abused
- **the user's bank**
 - correct communication protocol for local currency transfer from user's account to customer system
- **the company providing the web based service**
 - a service to verify customers' payment
 - conversion of customers' virtual currency to local currency
 - to be part of an approved network of services meeting standards set by e-commerce system
- **the owner of the e-commerce system -**
 - facility to conduct financial transactions with users' credit/banking facility
 - facility to conduct verification of virtual currency via web based service providers
 - assurance that payment to web based service provider be made only as a result of a bona fide transfer from a user of virtual currency that has already been paid for.

System Model

High level system models

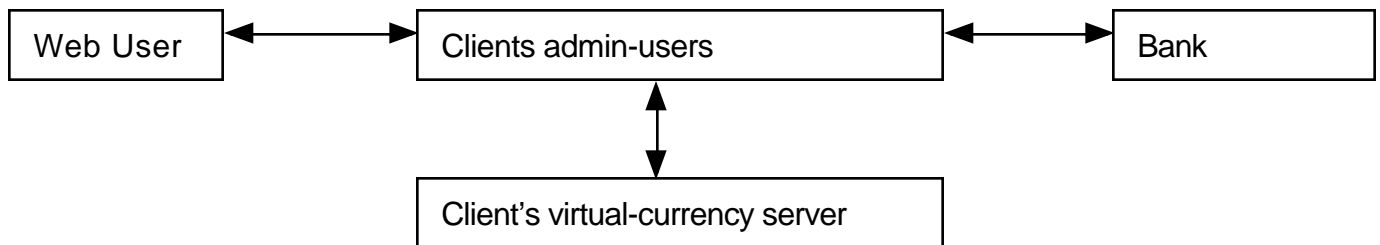
The following diagram represents the interaction of the parties involved in the system during normal use.

All groups involved



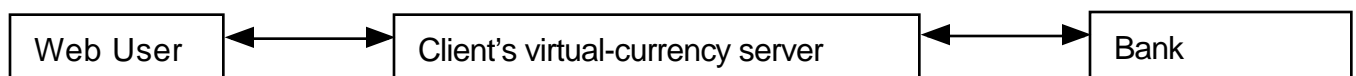
The following represents the interaction of groups involved in setting up an account with Direct Debit payment.

Direct Debit Account setup



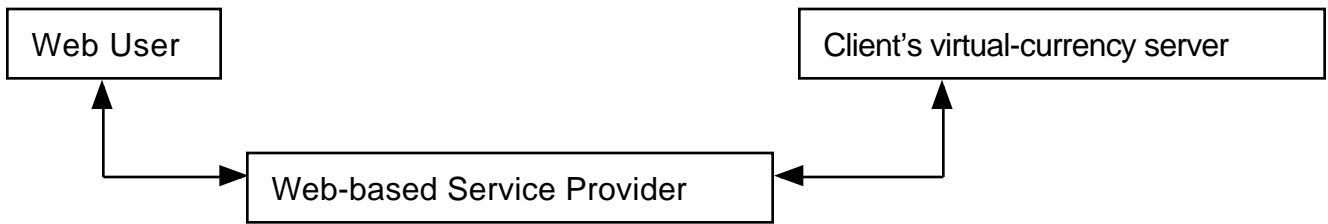
This diagram shows the interaction of groups involved in purchasing virtual currency with payment by credit card.

Credit Card Electronic Currency conversion



This diagram shows the interactions of groups involved in using the virtual currency to pay for a service on the internet.

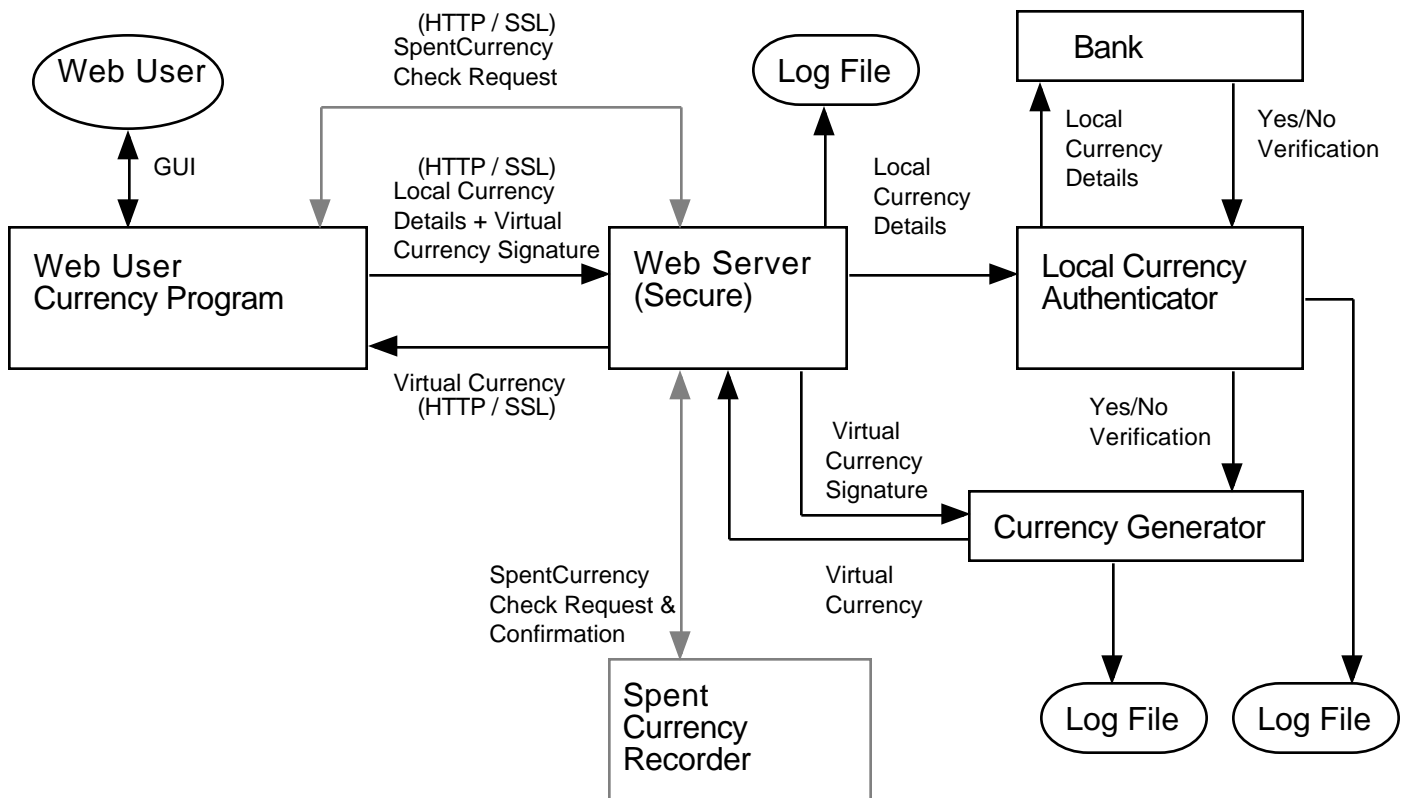
Web-User purchase from web-based service provider



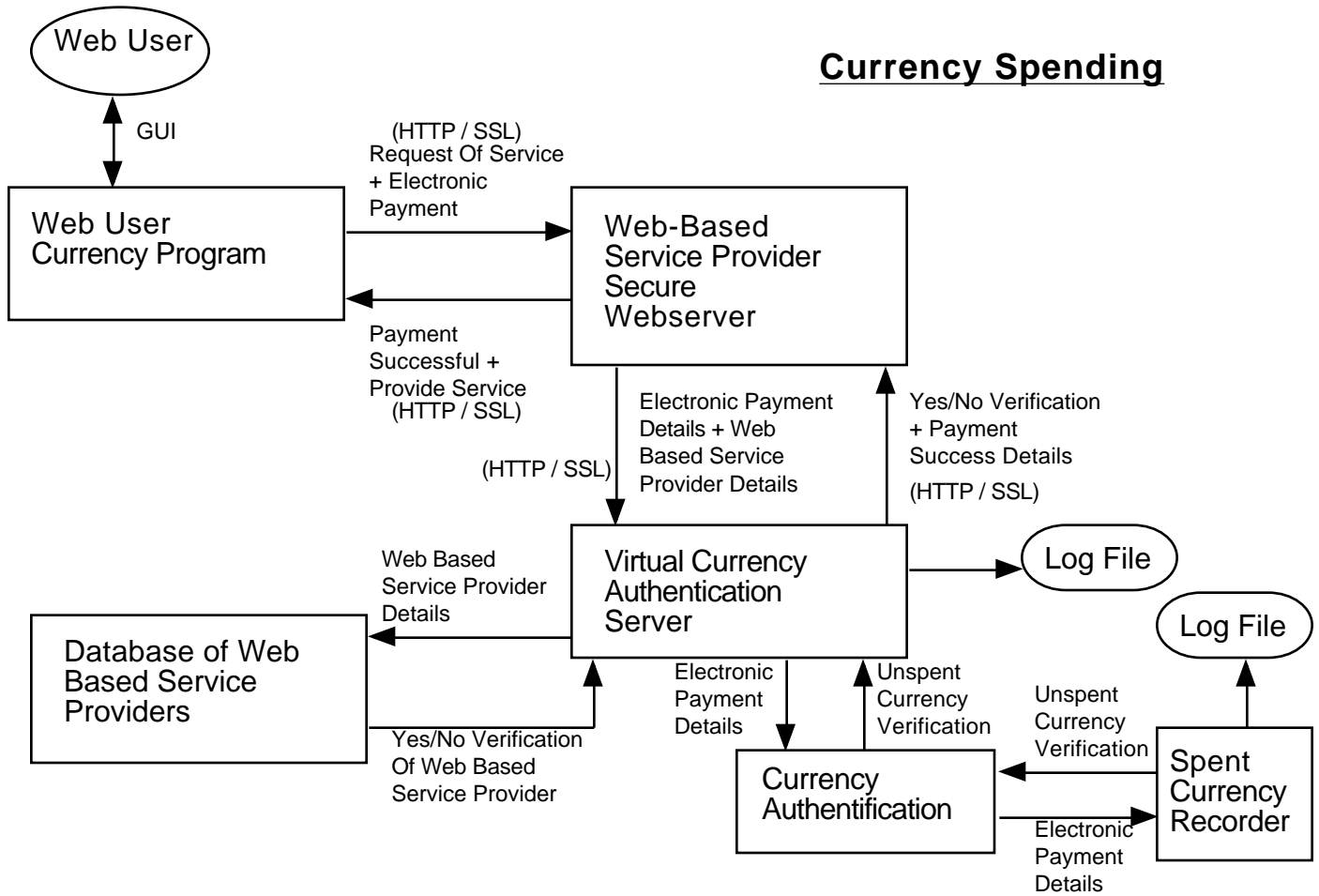
In depth system models

Here are two aspects of the system examined in more detail. The first is a model showing the interactions involved in a user obtaining virtual currency.

Currency Conversion



This second diagram shows the groups that must interact when a user is spending virtual currency on the internet.



System Evolution

Outline

It is prudent to assume that the environment in which the proposed system will operate, the constraints put upon its resources and the requirements of the system will change.

The following is an analysis of likely and possible changes in the environment which the system may have to adapt to. In addition, an exploration of the financial implications of such changes will be explored in terms of system requirements.

In particular the system should be designed so as to require the minimum developer involvement after product completion and acceptance.

Key assumptions liable to change

Demand

The client has provided specifications concerning quantity of users of the system and must have done financial analysis as to the profitability of the system. The system must however be able to adapt to an increasing demand.

The main implications for the system are:

- Bandwidth of connection between system and internet may need to be increased.
- Arrangements with credit companies and banks will need to accommodate the increased demand from the system for credit verification etc.
- Improved backup systems will be required if turnover much larger. These mirrors of the server should have separate physical internet connections reflecting the increased importance of avoiding any downtime of the system.

In addition, increased frequency of security reviews would be necessary as the service becomes a larger target for fraudulent attacks.

Legal changes to encryption and privacy laws

It may be that laws concerning the privacy of an individual will change. This may compromise the service the system is to provide. This may be solved by the relocation of the system.

If relocation isn't sufficient, the system could be adapted to make available information concerning transactions on the web to the relevant authorities without compromising the basic privacy of the user.

i.e. the users will continue to benefit from a service preventing their transactions from being detailed by credit and banking companies and appear as single payments to the customer.

Changes to encryption laws will result in a change in internet encryption protocol as described below.

Protocol Changes

The existing protocol for secure transactions with secure servers inability in the main web browsers is SSL, Secure Socket Layer.

If the working of SSL changes the system will be required to be updated. The extent of this task is dependent on implementation.

Should the system use the standard SSL capabilities of a popular browser, the browser will simply be able to be updated with a newer version of the browser.

Alternatively if the system uses an independent implementation of SSL, it will need to be modified. The user will simply be able to keep up with protocol changes by using latest versions of popular browsers.

Site relocation

The system may need to be relocated in two ways.

- The system and its sys-operators may be relocated geographically.
- The virtual location (IP-Address) of the system may change.

Should the system be relocated its virtual location may change and/or it may need to be run on different hardware. This is covered in platform changes.

Should the virtual location of the system change, certain changes will need to be made to the software. This would most likely be made possible by requiring the functionality of the sys-operator interface to include a simple utility to change the IP-Address.

Platform Changes (Portability) and hardware breakdown

The software may be required to be reinstalled due to hardware breakdown or change. The software will be designed to be used on any Unix system meeting certain minimum specification. The likelihood of the need for reinstallation and the need for the client's autonomy from the developer after product completion has been discussed. The system will come with automatic installation software that will not require experienced programmers to run.

Functional Requirements

General Customer Functional Requirements:

Currency Conversion:

Electronic Cash given to a web-user by the electronic cash server once payment details have been agreed. Amounts to conversion from local currency to Virtual Currency, where the local currency is passed from the web-users account to our clients (Edinburgh Leisure), and subsequent electronic passing of electronic currency to the web-user from our clients electronic currency server (digitally signed note).

Anonymous Currency Spending:

Electronic currency can be spent at a registered web based service provider that can handle our electronic currency (registered with our clients virtual currency server). This involves the web-user visiting a web based service provider's virtual shop through the web, sending details of what they wish to purchase, then sending electronic payment in the form of the our clients electronic currency. This is then registered by the shop and authenticity is confirmed with our virtual server so as to verify that it has been signed by our server and is authentic currency. Our clients virtual currency server should not be able to know the identity of the spender.

Currency Authentication:

Authentication involves a web based service provider registered with our clients virtual currency server, interacting with our clients virtual currency server. The registered web based service providers shop passed the electronic currency given to it by the web-user to the virtual currency server who verifies that the virtual currency (confirms the note was digitally signed by itself). The web based service provider must also prove to be registered with our virtual currency server.

Administration Functions:

Our clients require a number of administration tools to help with currency conversion, registration of web-based service providers, and observation and monitoring of the actions of the virtual currency server. Registration of web-based service providers, and subsequent handling of information to do with registering a web based service provider should be usable by the administration staff of our clients, and is required in the form of a simple flat file database which can be searched. This database already exists, our electronic currency server must interact with this database as appropriate. Observation and monitoring of the actions of the virtual currency server should be usable by development and technical staff of our clients.

Security Functions:

Security sending and receiving of data is required in HTTP protocol using SSL for security of all web transactions. A secure method of identity verification of a web based service provider is required when the web based service provider passes on payment from a web-user.

Inverse Requirements:

Electronic currency, local currency, and web based service provider identification details must never be sent through HTTP without encryption (without SSL). Spent Currency Recorder must never contain two identical spent currency records. (Same note is never recorded to have been spent twice). A web-user must not be given virtual currency without having provided a local currency payment method to our clients. (assumed to be credit cards)

Functional Requirements as Software Bodies:

The software involved in an anonymous electronic currency bank required by our clients falls into three user categories:

- Web users - require a program(s) to interact with the virtual currency server for currency conversion, and to interact with a web-based service provider. This program is outside our requirements, however it does require acknowledgment of its existence and the protocols it uses (HTTP over SSL)

- Web Based Service Provider - require a program(s) to interact with the virtual currency server for currency authentication, and to interact with a web-users when they purchase a service. This program is outside our requirements, however it does require acknowledgment of its existence and the protocols it uses (HTTP over SSL) the same protocols as the web-user.

- Client's Virtual Currency Server - This is the program(s) that we are writing the requirements for, it must interact with Web-Users for currency conversion (through HTTP over SSL) and web based service providers for currency authentication (also through HTTP over SSL), and client administration staff for registration of web-based service providers, and with client technical staff for observation and analysis of behavior.

Attributes and Measures

a list of attributes with their expected level of attainability follows. metrics corresponding to qualitative attributes are listed where possible.

security:

expected level: the three transfer lines of data previously identified in the system model (system-bank, system-company and system-customer) must use an encryption system which cannot be decoded by any financially feasible means. In other words the security of the system-company and the system-customer link need only represent the fact that a relatively small (< £100) amount of cash is at risk whereas as the system-bank transfer represents a much greater risk as credit-card details and/or large sums of money will be at risk. it is expected that the as-is encryption provided with major browsers such as Netscape Navigator/Communicator and Internet Explorer will suffice.

the internal security, e.g. the databases resistance to hacking attempts (both internal and external), security of internal valuable data (e.g. credit-card numbers, server-ID's etc... should be maximal. Sysops should have no way of accessing internal valuable data. Neither should they be able to create virtual currency or add a piece of virtual currency to the spent-currency-list without relevant authority.

maintainability

expected level: the system is expected to be very easy to maintain, both for sysops (when it is working) and for programmers (when it isn't).

ease-of-use

expected level: as the client wants to attract as many customers and companies as possible, and many of these may be not highly computer literate, they have specified that it is crucial that all of the above, except the sysop requirement, are very easy to do. obviously some stages may be difficult if the customer does not meet the requirements (good credit rating etc...) but, for a valid customer the system should be very easy to get set up with and use. sysops should be expected to be proficient with the latest browser technology and have a good working knowledge of databases, other than that no extraneous knowledge should be expected from the sysops.

performance (response time and load capability):

expected level: (NB: these figures were suggested last Sat for the constraints section but upon looking at the notes further, I reckon they belong here).

function	response time	load capability
cash request	5 seconds	30 users s-1
cash verification	5 seconds	180 users s-1

portability:

expected level: our client has announced that their long-term business plan only involves the use of Unix so platform portability is not required. it is expected also that all users (customers and sysops) of the system will have access to web browsers supporting sufficient security, the system should however be portable across all major browsers.

size:

expected level: the data storage capacity will increase as a record must always be kept of spent virtual currency. The log files will need 1Kb per transaction.

modularity:

expected level: those functions a described in the system model should be modular

Constraints

Completeness

The software we provide with initial installation support and hardware provided by client, will be sufficient as to constitute a system capable of meeting the requirements of the relevant groups as discussed in the introduction and other sections of this document.

Operating system of customer's server

The system will run on a Linux operating system.

Internet connection of client

The client's hardware includes a permanent land line connection to the internet. The client's current hardware is acting as a secure server.

Constraints regarding user's system

The system must be able to interact with the user without the need for the user to install specialist software. The implementation should take advantage of the security functionality of common web browsers and of browser plugins.

Transactions of virtual currency

These must be made as described above using standard web browsers and plugins.

Accessibility of system

The system is not to be accessible remotely other than via requests made to the system as described in the functional requirements and specification documents. In addition there must be provision in the system operator interface for differing levels of accessibility to reflect differing authority of sysop.

Security and anonymity

Security and anonymity are obviously two very important aspects of this system. It is very important that the users are confident that it is impossible for information regarding them stored on the system be passed on to anyone.

Security is to be handled at the users end, by means of browser encryption software. When browsers upgrade their security, our system must be able to cope without major upgrades being required.

Speed of system

Our system is to operate over the internet. As such response time of system to requests from other relevant groups must be minimised to provide a satisfactory service for those parties.

Should there be a delay, there must be facility to communicate this to the relevant party.

Threats and Hazards

Threats to the system will be mainly in regard to the transfer of the virtual currency, the requests for money from clients or the demands for payment from the sites. The possibility of breaking into the main sites to access the database of clients is also a threat on the system.

Threat to Virtual Currency Transfer

This threat is combated by its singular nature. This allows the system to check that the virtual currency has not been used before thus resulting in none validation.

The singular nature of the money also stops criminals from just finding the generation system and start 'printing' money.

Threat of System tampering

The threat of people breaking into the system either at the main centre or at the individual site systems is a major problem.

The limited access to the main system removes a lot of the risk to this.

Only allowing forms and small verification packets near the main system means it is possible to exclude the unwanted traffic to the main system.

The web based service provider programs require more protecting to stop it being possible to make it continually generate positive verifications.

It also has to be able to distinguish between an authorised verification and a false verification generated by the new user. The web based service provider system will only allow verification if the verification code matches some pre-generated type.

Fraudulent users

Clients requesting money who either do not have the required funds or provide incorrect bank account details will be another threat.

The first is not a problem for our system, the account will be charged and the subsequent problems arising from this will be between the user and their bank.

The submission of incorrect bank details is a serious problem. This is combated by the checking of the users bank details before they receive virtual currency.

Fraudulent sites

The singular nature of the virtual currency means that it will be very hard for the sites to charge twice, but due to there closer links to the system the threat is greater.

The main hazards the system will face is when either the main system is off line for some reason or customer sites crash.

Loss of data due to system failure

The main system has to backup frequently so there is a hard copy of the database. This combined with the log of the day's transactions should allow the system to return to full operation in the shortest time possible.

This backup will be kept on a separate computer, ideally in a separate geographical location.

In addition, a more accessible backup facility should record transactions as they are made. These precautions should make it possible to restart the system from a crash quickly, and rebuild the database.

System failure during data transmission and other transmission errors

If the failure of either the main system or web based service provider system interrupts a data transfer there is a possibility of a hazard. This is handled by not conducting any transaction until a full request has been sent.

Corruption of data transfer should not be too big a problem due to the small size of the information being transmitted.

The system should be able to recognise and compensate for low levels of corruption and should just ignore more corrupted messages and request the message be resent. This facility is provided by the standard File Transfer Protocol.

Functional requirements specification

Existing system which both lets a user store, and use electronic cash on a personal computer. (interacting seamlessly with web-browser).

Outline Of actions performed by Web-User Software:

- Send web-user request of local-currency to electronic-currency conversion, providing credit card details.
- Receive and store electronic currency.
- Encrypt virtual currency on local computer with password.
- Receive a request for payment from the local user, providing user verification.
- Sent currency to remote system.
- Receive and store successful payment slip.
- Web-User is able to view owed currency.
- Web-User is able to view payment successful slips.
- Web-User is able to delete payment successful slips.

Currency conversion from web-user's local-currency to client virtual-currency, via credit card. Currency taken from web-user in local currency is transferred to our clients.

- Receive request from web-user to transfer currency provided with credit card details.
- Receive request from web-user to transfer currency provided with verification of web-user and web-users existing debit account.
- Verify credit card.
- Verify web-user and existing debit account.
- Send User electronic currency.
- Send bank details of transaction from user's account to clients account.

Virtual-currency validation system to confirm that virtual-currency is valid.

- Validate a web-based service provider's identity verification.
- Receive virtual-currency from web-based service provider for validation.
- Validate the virtual-currency. (check if it's valid)
- Register that virtual currency has been spent.
- Send payment successful slip to web-based service provider.

Web-Based Service provider electronic currency functions

- Send Web-Based Service provider identity verification.
- Receive Electronic cash from user (non tamperable/usable by Web-Based Service provider).
- Request validation of currency with clients virtual-currency server.
- Receive validation of currency and payment successful slips.
- Send payment successful slips to web-user.

Administration for web-based service providers. Used by client administration staff

- View web-based service providers account.
- Delete web-based service providers account.
- Edit web-based service providers account.

Encryption Functions:

- Provide SSL encryption for all remote/web based transactions.
- Provide appropriate encryption for digital signatures for both the web-user, our clients server, and digital currency.
- Communication via HTTP.

Other notes on functions:

* Send User electronic currency / Receive and store electronic currency Will probably involves a process of handshaking (communication between both web-user and client's server)

Attributes and Measures Specification

a list of attributes with their expected level of attainability follows. metrics corresponding to qualitative attributes are listed where possible.

security:

metric: international standards governing the use of encryption and security exist. by law these must be met.

expected level: the three transfer lines of data previously identified in the system model (system-bank, system-company and system-customer) must use an encryption system which cannot be decoded by any financially feasible means. In other words the security of the system-company and the system-customer link need only represent the fact that a relatively small (< £100) amount of cash is at risk whereas as the system-bank transfer represents a much greater risk as credit-card details and/or large sums of money will be at risk. it is expected that the as-is encryption provided with major browsers such as Netscape Navigator/Communicator and Internet Explorer will suffice.

measure: the internal security, e.g. the databases resistance to hacking attempts (both internal and external), security of internal valuable data (e.g. credit-card numbers, server-ID's etc... should be maximal. Sysops should have no way of accessing internal valuable data. Neither should they be able to create virtual currency or add a piece of virtual currency to the spent-currency-list without relevant authority.

any encryption algorithms in the system must be measured by the international standards as per the law and must meet with these standards. ITU-T (CCITT) describes a standard for telecommunications systems, in particular certificates and it is suggested that this is conformed to. Other international standards include ISO, ANSI, PKCS and IEEE

maintainability

metric: one answer would be the average (or some other statistical value e.g. median) time taken to update system (fix a defect, produce a patch for the latest IE or whatever) however this is impossible to measure accurately and more importantly may not give meaningful results. Our client has stated that they are not concerned with a maintainability metric, rather they want high maintainability to be achieved through modularisation

expected level: the system is expected to be very easy to maintain, both for sysops (when it is working) and for programmers (when it isn't).

measure: the expected level of modularity should be reached. in addition a high level of documentation both functionally internal (describing how each function works) and functionally external (function inputs/outputs, overall program flow etc...) should be provided

ease-of-use

metric: many sub-attributes exist here

- how easy is it to learn to use the system (and get set up an account with client in order to do this)
- how easy the system is to use it once learnt
- how easy it is for web based service providers to integrate with the system
- how easy it is for web based service providers to operate the alternative payment method
- how easy it is for the sysops to operate the system

all of the above are difficult to measure precisely, and no true metric exists for this. they will have to be measured in loose, fuzzy terms such as 'easy', 'difficult', etc.

expected level: as the client wants to attract as many customers and companies as possible, and

many of these may be not highly computer literate, they have specified that it is crucial that all of the above, except the sysop requirement, are very easy to do. obviously some stages may be difficult if the customer does not meet the requirements (good credit rating etc...) but, for a valid customer the system should be very easy to get set up with and use. sysops should be expected to be proficient with the latest browser technology and have a good working knowledge of databases, other than that no extraneous knowledge should be expected from the sysops.

measure: our client demands the use of Beta testers, both to check the quality of the solution and to check that the system conforms with expected level of ease of use. the system must match up with the Beta testers analysis of 'very easy' and it would be suggested that in-house testing be performed prior to the Beta testing to limit the amount of Beta testing needed and keep to the development constraints specified in the constraints section.

performance (response time and load capability):

metric: response time: how long the system takes (ms) to respond to a request for more virtual currency and how long the system takes (ms) to confirm the attempt to spend cash through a web based service provider site.
load capability: how many requests the system can take (per second) for both of the above functions.

expected level:

function	response time	load capability
cash request	5 seconds	30 users s-1
cash verification	5 seconds	180 users s-1

measure: extensive testing on various speeds of networks should be used to ensure that the expected level is met for both of these critical user level functions

portability:

expected level: our client has announced that their long-term business plan only involves the use of Unix so platform portability is not required. It is expected also that all users (customers and sysops) of the system will have access to web browsers supporting sufficient security, the system should however be portable across all major browsers.

measure: the system should be rigorously tested on all major browsers. Assuming the portability of these browsers has been tested across all platforms the customers might use, portability across the browsers is then sufficient.

size:

metric: data storage for database and log files

expected level: the data storage capacity will increase as a record must always be kept of spent virtual currency. The log files will need 1Kb per transaction.

measure: both values are easily measurable using common OS utilities

modularity:

expected level: those functions a described in the system model should be modular
measure: modularity can be tested by the ability to replace a module inside the system with an equivalent (with the same interfaces) module.

Constraints Specification

Completeness

The software we provide with initial installation support and hardware provided by client, will be sufficient as to constitute a system capable of meeting the requirements of the relevant groups as discussed in the introduction and other sections of this document.

- Operable by system administration staff with no programming knowledge

Operating system of client's server

The system will run on a Linux operating system.

- The server side software must be compatible with Apache web server.

Internet connection of client

The client's hardware includes a permanent land line connection to the internet. The client's current hardware is acting as a secure server.

- Land line is a T3 permanent connection.
- Apache will handle all web transactions.

Constraints regarding user's system

The system must be able to interact with the user without the need for the user to install specialist software. The implementation should take advantage of the security functionality of common web browsers and of browser plugins.

- Usable with Netscape and Microsoft Internet Explorer versions 4 and above.
- The plugin must be run under Mac OS7 and above, Windows 95 and above and Linux release 5 and above.

Transactions of virtual currency

These must be made as described above using standard web browsers and plugins.

- Transactions must be through HTTP standard.
- Transactions must be encrypted with SSL

Accessibility of system

The system is not to be accessible remotely other than via requests made to the system as described in the functional requirements and specification documents.. In addition there must be provision in the system operator interface for differing levels of accessibility to reflect differing authority of sysop.

Security and anonymity

Security and anonymity are obviously two very important aspects of this system. It is very important that the users are confident that it is impossible for information regarding them stored on the system be passed on to anyone.

Security is to be handled at the users end, by means of browser encryption software. When browsers upgrade their security, our system must be able to cope without major upgrades being required.

- Encryption key size of encryption must be 128 bit.
- All remote transactions must be encrypted.
- It must be trivial to update encryption standard.

Speed of system

Our system is to operate over the internet. As such response time of system to requests from other relevant groups must be minimised to provide a satisfactory service for those parties.

Should there be a delay, there must be facility to communicate this to the relevant party.

- The system should be able to handle 30 currency conversions and 180 virtual currency authentications per minute.

Threats and Hazards Specification

Threat: Present virtual currency is attempted to be spent again

Reported: Currency Verification Function

Likelihood: Highly

Consequence if not reported: Free access to the sites and loss of revenue

Response to attempt: Report and log failure, notifying Web Based Service Provider.

Threat: Cracking the encryption on the virtual currency

Reported: Media, and change in Log statistics.

Likelihood: Very Unlikely (due to the level of RSA used)

Consequence if not reported: Ability to create currency by another body.

Response: Limit the lifespan of the money to a six months

Threat: Breaking the web based service provider program by hackers

Reported: Site Management, backed up log files.

Likelihood: Unlikely

Consequence if not reported: Possibly crashing/ stopping the authentication server, and

Response: There will be a simple checking program that can check the very simple code of the web based service provider program to spot any irregularities.

Threat: Generation of apparently virtual currency though familiarity with the system

Reported: Verification Function

Likelihood: Unlikely

Consequence if not reported: Free access to all the supported sites

Response: Report and log the transaction, then notify the Web Based Service Provider of the incorrect virtual currency.

Threats: Irrelevant forms

Reported By: Administrating Function

Likelihood: Very High

Consequence if not reported: Excessively large database of clients and pointless checking of details

Response: There is a basic check of the forms submitted automatically

Threats: requesting invalid currency conversion

Reported: CurrencyVerification Function at Bank

Likelihood: Unlikely due to the checking customers

Consequence if not reported: bank fraud, loass of local currency by bank

Response: None payment of the request and possible removal from the system.

Hazard: Main system crashes (this includes power failures etc)

Likelihood: Unlikely

Consequences: possible loss of transactions and forms received since last backup.

Response: Due to the daily backup of the system and the continual log of the events and actions of the system it should be possible to recreate the database at the time of the crash. The web based service provider sites will not allow access to until the system is back online and will inform people that other methods of payment are temporarily required.

Hazard: To many transactions overloading the system

Likelihood: Dependent on client's growth and exact hardware specification, currently unlikely.

Consequence if not reported: Entire system shutdown in worst case

Response: Limiting the number of transactions that are allowed by the system and give a polite request for the user to attempt again.

Hazard: Virtual currency is interrupted during mid transfer between web based service provider and virtual currency authenticator.

Likelihood: Moderate

Reported by User: Web transaction layer/WebServer

Consequence if not reported: Failure of Authentication, useless log entry

Response: Test transaction and try again, if web based service provider fails to respond, log and don't provide authentication report.

Hazard: virtual currency is interrupted during mid transfer between webuser and virtual currency authenticator.

Likelihood: Moderate

Reported by User: web transaction layer/WebServer

Consequence: User doesn't get his requested virtual currency, and confirmation of sent data fails

Response: Transmit another copy of the virtual currency if the user is still connected, else log transaction failure and don't charge user.

Glossary of terms

- **User** - user of virtual currency
- **Customer or Client** - the supplier or converter of virtual currency
- **Web based service provider** - A virtual shop where Web-Users spends virtual currency
- **Virtual currency** - our form of currency
- **Virtual-Currency-Server** — Our clients server, converts and authenticates currency.
- **HTTP** — the world wide web protocol. (Hyper Text Transfer Protocol)
- **SSL** — Secure Socket Layer - protocol used for encrypted transaction.
- **Verification** - the process of getting a reply from a site and saying whether they can enter or not.
- **System or site operator** - staff working for the system owner maintaining the database
- **Local currency** - currency in which the user's bank deals in
- **e-commerce - (electronic commerce)** - financial transactions made via the internet, the payment transaction being made entirely in the electronic domain
- **System operator or sysop** - Employee of the customer who maintains the database
- **Browser plugins** - Software written to be run by web browser

- **RSA cryptosystem and general encryption documents** - see Electronic Intercourse of a Commercial Nature [HTTP://www.elmorian.zetnet.co.uk/](http://www.elmorian.zetnet.co.uk/)

Index

• Attributes	11
• Attribute Requirements	17
• Constraints	13
• Constraints Specification	20
• EML Specification	26
• Functional Requirements	9
• Functional Requirements Specification	16
• Glossary	23
• Hazards and Threats	14
• Hazards and Threats Specification	21
• Introduction	3
• Requirements Documents	3
• Requirements Specifications	15
• Reviews	24
• System Evolution	7
• System Model	4

Review 15/11/99

The document review on the 15th November 1999:

The client requested a more detailed specification to confirm that an unregistered web based service provider could not act as a registered web based service provider, and that our client would be able to impose limitations on the use of electronic currency provided to them by the web user. In particular the client proposed a scenario where a pirate web based service provider posing as a registered web based service provider, could receive electronic currency, and use the electronic currency itself, without providing a service.

To further explain how the specification of our system would make the above threat impossible, the transaction between the virtual currency authentication server and the web based service provider is explained in further detail.

The steps in registration of a web based service provider include:

1. Registration with the database of web based service providers, this includes the IP address and location of their submission form.
2. The webpage and form program linking to the virtual currency authentication server from the web based service provider is then digitally signed by the virtual currency server.

The steps for payment and virtual currency authentication are:

3. When a Web-User connects to a site they can confirm the site to be a registered web based service provider by checking that the digital signature matches the page.
4. When a web based service provider sends their private details, the Currency Authentication Server ensures that it is a registered web based service provider.

Thus the additional information in threats and hazards would be.

Threat: Unregistered web based service provider attempt to appear to be a registered web based service provider. This could be done in two ways:

(1) Either by partially copying a registered web based service provider page, but not passing currency directly to the Virtual Currency Authenticator. In This case the web users Currency Program will let the web-user know that this site doesn't have a correct digital signature from the currency authentication server (checked by using currency authentication server public signature key).

(2) The second possible method of posing as a registered web based service provider is by copying a registered web based service provider's webpage exactly. This threat is handled as follows:

- firstly, the IP address and location of the registered web based service provider's webpage is part of the digital signature signed by the currency authentication server. Thus the web users Currency Program will let the web-user know that this site doesn't have a correct digital signature from the currency authentication server.
- Given the registration steps two and three above, the second possibility is to ignore this. The page would have to contain a direct link to the registered web based service provider's script for payment that would in turn return the user to the registered web based service provider's website, so the pirate page would act simply as a gateway to the registered web based service provider's site, and never let the pirate site have access to the web-users currency.

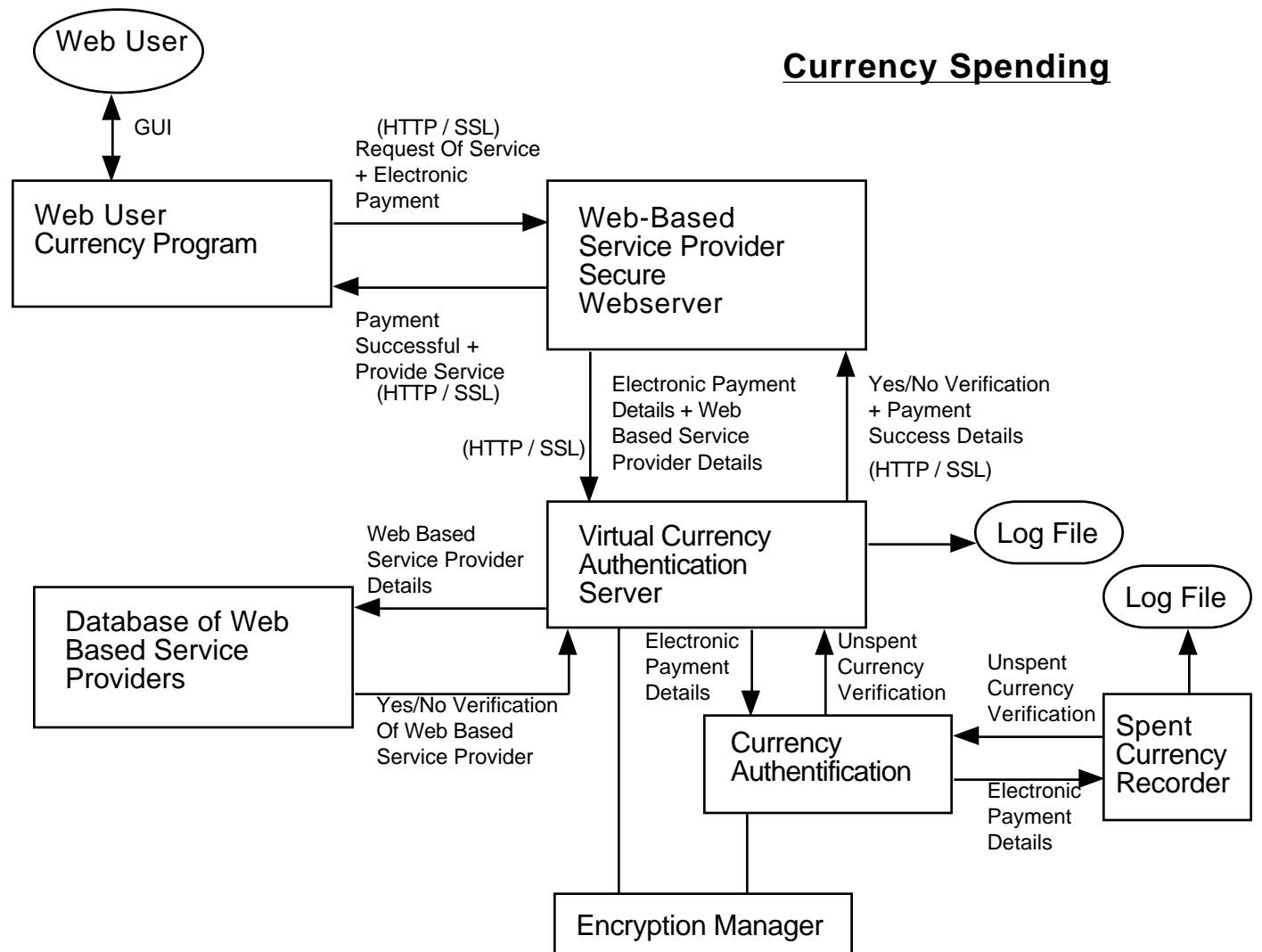
A second point brought up in the review was a request a more detailed encryption description for the client. To this end, two amendments to this document have been made, one to show the lower level functionality and functional requirements of an encryption manager, and second extracts from an essay on electronic payment systems.

Functional Requirements Addition:

Encryption Manager (Used by Currency Authentication Server, Client side Currency Program, and by Web Based Service Provider, as well as the possibility of our implementation of the SSL layer used under HTTP for secure web based transactions)

- GenerateKeys(in: keysize) (out:: publicKey, privateKey)
- EncodeDataWithPubKey(in: publicKey, data) (out: encodedData)
- DecodeDataWithPrivKey(in: privateKey, encodedData) (out: data)
- SignDataWithPrivKey(in: privateKey, data) (out: signature)
- CheckSignatureWithPubKey(in: publicKey, signedData, data) (out: yes/no)

Revised Model requirement for Currency Spending Including Encryption Manager



Programming Methodology:

Chosen subset to write EML specification for:

Virtual Currency Authentication Module, including the currency authenticator and the spent currency recorder. We have also included the Encryption manager. As it will be used by these modules.

This involves the following signatures and functions:

Currency Authentication Module

- ReceivePaymentRequest(output: WebBasedServiceProviderDetails, CurrencyDetails)
- SendVerificationReport(input: WebBasedServiceProviderDetails, VerificationReport)
- LogTransaction(input: WebBasedServiceProviderDetails, VerificationReport)

Currency authenticator

- VerifyAndRecordSpendingOfCurrency(input: VirtualCurrency) (output: VerificationReport)
- VerifyCurrency(input: currency to verify) (output: SubVerificationReportPartA)
- HasValidSignature(input: currency to verify) (output: yes/no)
- HasNotBeenPreSpent(input: currency to verify) (output: yes/no)
- RecordSpendingOfCurrency(input: VirtualCurrency) (output: SubVerificationReportPartB)
- CreateVerificationReport (input: SubVerificationReportPartA, SubVerificationReportPartB) (output: VerificationReport)

Spent Currency Recorder

- ReceiveCurrencyToRecordAsSpent(input: VirtualCurrency)
- ReceiveCurrencyToVerifyAsUnspent(input: VirtualCurrency) (output: Yes/No)

EncryptionManager

- GenerateKeys(in: keysize) (output: publicKey, privateKey)
- EncodeDataWithPubKey(in: publicKey, data) (out: encodedData)
- DecodeDataWithPrivKey(in: privateKey, encodedData) (out:data)
- SignDataWithPrivKey(in: privateKey, data) (out: signature)
- CheckSignatureWithPubKey(in: publicKey, signedData, data) (out: yes/no)

```
(* ----- *)
(* EML Specification for fragment of System *)
(* ----- *)
```

```
(*
The EML Specification is with relation to the mparts of the
models only and may have to be expanded to cover other area's
of the system, but this would require analysis and EML code
for the complete system
```

```
By Iain Reid,
Michael Jane,
Lucas Dixon,
James O'Donnell,
Jai Redden
```

```
*)
(* ----- *)
```

```
signature VirtualCurrencySig =
  sig
    eqtype money
    type loadsamoney
  end
```

```
(* ----- *)
```

```
signature VerificationReportSig =
  sig
```

```

    type report
  end

(* ----- *)

signature WebBasedServiceProviderDetailsSig =
  sig
    type details
  end

(* ----- *)

signature PrivateKeySig =
  sig
    eqtype    key_type

    val size_of : key_type -> int
    val isvalid : key_type -> bool
    (* Check to see if it's prime etc *)
  end

(* ----- *)

signature PublicKeySig =
  sig
    type key_type

    val size_of : key_type -> int
    val isvalid : key_type -> bool
    (* Check to see if it's prime etc *)
  end

(* ----- *)

signature EncodableData =
  sig
    type data_type

    val = : data_type -> data_type -> bool
  end

(* ----- *)
(*

Functional Specification for:
-----
EncryptionManager
  GenerateKeys( in: keysize ) (output: publicKey, privateKey )
  EncodeDataWithPubKey( in: publicKey, data ) (out: encodedData)
  DecodeDataWithPrivKey( in: privateKey, encodedData) (out: data)
  SignDataWithPrivKey( in: privateKey, data ) (out: signature)
  CheckSignatureWithPubKey( in: publicKey, signedData, data ) (out: yes/no)

*)

signature EncryptionManagerSig =
  sig
    structure PublicKey : PublicKeySig
    structure PrivateKey : PrivateKeySig

```

```

eqtype      datachunk

val GenerateKeys : int -> ( PublicKey.key_type * PrivateKey.key_type )
val EncodeDataWithPubKey : ( PublicKey.key_type * datachunk ) -> datachunk
val DecodeDataWithPrivKey : ( PrivateKey.key_type * datachunk ) -> datachunk
val SignDataWithPrivKey : ( PrivateKey.key_type * datachunk ) -> datachunk
val CheckSignatureWithPubKey : ( PublicKey.key_type * datachunk * datachunk ) ->
bool
val CheckKeys : ( PublicKey.key_type * PrivateKey.key_type ) -> bool

(* above is equivalent of saying: gcd( pub, piv ) = 1 *)

(* make sure generate keys function generates correct keys *)

axiom forall i => CheckKeys( GenerateKeys( i ) )

axiom forall i => exists pub_key => exists prv_key =>
  ( pub_key, prv_key ) == GenerateKeys( i ) implies
  PublicKey.isvalid( pub_key ) andalso
  PrivateKey.isvalid( prv_key )

axiom forall i => exists pub_key => exists prv_key =>
  (pub_key, prv_key) == GenerateKeys( i ) implies
  i == PublicKey.size_of( pub_key )
axiom forall i => exists pub_key => exists prv_key =>
  (pub_key, prv_key) == GenerateKeys( i ) implies
  i == PrivateKey.size_of( prv_key )
axiom forall i => exists pub_key => exists prv_key =>
  (pub_key, prv_key) == GenerateKeys( i ) implies
  CheckKeys( pub_key, prv_key )

(* if we have generated a private and public key then things must hold true *)

(* encoding decoding are the inverses of each other *)
(* axiom forall i => exists pub_key => exists prv_key => forall thedata =>
  ((pub_key, prv_key) == GenerateKeys( i )) implies
  thedata == DecodeDataWithPrivKey( prv_key,
    EncodeDataWithPubKey( pub_key, thedata ) ) *)

axiom forall i => exists pub_key => exists prv_key => forall thedata =>
  ((pub_key, prv_key) == GenerateKeys( i )) implies
  thedata == EncodeDataWithPubKey( pub_key,
    DecodeDataWithPrivKey( prv_key, thedata ) )

(* signed data must check ok with the check function. *)
axiom forall i => exists pub_key => exists prv_key => forall a_datachunk =>
  ((pub_key, prv_key) == GenerateKeys( i )) implies
  CheckSignatureWithPubKey( pub_key, a_datachunk,
    SignDataWithPrivKey( prv_key, a_datachunk ) )

axiom forall i => exists pub_key => exists prv_key => forall thedata =>
  ((pub_key, prv_key) == GenerateKeys( i )) implies
  EncodeDataWithPubKey( pub_key, thedata) /=
  DecodeDataWithPrivKey( prv_key, thedata)

end

```

(* ----- *)

signature SpentCurrencyRecorderSig =
sig

structure VirtualCurrency : VirtualCurrencySig

val ReceiveCurrencyToRecordAsSpent : VirtualCurrency.money -> bool
val ReceiveCurrencyToVerifyAsUnspent : VirtualCurrency.money -> bool

axiom forall m => ReceiveCurrencyToRecordAsSpent(m) implies
ReceiveCurrencyToVerifyAsUnspent(m) = false

end

(* ----- *)

signature VirtualCurrencyAuthenticationServerSig =
sig

structure VerificationReport : VerificationReportSig

structure CurrencyDetails : VirtualCurrencySig

structure WebBasedServiceProviderDetails : WebBasedServiceProviderDetailsSig

val ReceivePaymentRequest : unit -> (WebBasedServiceProviderDetails.details *
CurrencyDetails.loadsamoney)

val SendVerificationReport : (WebBasedServiceProviderDetails.details *
VerificationReport.report)

val LogTransaction : (WebBasedServiceProviderDetails.details *
VerificationReport.report)

end

(* ----- *)

signature CurrencyAuthenticatorSig =
sig

structure VerificationReport : VerificationReportSig

structure VirtualCurrency : VirtualCurrencySig

type SubVerificationReportA

type SubVerificationReportB

val VerifyAndRecordSpendingOfCurrency: VirtualCurrency.money ->
VerificationReport.report

val VerifyCurrency: VirtualCurrency.loadsamoney -> SubVerificationReportA

val HasValidSignature: VirtualCurrency.loadsamoney -> bool

val HasBeenPreSpent: VirtualCurrency.loadsamoney -> bool

val RecordSpendingOfCurrency: VirtualCurrency.money -> SubVerificationReportB

val CreateVerificationReport: (SubVerificationReportA * SubVerificationReportB) ->
VerificationReport.report

axiom forall m => not(HasValidSignature(m)) implies HasBeenPreSpent(m) = false

axiom forall m => HasBeenPreSpent(m) implies HasValidSignature(m) = true

axiom forall (x,y) => x/=y implies VerifyAndRecordSpendingOfCurrency(x) /=

```

VerifyAndRecordSpendingOfCurrency(y)
axiom forall (x, y) => x/=y implies VerifyCurrency(x) /= VerifyCurrency(y)
axiom forall (x, y) => x/=y implies RecordSpendingOfCurrency(x) /=
RecordSpendingOfCurrency(y)
axiom forall (x, y) => x/=y implies CreateVerificationReport(x) /=
CreateVerificationReport(y)

```

end

(* ----- *)

```
signature DBOfServiceProvidersLinkSig =
```

```
sig
```

```
    (* Undefined *)
```

```
end
```

```
;
```

(* ----- *)

```
functor VirtualCurrencyAuthenticationServerFUNCTOR(
    structure E : EncryptionManagerSig
    structure C : CurrencyAuthenticatorSig
    structure D : DBOfServiceProvidersLinkSig ) :
VirtualCurrencyAuthenticationServerSig = ?
```

```
;
```

(* ----- *)

```
functor CurrencyAuthenticatorFUNCTOR(
    structure E : EncryptionManagerSig
    structure S : SpentCurrencyRecorderSig ) :
CurrencyAuthenticatorSig = ?
```

```
;
```

(* ----- *)

(* Correspondences to the Models/Functional Specification

VirtualCurrencyAuthenticationServerSig
corresponds to: Virtual Currency Authentication Server

CurrencyAuthenticatorSig
corresponds to: Currency Authentication

DBOfServiceProvidersLinkSig
corresponds to: A link to the database of Web Based Service Providers

EncryptionManagerSig
corresponds to: The Encryption Manager

SpentCurrencyRecorderSig
corresponds to: The Spent Currency Recorder

*)