The Ricardian Contract

Ian Grigg
Systemics, Inc.
iang@iang.org

Abstract

Describing digital value for payment systems is not a trivial task. Simplistic methods of using numbers or country codes to describe currencies, and ticker tape symbols to issue bonds, shares, and other financial instruments soon run into shortcomings in their ability to handle dynamic and divergent demands. The seemingly arbitrary variations in the meanings of different instruments are best captured as contracts between issuers and holders. Thus, the digital issuance of instruments can be viewed as the issuance of contracts.

This paper proposes that the contract is the issue. A document form is described that encompasses the inherent contractual nature of the financial instrument yet copes with the requirements of being an integral part of a payment system.

1. Introduction

Little work has been done on classification and description of value in the field of financial cryptography. This paper presents the Ricardian Contract, a method to identify and describe issues of financial instruments as contracts [1]. It was originally developed by Ian Grigg and Gary Howland as part of the Ricardo payment system.

1.1. The Origins

The original application was a bond trading system [2]. For trading, a basic component is a transfer or payment system which receives and acts on transfer instructions to move instruments (cash, bonds) from one account to another. Each instruction therefore must identify the instrument.

A means was required to capture, identify, and describe the traded instruments. There are thousands of bonds, and potentially millions of other instruments that could be issued and traded, and each has unique characteristics that are not amenable to compression into databases. To such a system, cash is no different to

bonds, and requires the same description.

1.2. The Problem

When someone issues a currency (or bond or share) over the Internet, what is it? What does the recipient have?

Few systems for issuance of value (payment systems) treat these questions adequately. They generally refer to existing external units of currency and tidy up loose edges in a *user agreement*. For example, Paypal, an issuer of dollars, relies on the familiarity of the US dollar to define much of its service offering. Gold issuers lean more heavily on their user agreements as the metal unit is not so familiar.

For trading, it is not sufficient to refer to well-known familiar references, as each instrument is different in finicky ways and these differences matter to traders. Even with currencies, however, the user has difficulty in determining the security and safety of one dollar with respect to another.

Classification by numbers or symbols is a starting point. Almost all systems of digital issuance identify their basic issue by allocating numbers or letters as currencies (for example, 840, "USD", "AUG" [3]). These systems runs into trouble quickly.

An issuer with many currencies or many issuers with the same nominal currency raises difficult questions. Can an issuer have two or more dollar units? For example, within ISO3166-1, there are three different US dollars: 840/USD (*cash*), 998/USS (*same day*), and 997/USN (*next day*). Similarly, how does one Digital Gold Currency ("DGC") differentiate his gold over that of another issuer, when all are known as "AUG"?

1.3. The Solution

As bonds are, at their essence, contracts between issuers and users, our problem reduces to one of *issuing contracts*. Whereas other issues *have* contracts, our issues *are* contracts.

Our innovation is to express an issued instrument as a contract, and to link that contract into every aspect of the payment system. By this process, a document of some broad utility (readable by user and program) is drafted and digitally signed by the issuer of the instrument. This document, the *Ricardian Contract*, forms the basis for understanding an issue and every transaction within that issue.

By extension, all issues of value, such as currencies, shares, derivatives, loyalty systems and vouchers, can benefit from this approach.

1.4. Structure

This paper is structured as follows. In Section 2, we discuss conventional approaches to identifying and describing issuance, and explore questions and doubts surrounding these approaches. Then, in Section 3, a design to express issuance as a contract is presented. Finally, in Section 4, concluding remarks are added.

2. Issues of Value as Contracts

2.1. A First Generation Scheme

Consider the case of the pioneering digital cash scheme, eCash, as originally fielded by DigiCash BV. The first valuable currency, issued by Mark Twain Bank of the USA, was identified with the number 4. Lore has it that the early system allocated a small sequential number to each currency. Test systems had already acquired 0,1,2,3 and thus 4 was the next. DigiCash's marketing assumptions then changed to assume one issue per country. In time, this scheme was adjusted to issue currencies numbered after international dialling codes (e.g., 49 for Germany, 61 for Australia). The shortfalls of this scheme became apparent, so a new design was created [4]. One 32 bit number to describe the issue was used, on the pragmatic assumption that this would be large enough to cover foreseeable eventualities.

Yet the strains of *one issuer, one currency* were obvious almost immediately. A more advanced scheme could use a tuple of (*issuer*, *currency*) to describe a system whereby each issuer is empowered in some sense to issue multiple competing currencies [5]. It is easy to generalise this system by adding additional elements to the tuple: (*issuer*, *type*, *identifier*) tuple [6]. For example, a zero coupon bond issued by the Joint Universal and Nationwide Keiretsu that pays out in January of 2100 might have a tuple of (JUNK, zero, Jan_2100).

2.2. The Trouble with Numbers

Numbers as a space for identifying digital instruments are limiting, and having tuples as an extension is not really an answer.

Firstly, what do they describe? In the case of electronic cash systems, they can describe currencies and issuers. Is it one or both, and how do we generalise to other aspects? Secondly, what surety do we have that what is described is accurate? Whilst a lot can be achieved by simply relying on the reputation of the issuer, financial insiders know that the real value is expressed in the detail and the reliability of the claim. Thirdly, how are the numbers derived? Is a central registry required, or can any issuer of digital value acquire a number as per local requirements? Finally, is there a limit to the space? Integer numbers as expressed in packets are generally limited to some quantity of bits, such as 32. For practical software engineering, there need to be limits, but do these limits need to limit the business possibilities?

2.3. The Challenge of Success

Any successful system will be used in ways that make it appear to be broken. As software engineers, we need to present our inventions with the humility of toolmakers for future generations of builders, not as bureaucrats planning the zoning of the digital commerce space.

What happens when we have gone through the early adopters, dominated the moms and pops, and competition is fiercely turning onto our elderly retired set? Imagine mints in the pockets of billions of idle game-playing senior citizens. Or, imagine a world with an issuer of digital loyalty points on every parking meter, or where students must pay for tuition with shares of future earnings. Already we have seen popular musicians selling bonds backed by their music [7], and proposals for software bug fixes financed by securitized issues to anonymous users [8].

2.4. The Zero Coupon Bond

Consider the zero coupon bond, an instrument that pays a *face value* of a currency on a given date. The *zero* is perhaps the simplest general financial instrument in common use, and it formed the benchmark for our design.

To describe the face value, the currency of the face value, and the expiry date of this bond, we would add additional elements to the above tuple. But this is only a beginning. In his description of Eurobonds, Noel Clarke expects dozens or hundreds of fields [9]. If we examine just one of these characteristics, for example *Event-Related Put Options*, we find that a bond needs to describe what happens in the event of:

- a hostile or friendly takeover of the issuer,
- a takeover by the issuer of another party,
- a recapitalisation,
- a repurchase program by the issuer of its own shares, or
- a distribution of assets above a certain percentage of the issue's net worth.

These items bind tightly to the instrument in question, but they represent difficulties to the software architect. We can make a number of observations.

Firstly, each event is not simple. Today, one may be able to shoehorn the notion of "a hostile or friendly takeover" into a single name-value pair, but this would not survive the evolving scene of regulation and litigation that applies to such events.

Secondly, there is no reason to believe that the above list is complete.

Thirdly, not only is it going to be hard to design a single field of any sort to cope with these, they are mostly going to be full of legal text.

Consider a data layout point of view. To describe the document that forms the basis of a bond we will need a tree-structured database of tuples, as a minimum. More, that layout is only going to work for one instrument, or one extremely tight, nearly fungible set of instruments.

2.5. Cash is King

Currencies, or cash, might be that tight set. After all, a dollar is a dollar is a dollar. Can we describe money with some simple set of tuples? Even for cash, we argue that a layout of tuples is not sufficient.

Take the case of a digital dollar issued by a bank. The digital dollars would be derivatives, often backed by deposits in the same amount. This may be sufficient for marketing purposes but it would not survive a serious financial analysis.

Compare such derivative dollars to those issued by the US Federal Reserve Board. The Fed has yet to deny acceptance of its notes if presented with same, if only as a claim on another bunch of the same instrument, or for taxation liabilities. Radical interpretations aside, the Fed has never filed for bankruptcy and remains a pretty solid bet.

The same cannot be said of just any bank issuer of derivative dollars. Its digital dollars would be backed by deposits with ... the very same institution. Such a bank can close its doors at any time, and, given the history of the banking sector in the 20th Century, an analyst should take this risk seriously. Further, in the USA at

least, the FDIC has already ruled that funds so held on a user's PC are considered to be uninsured deposits [10].

This is not to suggest that any given bank is about to close doors, but to ask what happens when an issuer does indeed default on its promise?

Any holder of any asset will carry a risk. A holder of electronic dollars will carry the risk that the issuer fails, and the holder of another issuer's dollars carries a similar, comparable, but *distinct* risk. Each of those risks result in a cost, which should be subtracted from the face value of the dollar to calculate a comparative value. In this risk distinction lies the inescapable fact that any given dollar is not of constant value, even when measured against some well-known dollar such as that issued by the Federal Reserve.

2.6. The Fine Print of the Contract

If there is no such thing as a single dollar, what is left? Clearly, we must describe each and every dollar for what it is. This would seem to be a task of fine print and detail, and, indeed, *every distinct issued currency is a distinct contract between the issuer and the holder.*

A contract can encapsulate the detail. Consider the original sovereign currency contracts, in which the issuer promised to pay the bearer in ounces of precious metal. That is four datum in the contract already: which sovereign, "pay to bearer," what to pay, and how much of it.

So it is with every bond, every currency, and any financial instrument of any complexity In fact, within the digital domain, the question of how to treat a financial instrument reduces in great part to how to treat a contract.

Or, an issue *is* a contract. Issues within other payment systems have contracts but only as adjunct documents such as user agreements. Often, their role and importance is subject to battles; Marketing wants them hidden, while Legal asks for them to be thrust in the user's face at all times.

Once we accept that the issue is a contract, the task becomes simple: create a contract that can be linked into the payment system as the centerpiece. That is the subject of the next section.

3. A Digital Contracts System for Issuance

Almost all aspects of Ricardian Contracts are best seen by examining examples, and this section only briefly covers the salient details, before discussing the ramifications. Examples can be found at webfunds.org/ricardo/contracts/ .

3.1. Definition

A Ricardian Contract can be defined as a single document that is a) a contract offered by an issuer to holders, b) for a valuable right held by holders, and managed by the issuer, c) easily readable by people (like a contract on paper), d) readable by programs (parsable like a database), e) digitally signed, f) carries the keys and server information, and g) allied with a unique and secure identifier.

In the simplest possible terms, a Ricardian Contract is a document defining a type of value for issuance over

the Internet [11]. It identifies the Issuer, being the signatory, and any terms and clauses the Issuer sees fit to add in to make the document stand as a contract.

The same document has to be both readable by people and parsable by programs. The Ricardian Contract is formatted as a text file that can be easily read (displayed or printed), and programs can convert it into internal forms for searching for name-value pairs. It includes a special section for each type of contract, such as bond, share, currency, etc. Further sections within describe, in program-parsable terms, usage of decimal points, titles, and symbols.

As legal signatory, the Issuer signs the document in the OpenPGP cleartext form with his contract signing key [12]. He includes the full chain of OpenPGP keys within the document to permit programs to directly verify and authenticate.

To uniquely identify the contract, any user can calculate a *canonical message digest* over the clearsigned document. This message digest is included in all records of transactions, and provides a secure (unforgeable) link from the document to the accounting of the issue.

E.g., e3b445c2a6d82df81ef46b54d386da23ce8f3775 is the full message digest for Systemics Inc's issue of prepaid services dollars. Commonly called a hash, the message digest is a cryptographic technique to create a relatively small number that is one to one with the document. That is, for each document, there is only one hash, and the hash refers uniquely to that document. The algorithm is the well-known standard, SHA1.

3.2. Some Observations

The following observations highlight how strong the result is.

Hash Limits Frog-Boiling. A gradual change in contract by the stronger party over time is known as *frog-boiling*. The stronger party is generally the issuer, and can be expected to change the contract if there is a benefit. This is a frequent attack. One result of the use of the hash identifier is that neither party can change the contract arbitrarily or surreptitiously.

To see this is true, we need to examine the records that refer to the hash. An application can sign all important records (e.g., payments, tokens, receipts, balances), and these signed records include the hash of a Ricardian Contract. The hash within the record cannot be changed without losing its ability to pass a test of signature validity. Likewise, the contract cannot be changed without losing its relationship to records already signed and delivered. In other words, every record, held by every user, incorporates an unalterable copy of that hash. Any change to the contract creates a new hash, and that new hash is not the one which the users have or value.

This crystallises the contract for both parties, stopping the stronger party from modifying the contract subtly at some later stage. To some extent, this redresses the imbalance of power between provider and customer in the offering of a form contract. The lesser party has no option to negotiate, but neither has the greater party the option to claim a distinct contract at a later time. The limitation comes at some cost as it can be a nuisance for the support team of that financial instrument.

The Ricardian PKI Delivers Clarity. Ricardian Contracts carry their own Public Key Infrastructure ("PKI") with them. The Issuer's top level public key is included in the contract, and it signs his contract-signing key, also included. The contract-signing key signs the contract itself.

This achieves several things. Firstly, client software can check the entire digital signature chain in one automated sequence.

Secondly, there is no need for a complex multi-party PKI. All the keys are present, and there is no need to go looking for them on the net. This eliminates substitution attacks, whereby a key that might pass some checks could be inserted in some key lookup phase. It also reduces costs dramatically.

Thirdly, the canonical hash of the contract also represents a signature on the contract. It is recorded in all relevant records, and thus entangles the contract with those activities [13]. Once the contract has been in play for a while, it establishes its provenance through presence and reliance by the user public. This provides far more persuasive evidence than the issuer's signature itself; once the issuer and the public have spent time and money relying on the contract, via the hash, it is hard for the issuer to renege on the nature of the contract or his signature.

The result is a PKI that delivers strong end-to-end reliability, based on a single document. This is simply not present in other designs for PKIs [14]. This reliability pays off in the dispute resolution phase, where, we suggest, the Ricardian Contract can stand alone on its merits and requires no complex descriptions of PKI, digital signatures, or references to uncertain third parties to bolster its provenance. By including the keys, we can draw a couple of simple lines within the contract, asserting "this key signs that key, and that latter signs the contract. The first key is the top-level key of the individual that signed this contract. That's the whole story, mi'lud."

Validating the Issuer's Key. All good crypto protocols divide into two parts, the first of which says to the second, "trust this key completely."

The top-level key of the Issuer ultimately authenticates the contract. The keys and other information in the contract also permit a protocol such as SOX to bootstrap a strongly secured connection to the server [15].

How then to verify that this ultimate key is really the Issuer's? This is not difficult. The business process of digital issuance involves a great deal of relationship-building between Issuers and Users. Many different interactions involve chances to establish trust. For example, from his web site, the Issuer can publish the contract, keys and hashes, and have other sites mirror them. The value so issued will be distributed via payments that include the hash. An already trusted party usually delivers these payments. The payments validly identify the contract, and derive their own validity, via the hash.

Contrast this to assumptions in the x.509 PKI behind SSL/HTTPS browsing (the following is highly debatable, but is presented for comparison only). In that PKI, it was originally claimed that a user would present her credit card to sites with which she had no prior relationship and no way for her to establish the provenance of the site's key. Thus, a trusted third party, the Certificate Authority, was put in place to confirm the key.

Payments, trading and matters of finance are fundamentally relationship-rich. The nature of money and finance is that participants always conduct their own due diligence, they prefer to listen to peers they already trust, and do not readily accept the word of an independent party. Thus, there is no place for a central third party to stand in and authenticate players. Before the user desires to place any value on a given payment, she has almost certainly been made aware of the contract via other means.

Presumption of Possession. The use of the hash as an identifier is a compromise as it is unintelligible to

humans [16]. Yet this very compromise delivers an unexpected benefit: *Use of the issue leads to a presumption that the user has the contract*. To use an issue of value, such as a currency, the user must have the hash in the applicable records. That is, if the user receives a payment, that payment record will include the hash. As the hash is not descriptive, this implies that the user has the contract in order to interpret the issue.

To see that this is true, imagine having a record with the hash but without having the contract. The first thing the user will need is a database of parameters telling her what the hash refers to. Unlike a payment in 10 of "GBP", a payment of 1000 in "972097bb..." is not intelligible.

Yet how could software predict what the user needs to know about the hash? Very quickly it becomes apparent that the software is better off storing the source of the information - the full contract itself - as it removes an unlimited degree of complexity in storing intermediate or secondary information.

Software can still function with only the hash. However, it would be entirely blind to the semantics of the instrument. Such a cavalier approach might be acceptable for communications and storage, but for user software, it is equivalent to a traumatic failure. To cope with this, the client-side software takes especial care to acquire and keep contracts. Hence, we can state the presumption with some degree of confidence: in a functioning system, the user has available the full Ricardian Contract (albeit under software control).

This is only a small step for the client software, but is a giant leap forward for the relationship between the issuer and the holder. Specifically, having a strong presumption that the user has the full contract available will simplify many legal aspects about the issuer's responsibilities. (We suggest and thus acknowledge the legal ramifications of the term *presumption*, but neither space nor expertise permits more in this paper.)

3.3. The Four Corners of the Page

The Ricardian Contract delivers a rich source of primary, complete information. The full story is right there in textual form, in parsable parameters, and in the signature chain. Thus, within a dispute, a hostile legal attack has less room to manoeuvre, and can only confirm the facts as laid out in the contract.

Our intent is that the contract is the beginning and the end of the discussion; we call this principle *the rule of one contract*. The legal fraternity refers to "the contract being bounded by *the four corners of the page*." By showing how we have carefully laid out a readable document, with a verifiable digital signature, and an unforgeable identifier linking to every record, we can more readily ask the judiciary to accept that the single document which is being presented is indeed the valid contract agreed to by the parties.

4. Conclusion

The contract is the keystone of issuance [17]. Our innovation is to express all the salient details of an issuance as an unforgeable contract, unforgeably linked into every action within a payment system. In this way, financial innovation can develop along the lines it always has done - by means of innovation within contracts. By translating the institution of the contract into the digital domain, we build upon centuries' and even millenia' worth of experience in documenting, sharing and disputing the meaning of agreements between parties.

4.1. The Challenge of Complexity

To capture complexity, we can put documents such as contracts into electronic form and sign them using digital signature technologies such as OpenPGP. The result is a reasonable analogue of the paper and ink contracts that most people and businesses are familiar with, bolstered with cryptographic integrity.

With the hash as the identifier, software can now uniquely identify a given financial arrangement and can confirm a strong chain of signatures. The hash strongly implies the user has the contract available at all times, and it cannot be changed without being noticed.

The Ricardian Contract delivers one huge benefit to the issuer - clarity in many legal and customer support questions. The user benefits from lower overall costs, and better presentation of information, within a more consistent framework.

4.2. Lessons Learnt

The form has been in successful use since 1996. Since that time, it has delivered about 20 financial instruments without failure.

Disputes. The Ricardian Contract has appeared in two distinct forums of dispute resolution to resolve claims [18]. Anecdotally, each claim was resolved directly and efficiently, and without undue fuss, simply by referring to the applicable Ricardian Contract.

Automation. Relatively little has needed to be automated. In practice, fields have been inserted and standardised so that programs can extract decimalisation (dollars versus cents), labels for units (USD versus $), and titles for the issuer and the issue. In contrast to expectations, there has been no demand to parse every field.

Cost. The cost of the concept has compared favourably with that incurred with other payment systems. The preparation of the contract text carries some costs, but no more so than a user agreement. OpenPGP infrastructure requirements (keys and signing) add some minor costs to issuers but they are easily offset by the benefits of risk reduction from contract distribution. Custom signing editors have helped to reduce those costs [19].

4.3. Challenges for the Future

Layering. Layering of contracts is an impending need. Many businesses can take a standard and defined set of terms and draw on them directly. Other contracts result from earlier contracts and need to reference them.

XML. Initial efforts suggested that XML would break the rule of one contract, but it seems that we will need something better than the archaic INI format [20]. One recent proposal, the XML Voucher, stops short of presenting itself as a contract [21].

Law of Contract. The treatment of the Ricardian Contract as a *contract* may raise more legal questions than it answers. For example, is this form indeed a contract? How do distinct jurisidictions view the concept (common law, civil law, UCC, Koranic code)? Is this a negotiated or a form contract? When did the user accept the contract? How strong, or rebuttable, is the presumption that the user has the contract?

Smart Contracts. By unifying all information in a program-readable file, there is the enhanced potential of

smart contracts [22]. We have not gone further in this direction than methods to handle decimals. This is partly for lack of demand, and partly because it is not clear how a court would treat a computer program presented as a contract.

5. References

[1] Originally introduced in Ian Grigg, "Financial Cryptography in 7 Layers," *4th Conference on Financial Cryptography,* Anguilla, 2000, Springer-Verlag LNCS 1962. All papers are at http://iang.org/papers/
[2] Ian Grigg, "Digital Trading," *Virtual Finance Report*, November 1997.
[3] Country and Currency Codes, ISO3166-1.
[4] Bryce Wilcox, open design review, *DigiCash's developer list,* <ecash-dev@digicash.com>, August 1997.
[5] Ibid, Rachel Willmer, 14 August 1997.
[6] Robert Hettinga, " What's a Digital Bearer Bond?" *e$ rants* , 19th November, 1995
[7] Alex Tajirian, " David Bowie Bonds,"
[8] Ian Grigg and C. Petro, "Using Electronic Markets Achieve Efficient Task Distribution," *1st Conference on Financial Cryptography* , Anguilla, 1997, Springer-Verlag LNCS 1318.
[9] Noel Clarke, *Guide to Eurobonds*, The Economist Intelligence Unit, 1993.
[10] FDIC General Counsel's Opinion No. 8; Stored Value Cards, *Federal Register*, August 2, 1996. Also see the (readable) Press Release entitled FDIC will Continue to rely on General Counsel Opinion rather than issue rules on Stored-Value Cards, 24 June 97.
[11] Ian Grigg, Guide to Ricardian Contracts, *WebFunds project*.
[12] Jon Callas, et al, "OpenPGP Message Format," *Internet Draft*, RFC2440bis (-10 draft).
[13] Petros Maniatis, Mary Baker "Secure History Preservation through Timeline Entanglement" , *11th USENIX Security Symposium*, San Francisco, USA. August 2002.
[14] Jane K. Winn, " Couriers without Luggage" 49 *South Carolina Law Review* 739 (1998)
[15] Gary Howland, "Development of an Open and Flexible Payment System" 1996.
[16] Bryce Wilcox, "Names: Decentralized, Secure, Human-Meaningful: Choose Two", 2003
[17] Metaphor by Martin (Hasan) Bramwell. See "The Contract is the Keystone of Issuance," *Financial Cryptography blog* , 19th September 2003.
[18] *DigiGold v. Systemics*, before the Supreme Court of Anguilla (2001), and thereafter referred to the American Arbitration Association (2002).
[19] Edwin Woudt, ContractSignWizard, *WebFunds project*.
[20] Erwin van der Koogh, "Ricardian Contracts in XML," (presented at) *Edinburgh Financial Cryptography Engineering* (EFCE-2), 2001.
[21] Ko Fujimura and Masayuki Terada, XML Voucher: Generic Voucher Language, *Internet Draft*.
[22] Nick Szabo, " The Idea of Smart Contracts," 1997.