Financial Cryptography in 7 Layers

Ian Grigg [1] [2]

1998 - 2000

Abstract: Financial Cryptography is substantially complex, requiring skills drawn from diverse and incompatible, or at least, unfriendly, disciplines. Caught between Central Banking and Cryptography, or between accountants and programmers, there is a grave danger that efforts to construct Financial Cryptography systems will simplify or omit critical disciplines.

This paper presents a model that seeks to encompass the breadth of Financial Cryptography (at the clear expense of the depth of each area). By placing each discipline into a seven layer model of introductory nature, where the relationship between each adjacent layer is clear, this model should assist project, managerial and requirements people.

Whilst this model is presented as efficacious, there are limits to any model. This one does not propose a methodology for design, nor a checklist for protocols. Further, given the young heritage of the model, and of the field itself, it should be taken as a hint of complexity rather than a defining guide.

Introduction

Financial Cryptography is substantially complex [3]. For a field that is nominally only half a decade old, by some viewpoints, it is apparent from the implementation work that has been done that many more aspects were involved than envisaged by early pioneers.

Financial Cryptography appears to be a science, or perhaps an art, that sits at the intersection of many previously unrelated disciplines:

- Accountancy and Auditing
- Programming
- Systems Architecture
- Cryptography
- Economics
- Internet
- Security
- Finance and Banking
- Risk
- Marketing and Distribution
- Central Banking

[4]

At such a busy juncture of so many distinctive bases of knowledge, problems are bound to arise. Not only the inevitable confusion and wasted resources, but the difficulty in acquiring technical, management and marketing talent that can comfortably work in the field is an issue.

As a preliminary step to the better understanding of Financial Cryptography projects, it is often of some interest to structure these disciplines into models that aid dialogue, comparisons and decision making.

This paper presents one such model that attempts to describe the field in an introductory manner, as a preamble to greater learning. In this model, the terms Finance and Cryptography are stretched out in order to reveal the disciplines that might have been hidden within the name.

Of course, no one model can plausibly cover the depth and breadth of a complex subject. The intent of this present model is to allow the reader to conceptualise the entire field, identifying the relationships of the disciplines, without spending too much time on the detailed nature of each component. Depth is sacrificed for breadth.

The 7 Layer Model

This paper introduces a 7 layer model, akin to the Open Systems Interconnect Reference Model of networking fame, as shown in Figure 1 [5] [6]. In this model, Finance and Cryptography are stretched out, revealing five more areas of interest.

7. *Finance*	Applications for financial users, issuers of digital value, and trading and market operations
6. *Value*	Instruments that carry monetary or other value.
5. *Governance*	Protection of the system from non-technical threats.
4. *Accounting*	Framework that contains value within defined and manageable places.
3. *Rights*	An authentication concept, with ownership allocated to unit-value, and methods of moving unit-values between unit-identities.
2. *Software Engineering*	The tools to move instructions over the net, and hold numbers and information reliably constant on nodes.
1. *Cryptography*	Mathematical techniques to state certain truths that could be shared between parties for passing value.

Figure 1: Financial Cryptography in 7 Layers

An advantage of this model is traversal from the technical to the application, giving major stakeholders easy points of entry.

We can start at the top, the Finance layer, and work top-down; this is a process of mapping requirements and following them down into lower layers. This might be the place to start if engaged in high-level application discussions.

Or, we can start at the bottom, the Cryptography layer, and describe tool kits to offer the higher layers. From ever more sophisticated lower layers, we can build our way up to offering a smorgasboard of options to the all-encompassing financial applications layer.

Here, we choose a descriptive presentation that traverses bottom-to-top. Later, an example is presented in the reverse order, top-to-bottom.

Cryptography

At the bottom is Cryptography [7]. To some extent, the pure science domain of cryptography solves problems in a mathematical sense only, but it delivers useful properties, including:

- Confidentiality - encryption algorithms
- Integrity - hashes and message digests
- Authentication - digital signatures, hash chains

Cryptography also can solve special problems, when correctly formulated [8]. For example, how can Alice sign a statement of Bob's without being aware of the contents of the statement [9]?

Software Engineering

It takes Software Engineering, layer 2, to usefully benefit from the properties of cryptography. We draw from database theory (atomicity, transactional integrity and recovery) and networking theory (feedback and idempotency) in order to add such properties as reliability and robustness in the face of network and nodal unreliability, or, designed unavailability such as smart cards and handhelds [10] [11].

Software engineering provides us with a practical network. We can talk about sending a message across an open network and know that a message will eventually get to the addressee. With the integrity techniques of the previous layer, we can know that the information received by the addressee is as intended by the addressor. By using the specialised sequences of database theory, we can preserve the integrity of the messages over time, in the face of software and hardware failure.

Rights

With both cryptography and software engineering providing a network upon which we can rely, we can think about distributing messages that are designed to Financial Cryptographic purposes [12]. In the Rights layer, we are looking for a protocol that provides a user with control over assets, in an unequivocable, determinable fashion [13]. Techniques aimed at achieving this include:

- Identity-based systems, such as those operated by banks. Generally, such systems are based on the supply (to an existing account holder) of an account number and password that can access the user's account via an SSL-encrypted web page [14].

- Token Money that emulates the bearer cash instruments with which consumers are familiar [15].

- Transport mechanisms for other payment systems, such as the use of SSL-based systems to carry credit card information.

- Hybrid systems, that eschew emulation in favour of bottom-up solutions more in tune with the power and limitations of the network. For example, SOX is such a system, presented in the next section [16]. A variation on this theme of environmental empathy, the E language is built from powerful capabilities concepts, and can thus be easily turned to Financial Cryptography [17].

- Hardware-based solutions, such as smart cards [18].

although this is not an exclusive list [19] [20].

Accounting

The previous layers provide methods reliable enough to be used for passing something of value, which we call rights, over an otherwise unsuitable network. Now, we need the techniques of Accounting in order to store and manage rights over time, To financial cryptographers, accounting is a mundane field, and it has perhaps been attractive to ignore it, but experience shows that systems without conventional accounting features tend to lose the value entrusted to them.

The techniques of the accounting discipline include double-entry bookkeeping, balance sheets, and the accounting equation [21]. Accounting concepts permit builders of Financial Cryptography systems to build complex systems that guarantee not to lose value as long as everyone follows the rules; and to efficiently identify where the rules are not followed.

The above layer, Rights, defines what needs to be accounted for. As an example, the most basic method would be token money. An accounting model based on tokens or coins would need a simple store of coins for the client. The server would be more complex, requiring an account for unissued value, a float account, and a double spend database that matches the float amount. [22].

Governance

Once there is a guarantee that the digital amounts - the accounting numbers - under management can be securely passed over the net, and stored on nodes safely, we need to cast our view wider to threats outside the technical domain [23].

In any working technology, whether it be trading or cash purchasing, the threat of theft or abuse exists from parties who are trusted to manage the system. This problem, known as *the agency problem*, can be overcome with a wide variety of techniques that here I will label *governance* [24].

Governance includes these techniques:

- Escrow of value with trusted third parties. For example, funds underlying a dollar currency would be placed in a bank account.
- Separation of powers: routine management from value creation, authentication from accounting, systems from marketing [25] [26] [27].
- Dispute resolution procedures such as mediation, arbitration, ombudsmen, judiciary, and force [28].
- Use of third parties for some part of the protocol, such as creation of value within a closed system.
- Auditing techniques that permit external monitoring of performance and assets [29].
- Reports generation to keep information flowing to interested parties. For example, user-driven display of the reserved funds against which a currency is backed [30].

As technologists, we strive to make the protocols that we build as secure and self-sustaining as possible; our art is expressed in pushing problem resolution into the lower layers. This is an ideal, however, to which we can only aspire; there will always be some value somewhere that must be protected by non-protocol means.

Our task is made easier if we recognise the existance of this gap in the technological armoury, and seek to fill it with the tools of Governance. The design of a system is often ultimately expressed in a compromise between Governance and the lower layers: what we can do in the lower layers, we do; and what we cannot is cleaned up in Governance [31].

Value

With a system that provides internal and external stability and security, we are now in a position to assign value to the structure. By value, we mean the unit of account, the meaning of that unit, and the range of numbers that are applicable.

For example, a Value layer might ascribe any one of the following to the virginal numbers of lower layers:

- US dollars with a transaction range of 25 cents up to 500 dollars [32].
- Bonds and stock, representing tradeable assets for the purpose of raising capital.
- Loyalty Points that can be awarded for purchase of goods.
- Public goods such as tonnes of fish, or of public bads such as tonnes of pollution [33] [34].
- Shares in virtual projects.
- Funny money, being internal money for corporate groups.

As the software is somewhat unconcerned about this decision, we could just as easily used the software for any other value - but the business needs to harmonise the security and cost implications.

We might also call this the Contract layer, as any value in electronic form is an agreement between the holder and the owner [35]. It is here that we design the contract that formalises the agreement between an Issuer and a user.

Finance

Finally, on top of the value layer, which provides a structure for financial transactions, we can build our application. As we are concerned with Financial Cryptography, it is convenient to call this last layer the Finance layer. Here, we build an application that adds financial meaning to our designs.

In the Finance layer, we construct any and all applications that might readily be useful to users. For example,

- Retail trading involving the purchase of goods [36].
- Investment trading of securities [37].
- Loyalty systems and Gift systems to encourage repeat business but not to necessarily replace existing methods of payment [38].
- Markets for the fair allocation of limited public goods, such as depletable fishing zones or pollution [39].
- Intermediation of Labour markets [40].
- Closed or limited purpose systems such as shareware sales or corporate group accounting systems.

And many more.

An Example - the Ricardo System

In order to see the model in its descriptive role, I present an example, starting from the Finance layer and working down, by following the roadmap of requirements.

In practice, the model is not a design methodology for setting and mapping requirements, but can be used to reverse-engineer an existing design, for the purposes of presentation and discussion of the mutually agreed contract between the builders and the stakeholders. The following description reflects such a process.

Finance

Systemics, a company specialising in Financial Cryptography, built a system to trade financial securities [41]. The Ricardo system, as an application, required clients and servers to maintain securities, and they communicated using a value system suitable to manage securities and cash [42].

As trials evolved into experience, and strategic analysis of the securities industry evolved into appreciation, if not wisdom, the following primary requirements were built up.

> **Suitable for all securities,**
>
> **Cheap,**
>
> **Fair to all parties.**

These led to many subsidiary requirements:

- There must be no arbitrary limits on the activities of parties.
- Arbitrary amounts of value to be managed.
- All can issue, trade and redeem securities.
- "nothing but net."
- Secure.
- Real Time Gross Settlement, or RTGS.
- Minimise disputes, by eliminating failures / RTGS.
- Privacy, competition both help to keep cost down.

The following discussion concentrates on the value architecture of the Ricardo system built by Systemics, rather than the trading aspects. However, experience shows that trading becomes a tractable problem if the value architecture is solid.

Value

The requirements of the Finance layer result in a derivative requirement for a Value architecture, amongst other things. This Value architecture follows directly after the Finance layer, as the former defines the scope of the security requirements for the remaining layers.

We developed a notion of instruments as follows:

- Definition of Securities is broad:
  - From small to very large value.
  - Extreme flexibility in design of instruments.

- Fair. The system should:
  - protect individual information, but
  - reveal aggregate information to permit user auditing.
  - Especially, it should reveal changes in assets.

- Cheap to operate:
  - Full auditability in the event of disputes.
  - No permission is required to participate.
  - No assumptions are permitted with respect to the use of law or physical force (i.e., cannot rely on laws that "disallow" certain activity or "enforce" certain contracts).

To meet many of these requirements, the notion of a contract for value was developed [43]. This document, which we call a Ricardian contract, documents an agreement between the holder of a security and the issuer of that security, and provides for the flexibility requirement by allowing many and arbitrary clauses to be included.

It is both program- and user- readable, and is signed by the Issuer of the instrument as a binding agreement for any holder of units of that issue. By having a strong basis to determine the nature of the contract, in both human and program terms, we support the auditability requirement, and we can clearly identify the regime for resolution of disputes.

Once set in stone with a digital signature, an identifier can be allocated, leading to efficient description in packets. Thus, this invention requires two things of lower layers - a signature form and a unique document identifier - which are addressed below.

Governance

Once the Value context is defined, indicating the size and nature of instruments, we can address the Governance issues of payment systems and trading.

These are substantially complex [44]. In order to preserve systems intact in the presence of active fraud in the non-technical domain, many disclosure and informational duties abound. In the Ricardo system, we address the governance layer in three main ways:

> **Static Governance**: persistance and availability of contract.

Structural Governance: separation of concerns and ensuring that reliable parties are employed to carry out singular elements of the protocol.

Dynamic Governance: real time auditing of the balance sheet and other key values.

Each of these is discussed below [45].

Static Governance

In static governance, we ensure that the user has the contract, and that all concerned know that the user has the contract [46].

In order to ensure that the Ricardian contract is always present and available to the user, and is continuously binding to the Issuer, we take the message digest of the document and use that message digest as the identifier of the instrument [47].

Consider a message digest, for example, *9c7c9e7bb564224977aea8674623a37407b8f6ee* being a large number of bits encoded in hexadecimal. The user cannot meaningfully interpret this string of apparently random information, so the software (and thus, the software engineer) is more or less forced to maintain a database that describes what the message digest represents. As the contract is readable by software, it makes a superior source of data than any other (such as an intermediate database that holds the contents) and thus we can reasonably assume, to the extent that the software can, that the user has the full contract available [48].

The system will thus ensure that, to all practical intent, the user has the contract. This provides two cost savings, limiting both on-going support and the likelihood of litigation [49].

Structural Governance

Within structural governance, we consider the question of insider fraud, the theft of both digital value within the Financial Cryptography system and of any physical value that underlies the virtual value managed by the system.

With any payment system, there is an ability to create new assets, or misdirect existing assets, all with no more work than a few button pushes. To address this, we use the approach of *separation of concerns* to address the agency problem of holding owners' assets, but protecting them from internal attack. This problem is normally handled by separating out management of day-to-day assets with the creation of assets in the system, and increasing the work required for any fraudulent transactions.

The general schema that is advised to Issuers is as follows [50]. In order to limit the creation of value, for each issuance, a special account is designated as *the mint*, or the creator of value. This account is placed in the hands of a reliable professional source such as an accountant or lawyer, who will hopefully only have an interest in using the account under the probity of the governance regime.

Then, a manager account is designated that receives any new float from the mint, and also returns any redemptions.

It thus becomes the Issuer's responsibility to ensure that the mint account is rarely used, and then with full authorisation and wide scrutiny. Meanwhile, the manager's account is regularly used, but holds only limited amounts of value for day to day requirements.

The above are general techniques that are supported within the Ricardo system, but are as applicable elsewhere. Certain features get specific support, such as value caps on accounts and target account limitatons.

Note how these protection techniques that we use are partly outside the domain of the technical system. Rather than being outside scope, their discussion here is simply a reflection of the claims that the total security of the system is a holistic issue, and governance is the layer where we solve the security challenges that remain after we have attempted to solve as many as possible in the lower layers.

Dynamic Governance

Finally, in dynamic governance, we provide for monitoring of key values by the user community, and thus share the auditing burden. These values can be audited in an issued currency within the Ricardo system:

- Total value of digital float: the value issued by the mint account.
- Amount currently held in the issuance manager's account. From this account is drawn new value to be sold, or into this account, old user value is redeemed.
- The balance sheet of the currency, which is effectively the above numbers, and the total of user value outstanding. As a balance sheet, the total float, minus the manager's account, should equal the outstanding users' total value.
- With some limitations, it is useful to provide summaries of movements such as bought and sold values through the manager's account, the mint account, and number of user accounts.

Mirroring the Governance Model

It is also worth noting that when a currency is reserved by an underlying asset (for example, if a gold-denominated currency had physical metal escrowed to reserve it) then the above governance features should be mirrored for the reserves.

That is, to continue the example of gold, there should be separate parties responsible for the ingress and egress of metal into storage, and there should be independent verification of the number of bars currently placed in escrow.

Accounting

In order to meet the conflicting objectives of privacy and flexibility, Ricardo uses a conventional accounting model with some additional features:

- Accounts are units of allocation of ownership, and are not identities. A user may create these on demand, and likewise dispose of them. Lower layers must provide some mechanism for these accounts.

- Sub-accounts manage a particular form of value within an account. The sub-account is simply the intersection of ownership authentication (the account) with the value description (the contract).
- Transactions are movements of value from one account to another, within the sub-account of the instrument.
- The backend accounting engine is responsible for guaranteeing that each transaction is atomic and persistant. The result of transaction completion is the issuance of a signed receipt.
- Each transaction settles in real time, as measured by the issuance of the receipt.
- Both backend and client keep a list of receipts as the sub-account.
- In order to meet Governance layer requirements for open auditability, some accounts must present balance sheets on demand, and select accounts must be examinable.

Because of the top level requirement for cheapness, the accounting model was designed for complete reliability, right up to the support desk level. It does this by employing a group of non-obvious techniques:

- the Issuer backend forces the client to maintain the same data, as discussed in the Rights layer, below [51]. This helps to reduce the frequency of the "request for information" support call [52].
- A signed request from the user is merely a request for the backend to attempt a transaction, and the backend is at liberty to ignore it [53]. Only the signed receipt is evidence of a transaction [54].
- In order to raise the profile of the signed receipt, balances are not kept anywhere in the system [55].

Using these techniques, the accounting model supports the Finance level requirement of being cheap to operate. If the client software is missing something, then it is a bug, and it properly belongs with the software developer, rather than being covered up as an Issuer help desk problem.

Rights

In order to ensure that owners maintain rights to assets that are managed on the servers, the SOX protocol provides these three major features [56]:

1. Each user creates key(s) which are registered with the server. These keys are as determined by Cryptography layer, below, and are required to provide a unique identifier.
2. Value transfer is via three components:
    a. A key can be used to sign a payment order. This payment order can be directed to a target account, or be open (bearer), and it has a fixed amount of some determined type of value [57].

       In this sense, the payment is analogous to a cheque. It differs from chequing systems in that the SOX payment has no value until settled, whereas a cheque is expected to have value on signing [58].

b. A payment order can be deposited to a sub-account. Settlement depends on a number of checks, such as funds in the source sub-account, and a valid payment order signature from the source key.

c. The Issuer server returns the receipt, mentioned in the above Accounting layer.

3. Finally, in order to cope with network failure, the SOX protocol includes a mail feature, that allows the server to communicate reliably with the client. Packets that must be delivered to the client are placed in the mailbox, and returned on every mail request. Each piece of mail must be signed for, and if not signed for, is simply returned again.

   In the context of the value transfer above, there is only one piece of mail, being the receipt.

SOX is a flexible protocol. By replacing the deposit request, above, with trading requests, it can be used for market trades as well as settlements [59].

In the trading context, requests that are implemented emulate standard market functions such as looking at the order book for an instrument, placing an order (buy or sell), monitoring the progress of an order and cancelling an order. The SOX mailbox is used for the return of orders (assets and results).

Software Engineering

SOX as a protocol spans both the Rights layer and the Software Engineering layer.

In networking, every transmission must be considered as a contender for failure. As a corollary to this, relying on a connection-oriented protocol such as TCP will not guarantee reliability, as its promise is only that that the data that gets there is the correct data as sent [60].

To cope with these problems, SOX asumes a datagram network only, and handles reliability itself [61].

Secondly, it bases communications on a request model, with each request being independent of the next, and each request only being complete when positive feedback is received.

Thirdly, SOX requests are idempotent, so they can simply be repeated until some confirmation comes back that one attempt has succeeded. Unique request identifiers are included and used to filter out retries.

Fourthly, in order to implement SOX, a client must treat each request as unreliable. For example, when a payment is written by the current client, that payment is recorded as pending, which is eventually matched up with a receipt arriving from the Issuer.

Or, the client gives the user the opportunity to cancel the payment simply by re-using the unique identifer, and thus stopping the lost payment ever settling. In this way, where it is impossible to guarantee a result, Ricardo extends reliability management out to include the user.

Finally, SOX includes a *comms layer* that provides for key exchange for confidentiality and authentication purposes.

Cryptography

The cryptography demanded by the upper layers includes:

- A key exchange method. Newer generations of SOX use Diffie-Hellman key exchange, whilst older versions use RSA.
- A secret key encryption algorithm. IDEA was used in the past, Triple-DES in current versions, and one of the AES algorithms is a likely contender for the future.
- A public key signature scheme. For newer, DSA, and for older, RSA.
- A message digest method. SHA-1 is used, although MD5 has been used in the past.

All of these algorithms are implemented as part of Cryptix, an open source project that was spun off by Systemics in 1996. Cryptography and the cryptographic techniques used in Ricardo are well discussed in the literature [62].

Concluding Remarks

Advantages of the Model

The model works well in tackling and reducing the inherent complexities of Financial Cryptography. It does this by dividing the field into 7 areas, and providing an interconnection method (layering).

Once a project is so layered, professionals within different disciplines can clearly deliniate those areas within their expertise, and those which call for other specialisations. Thus, lawyers can recognise the Governance layer as their bailiwick, and pay due attention to it. Other layers can be treated, more or less, as black boxes, interconnecting with requirements down and features up. Likewise, programmers can concentrate on Software Engineering and Rights, with more interest in Accounting than Governance.

A project manager, with responsibility for delivery of a Financial Cryptography system, finds this even more powerful, as the model offers a natural checklist and vocabulary for coordinating the activity.

As an analogue of the 7 layer ISO Reference Model, it also wins on easy familiarity with what we are trying to achieve.

What the Model is Not!

The designation of 7 layers does not, in and of itself, encourage the design or implementation of system components that fall neatly into one layer or another. The notion of a layer 3 protocol providing services to a layer 4 protocol simply does not work in practice [63].

Likewise, this model is not a design methodology. The description of a top-down requirements process is illusory, and in practice, the requirements analysis is more modelled by continuing and volatile negotiations between the layers. Whilst it is descriptive to state that a requirement is bouncing up and down between layers one and five, inclusive, this does not give much assistance to a team leader in assisting a design process.

Criticisms of the Model

It is easy to criticise any model, as by definition, a model falls short of reality. Here are some points:

- Does the set of layers describe Financial Cryptography accurately? Hettinga suggests, perhaps only partly in jest, the name *cryptographic finance*, implying that layers one to three may have greater claim to the original term [64].

- The 7 layer model is static rather than dynamic. Once described, it works, but how did we manage to construct it in the first place?

- Are there really 7 layers? Are the layers as described? About each of the different layers we can ask many questions, including some troublesome ones:

  - is the carve-up between Cryptography, Software Engineering and Rights the best one?
  - does Accounting deserve a full layer?
  - does Governance really sit between Accounting and Value?
  - can we quietly ignore Hardware, or slid it into Software Engineering, where most applicable expertise lies?

  My answer, today, is 'yes' to each, but only time will provide the real answer.

- The top-down requirements example of Ricardo seems to indicate a natural design flow or methodology, but in practice the design process does not follow that path.

  Experience has shown that concentration on Finance, and then Value is worthwhile. Then, the vertical flow breaks down; in particular, a lot of time is spent bouncing around the lower 4 layers in a negotiation for the best compromise, with occasional forays upwards in order to tune the requirements. Governance always seems to come last in the design process, as its contents are an admission of what the rest of the architecture has failed to cover.

- Layers one to four, up to Accounting, are fairly solid in terms of their disciplines, practices and methodologies. Layers five and up (Governance, Value, and Application) are less well-defined.

  This might represent a flaw, or it might indicate an intrinsically messy area. Perhaps coincidentally, the ISO Reference Model exhibits the same pattern.

I believe that these criticisms are valuable in indicating that the model is promising, as they help to refine ideas, rather than destroy them.

References

[1] Ian Grigg can be reached at iang@systemics.com . He is a founder of Systemics, Inc, a developer of Internet Financial Systems software. Back.

[2] This paper was presented at FC00 and is originally published in the Proceedings of *Financial Cryptography Fourth International Conference*, Anguilla, British West Indies, 21st - 24th February 2000. A web copy is located at http://www.iang.org/papers/.

The model was initially inspired by discussions on the DBS mailing list, and was progressively refined in discussions with Twan Van Der Schoot. This paper has also benefitted from review remarks by Ian Brown, Zooko Journeyman and Rachel Willmer. Back.

[3] The term Financial Cryptography was invented by Robert Hettinga as a name for a conference held annually in Anguilla. Back.

[4] Ian Grigg, *Virtual Finance Report* , Digital Trading, November 1997. Back.

[5] Search on Google for ISO OSI Reference Model Seven Layer Back.

[6] It is mostly coincidence that there are 7 layers, and it may change if we find compelling reasons to add or subtract layers. Back.

[7] The Cryptix Resources Page lists popular cryptography books, including links for purchasing. Back.

[8] A large area of such problems, including the blinding property, is described in *Rethinking public key infrastructures and digital certificates --- building in privacy* Stefan Brands, ISBN 90-901-3059-4, 1999. Back.

[9] The blinding concept is most easily accessible in Achieving Electronic Privacy *Scientific American* David Chaum, August 1992. Back.

[10] *An Introduction to Database Systems, Volume 2*, by C.J Date, 6th Edition, Addison Wesley, 1995 Back.

[11] I studied with this text book nigh on 20 years ago, and it still appears to be the main text in the field of protocols and networking: *Computer Networks*, by Andrew S. Tannenbaum, 3rd ed., Prentice Hall, 1996 Back.

[12] A fullsome page of links to electronic purses - implementations of Rights protocols - is included in Leo Van Hove's bibliography. Back.

[13] I am indebted to Mark Miller for providing me with the name of this layer. Back.

[14] At the time of writing, the canonical example would be www.e-gold.com which provides identity-based access to currencies reserved in precious metals. Back.

[15] For example, the eCash (tm) tokens as implemented by eCash Technologies, Inc. Back.

[16] Originally presented in the Gary Howland, Development of an Open and Flexible Payment System, Back.

[17] Mark S. Miller, Chip Morningstar, Bill Frantz, Capability-based Financial Instruments, accepted by Financial Cryptography 2000, Anguilla, February 2000. Back.

[18] Systems such as Chipper and Mondex. Note that there is no need for a new hardware layer - the distinction here is that the hardware is supplied, rather than assumed. Back.

[19] For many more examples of theoretical approaches, see *Financial Cryptography* First through Fourth International Conferences, Anguilla, British West Indies, February 1997-2000. Back.

[20] For examples of approaches that have reached practical implementation stage, if not to market, see *Edinburgh Financial Cryptography Engineering* a new workshop that includes presentations of running code only. Back.

[21] Check any basic accounting text book for these terms. Google may provide some assistance on these terms. Back.

[22] As a wider comment, it is possible to model any electronic value scheme as a method of accounting. See Alan Tyree, The legal nature of electronic money.

Whilst a valuable modelling exercise, caution is advised, as most conclusions drawn from such exercises are too broad. Specifically, institutional observers tend towards a line of logic: "it can be modelled as a series of accounts, therefore it should be regulated like banking;" such an approach is fraught with difficulties and unlikely to be satisfactory. Back.

[23] For general articles on the Governance aspects of Financial Cryptography, check John Muller's ABA site Electronic Financial Services Resources. Back.

[24] The Agency Problem:

> Also sometimes referred to as the principal-agent problem. The difficult but extremely important and recurrent organizational design problem of how organizations can structure incentives so that people ("agents") who are placed in control over resources that are not their own with a contractual obligation to use these resources in the interests of some other person or group of people actually will perform this obligation as promised -- instead of using their delegated authority over other people's resources to feather their own nests at the expense of those whose interests they are supposed to be serving (their "principals"). Enforcing such contracts will involve transaction costs (often referred to as agency costs), and these costs may sometimes be very high indeed.

A Glossary of Political Economy Terms Paul M. Johnson. See also Google. Back.

[25] Michael Froomkin's writings on Separation of Powers. Back.

[26] Robert Hettinga suggests some models in The Players Back.

[27] In Ricardo documentation, and also further below in the section on Structural Governance I suggest breaking up the system into 5 parties, Owner, Mint, Manager, Users, Operator. Back.

[28] See Jane Kaufman Winn's writings on the validity of current contracts in governance: Jane Kaufman Winn, Couriers without Luggage: Negotiable Instruments and Digital Signatures, *South Carolina Law Review*, 1998. Back.

[29] See the DigiGold Page for an example of a real time report on the currency balance sheet. Back.

[30] See the e-gold Examiner for an example of a real time report on reserves. Back.

[31] See Jane Kaufman Winn, op cit, for a classic description of the Certificate Authority industry's attempts to clean up a poor security model with an implausible contract. Back.

[32] 25 cents is a fair minimum for credit cards, due to the cost of these transactions. $500 is a popular upper limit imposed on smart cards by the threat model (actually, it is 500 of the local unit, for some obscure reason). Back.

[33] For a description of Individual Transferable Quotas - ITQs - describing instruments for quantities of fish stocks, see *Policy* , Fencing the Oceans A Rights-based Approach to Privatising Fisheries, Professor Birgir Runolfsson, Autumn 1998. Back.

[34] Ian Brown points out that pollution is in fact a public bad. Back.

[35] Ian Grigg, Universal Value, *work in progress*. This is introduced later in the example. Back.

[36] For example, this was the target application of Cybercash Inc, First Virtual Inc, and DigiCash BV (now eCash Technologies Inc). Back.

[37] Digital Trading, op cit. Back.

[38] For example, see the so-called second wavers: Beenz, Flooz, Cybergold. Back.

[39] See Fencing the Oceans, op cit. Whilst not discussed in the article, there are a small number of marketmakers in Iceland that work the thin market in ITQs. Back.

[40] Ian Grigg and C. Petros, *Proceedings of Financial Cryptography* , Using Electronic Markets to Achieve Efficient Task Distribution, February 1996. Back.

[41] Ian Grigg and Gary Howland designed the Ricardo system in 1996-1997. Back.

[42] The Ricardo system is currently in use for a series of metal based currencies managed by DigiGold. Back.

[43] Ian Grigg, *work in progress*, Universal Value, Back.

[44] Many designs of Financial Cryptography systems have limited Issuers to being banks, which allows the designer to assume away many complications. Back.

[45] This section is based upon Ian Grigg, *Talk on DigiGold Governance*, Financial Cryptography 1999, commercial sessions. Back.

[46] The same logic would also imply that the user must have access to dynamic trading information such as prices, but we pass over that here. Back.

[47] Having abstracted the contents from the identity of the document by taking a message digest of it, we can discuss value, from payment systems perspective, as being fully and uniquely defined by the message digest. This ensures that the Issuer of the security cannot change the terms of the contract in any way without offering to the user terms for exchange. Back.

[48] This also has a secondary effect of shortening the distance between the contract and the software that manages it, thus simplifying the design. However, the prime objective was, and remains, a system where we know that the user has strong access to contract information. Back.

[49] Such a scheme might not prevent the software engineer from providing a client application that misrepresents the contract. However, this would be an issue between the user and the software supplier, rather than the system itself, especially, the operators of the system and issuers of securities would clearly not be at fault. Back.

[50] This is described more fully in a FAQ question on Structural Governance Back.

[51] In order to force the client to maintain the data, the SOX mail facility, introduced in layer 3, requires signatures for all important documents such as receipts. Back.

[52] Or, more correctly, to treat such a support call as a bug, in that the client is not making the information available. Back.

[53] A signed request from the user has more meaning to the user - the client software must keep track of these as promises to pay, and in this sense, the system is analogous to a cheque system. Back.

[54] The receipt includes the authentication request supplied by the client in order to provide the chain of authentication back to the user. Back.

[55] In programming terms, stored balances are banned. The balances that are displayed by the software client are calculated on the fly, including every time the client redraws. Getting this right has proven to be a sizeable cost in development time, but it is believed that the requirements are valid and the costs are covered in the long run. Back.

[56] SOX is variously *Systemics Open Transactions* or *Secure Open Transactions*. It is discussed by its original author in Development of an Open and Flexible Payment System, Gary Howland, 1996, and also in an Executive Summary. An implementation exists in open form as a part of the WebFunds Project. Back.

[57] SOX provides a string or byte array that determines the type of value, which is open as an implementation detail. But, practically, this is the unique identifier for the Ricardian Contract as discussed in the Value layer. Back.

[58] Note the way in which SOX melds with the Internet, as implication of layer 2. When passing a payment to someone on the other side of the planet, that payment only has value if it is settled and cleared by the Issuer. Otherwise, the payment is an uninteresting series of bits, with similar value to any other random nonsense.

In contrast, the passing of rubber cheques is illegal in some countries, and traumatic in most others. SOX payments are not cheques in that sense. Back.

[59] The value Issuers are distinct servers to market servers, it is just the protocol that is common. The protocol can also be used for other purposes, wherever a primary requirement is made for a reliable delivey. Back.

[60] The specific problem with a connection protocol arises when the connection dies. Did the last few bytes make it to the other end or not? With such protocols, there is generally no way to recover from this uncertainty *without building an additional reliable protocol over the top of the first*. Which of course raises interesting design questions that may lead to alternate paths such as connectionless protocols. Back.

[61] SOX packets can, and are, sent over TCP connections, but mostly so that firewalls may be easily navigated. Back.

[62] The Cryptix Resources Page. Back.

[63] Indeed, in my opinion, neither is it useful for networking. For critiques of the OSI 7 layer models, see M.A. Padlipsky, *Elements of Networking Style*, and RFC 874. Back.

[64] Robert Hettinga, email on *dbs@philodox.com* Back.