Conditional Purchase Orders

John Kelsey* Bruce Schneier†

Abstract

We propose a system of commerce based on the Conditional Purchase Order (CPO). Individual buyers issue CPOs, which are evaluated and fulfilled by sellers. There are mechanisms to bind the buyer to the transaction once the conditions are met. Additional enhancements include the ability to add anonymity, and assurances of product quality by trusted third parties.

1 Introduction

There are dozens of different buyer/seller protocols in use today. However, most of these systems focus on the seller, allowing him to price, package, or configure goods and services more effectively. Stores, catalogues, classified advertisements, telemarketing, auction houses, even on-line computerized reservations systems for airlines, are all *seller-driven*. The seller's job is to attract buyers and then close the sale.

Other commerce systems are *exchange-driven*. These systems–for example, the New York Stock Exchange–match buyers and sellers by offering an efficient, fair, and orderly marketplace. They favor neither buyers and sellers, but simply act as a location that allows for the matching process.

As commerce seeks to utilize the inherent advantages of the internet, we are seeing many forms of commerce systems being ported over to the net and many different payment systems being created [OO90, OO92, FY92, Bra93a, Bra93b, MN93, CR93, Fer94, LMP94, Oka95, NM95, BGH+95, ST95, TMSW95, 12]. Most of these approaches and systems seek to create better seller or exchange-driven systems. However, one area of electronic commerce that has not been widely explored is the ability to implement an efficient *buyer-driven* commerce system.

Buyer-driven systems represent an extremely small portion of the world's commerce. Buyers do not want to spend money on communication costs (i.e. advertising) in order to locate potential sellers. Almost all classic commerce systems impose the advertising costs on the seller. Additionally, buyers do not want to be inundated with numerous marginal or unqualified sellers, and buyers do not want to pay the inherent cost per transaction of having sellers who must customize each item they well to each individual buyer's needs.

Our aim is to create an efficient buyer-driven system of commerce. Buyers can post their requirements, much like classified advertisements, and sellers can evaluate these requirements and decide whether or not to fulfill them. Moreover, the system provides a mechanism to bind the buyer to a seller once the seller meets those requirements: a buyer cannot renege on a deal once a buyer has accepted unless the buyer fails to meet the stated requirements.

2 Conditional Purchase Orders

A *Conditional Purchase Order* (CPO) is a purchase order which (after some interaction) can be placed on a public server. Potential suppliers can then browse these CPOs, and bind any that they wish. By "binding" a CPO, a supplier effectively converts it into a signed contract.

The purpose of designing a system for CPOs is to allow small-volume buyers to effectively solicit for bids for their desired purchases, in such a way that:

1. Buyers drive the market.
2. Both buyers and sellers are protected from frivolous bids, or bids that don't meet their criteria.

This is somewhat different from the currently common model of buyer-driven commerce, in which the buyer solicits bids. With a CPO, we expect the total amount of money changing hands to be relatively small–for larger transaction amounts, we would recommend using a modified version of the protocol that collected bids. (The protocols can be altered to accept bids, though this changes the character of the system somewhat.) CPOs are somewhat like interactive classified advertisements, backed with the promise that, if someone is currently reading this ad, then it almost certainly is still available.

We need several different protocols to implement a CPO system:

1. Bob interacts with an Arbiter (Alice) and a server

*Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419; jmkelsey@counterpane.com

†Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419; schneier@counterpane.com

(Carol) to have his CPO made available to many potential sellers. This is known as *posting the CPO.*

2. Many sellers *browse the CPOs*, examining them for worthwhile offers.
3. One seller, Pam, browses Bob's offer, and decides to accept. She interacts with the the server to claim this CPO. If her claim is accepted, the CPO is removed from browsing, and Pam is able to complete the transaction. This is known as *binding the CPO.*
4. After the CPO has been bound, Pam delivers the goods to Bob, and is paid. (Bob's identity may be revealed when the CPO is bound, or Pam may have to go through an intermediary to complete the transaction.)

Some potential problems come up here:

1. It's possible for two or more sellers to try to redeem the CPO at the same time. This must be dealt with unambiguously in the protocol.
2. There must be some way to prevent bogus attempts to bind a CPO.
3. There must be some way to prevent bogus CPOs from being posted.
4. The Buyer and Seller should remain mutually anonymous (and anonymous to the Arbiter) unless there is a good reason for the system to be designed differently. The Arbiter and Server cannot be anonymous, since they have to be known and trusted by both parties. (It is anticipated that in many systems, the Arbiter and Server will be the same person or organization.)
5. All the transaction details that aren't made specifically public should be revealed only to those parties in the protocol that need to know them.
6. The buyer and seller must be qualified to carry out the transaction. This is the point of the bond certificates– if either promises something he can't deliver, his bonding agency pays the injured party a fee, and he probably can't get another bond certificate without paying a lot of money.

3 Protocols for the CPO

3.1 Players

These protocols use the following players:

1. Bob the Buyer.
2. Pam the (Potential) Seller.
3. Alice the Arbiter.
4. The Server.

3.2 Assumptions

All of these protocols assume the following:

1. A strong public-key encryption and signature scheme. Several such schemes are discussed in [Sch96].
2. A strong encryption scheme, such as 3DES [NBS77].
3. A cryptographic hash function, such as SHA1 [NIST93].
4. That everyone knows the public keys of Alice, the Server, and the Bonding Agencies. (These are all well-known parties.)
5. That Pam and Bob have "bonding certificates," as discussed below.
6. The availability of key pairs for both signing and encryption for each user. These may be the same key-pairs for signing and encryption, or different ones, depending on the application.
7. A good source of random or cryptographically strong pseudorandom numbers available to each user in the protocol.

Many of these protocols make use of the idea of a public key certificate, not as a statement of identity, but as a statement of authorization to carry out some set of operations.

3.3 Notation

Notation in the remainder of this document is as follows:

- $E_{K_A}(X)$ means that X is encrypted under key K_A, using some symmetric encryption algorithm.
- $MAC_{K_A}(X)$ means that X is authenticated under key K_A, using some symmetric message authentication algorithm.
- $PKE_{K_A}(X)$ means that X is encrypted under public key K_A.
- $SIGN_{K_A}(X)$ means that X is digitally signed under private key K_A.
- X, Y means that X is concatenated with Y.

3.4 The Structure of a CPO

This is the structure of a Candidate CPO (CCPO):

a. ID—A random 160-bit identifier.
b. ArbiterID—Unique ID of the arbiter.
c. ServerID—Unique ID of the server.
d. Goods—A text description of the goods being purchased.
e. Price—A 64-bit integer describing the price in cents.
f. StartDate—A 32-bit representation of the time and date when the CPO will become active.
g. EndDate—A 32-bit representation of the time and date when the CPO will expire.
h. Terms—A text description of all of the terms of the CPO, including those described above.
i. Bond Certificate—The Buyer's Bond Certificate.
j. Checksum—The hash of fields a–i, above.

To become a CPO, the above data must bear the signature of both the arbiter referenced in the ArbiterID field, and the server referenced in the ServerID field.

3.5 Bond Certificates

Anyone who wants to take part in this scheme must provide some guarantee that they will not back out on their commitments. This should be available from many sources, especially credit card issuers and banks. (Indeed, the simplest implementation might be a credit-card payment to an arbiter, with an agreement about under what conditions this will be partially or entirely refunded.) It is necessary to require this in order to prevent spurious CPOs and spurious bindings. We call the proof of arrangements to have these commitments paid a "bond certificate."

A "bond certificate" is a public key certificate, such that the certifier (the bonding agency) specifies a set of valid dates for the bond certificate, a limit to the amount covered, and a set of additional conditions, which may require online checking of a revocation list, may specify a Server and Arbiter to be used, etc. The private key corresponding to the public key certified is *not* known to the bonding agency—only to the user. Knowledge of that private key is used as proof of identity for the bondholder. (This allows buyer- and seller-anonymity in many cases, though of course, neither will be anonymous to the bonding agency except in very special cases.)

A bond certificate for Bob will be referred to as BC_B, while the corresponding public and private keys will be referred to as PK_B and SK_B, respectively.

3.6 Posting the CPO

A CPO is *posted* by an interaction between Bob (the buyer), Alice (the arbiter), and Carol (the server). This actually turns out to be rather complicated, especially as this part of the protocol really ought to be possible with nothing more than encrypted e-mail.

3.6.1 Getting the Arbiter's Approval

Before the CPO may be posted, the buyer must get an arbiter's approval. This is required so that both Bob and the server know that the arbiter they've designated to decide whether or not the contract has been fulfilled is actually willing to accept the CPO. (She may refuse to deal with people who have caused problems before, or to participate in the sale of items that are especially likely to cause disputes, such as used cars or some kinds of investments.) (Presumably, she is paid for doing this.) The Server will not accept a candidate CPO without an $ARBITER\_ACCEPTANCE$.

An $ARBITER\_ACCEPTANCE$ will not be issued by the arbiter unless she is convinced that Bob's candidate CPO is fresh (not a replay), and that it is guaranteed by some bonding agency. Bob must also be convinced that he is being issued a fresh $ARBITER\_ACCEPTANCE$. This works as follows:

1. Bob forms

   $U_0 = 1, 1$
   $R_0 =$ a 160-bit random number
   $ID_S =$ ID of the server.
   $X_0 = U_0, CCPO, ID_S, R_0,$ Additional Terms

   and sends to Alice

   $M_0 = PKE_{PK_A}(X_0, Sign_{SK_B}(X_0))$.

2. Alice responds with

   $U_1 = 1, 2$
   $R_1 =$ a 160-bit random number
   $X_1 = U_1, \text{hash}(X_0), R_1$

   and sends to Bob

   $M_1 = PKE_{PK_B}(X_1, Sign_{SK_A}(X_1))$.

3. Bob responds to this with

   $U_2 = 1, 3$
   $X_2 = U_2, \text{hash}(X_1)$

   and sends to Alice

   $M_2 = PKE_{PK_A}(X_2, Sign_{SK_B}(X_2))$.

4. Alice finally responds with

   $U_3 = 1, 4$
   $T_3 =$ Timestamp
   $X_3 = U_3, \text{hash}(X_2), T_3, CCPO, ID_S$

   and sends to Bob

   $M_3 = PKE_{PK_B}(X_3, Sign_{SK_A}(X_3))$.

5. Bob stores $X_3, Sign_{SK_A}(X_3)$ as the $ARBITER\_ACCEPTANCE$.

Note that after step 2, Bob knows he's talking to Alice in real-time. After step 3, Alice knows she's talking to Bob in real time. The running hash chains in all of the messages guarantee that the replay of individual messages from previous protocols is not possible. Before Alice issues the Arbiter's Acceptance in step 4, she is convinced that she's talking to a person who knows Bob's private signing key and who has a valid bonding certificate, and she's seen the terms of the CPO, as well as any other terms he may have sent her (perhaps her fee for acting as arbiter for this CPO). The final Arbiter's Acceptance is timestamped, and linked to a single server. (This prevents using a single Arbiter's Acceptance to issue many different CPOs with different servers.) It contains the CPO proposed, and so cannot be used for any other CPOs.

Note that all messages are encrypted. This is done to prevent leakage of any internal information until it is specifically published.

Note that each message has a unique 3-tupple of integers at its beginning. These work as follows: The first number denotes which system this is–arbitrarily set to 1 here. The second number denotes which protocol this is–ranging from 1 to 6. The third number denotes which message in this protocol this message is. These are used both to simplify message-processing software, and to make it impossible for an attacker to ever get an identical message from different protocols.

3.7 Variation: Getting Arbiter's Approval for Bidding

Adding bids to the CPO protocol adds a little more work for the Arbiter. This requires the Arbiter's approval, which requires only one very small change in the above protocol: The protocol ID goes from being 1 to being 5. Thus $U_0 = 5,1$, $U_1 = 5,2$, etc. This different protocol ID is enough to ensure that the Arbiter knows what she's getting herself into. The Server, seeing the different protocol ID in the arbiter's approval, will use the bidding variation of the binding protocol.

3.7.1 Posting the CPO to the Server

In order for the Server to post the CPO, it must be convinced that the CPO has a fresh $ARBITER\_ACCEPTANCE$, and that it is guaranteed by a bonding agency. This works as follows:

1. Bob forms

   $R_0 =$ random 160-bit number
   $U_0 = 2,1$
   $X_0 = U_0, R_0, ARBITER\_ACCEPTANCE$

   and then sends to the Server

   $M_0 = PKE_{PK_S}(X_0, \text{Sign}_{SK_B}(X_0))$.

2. Server receives M_0 and verifies it. If it's not stale, and if the server is willing to post the CPO, it forms

   $R_1 =$ a random 160-bit number
   $U_1 = 2,2$
   $X_1 = U_1, \text{hash}(X_0), R_1$

   and then encrypts and sends to Bob

   $M_1 = PKE_{PK_B}(X_1, \text{Sign}_{SK_S}(X_1))$.

3. Bob forms

   $U_2 = 2,3$
   $X_2 = U_2, hash(X_1)$

   and then sends to the Server

   $M_2 = PKE_{PK_S}(X_2, \text{Sign}_{SK_B}(X_2))$.

4. If this message's signature verifies properly, then the Server posts the CPO. The Server forms

   $U_3 = 2,4$
   $CPO = U_3, hash(X_2), CCPO$.

   It then sends to Bob

   $M_3 = PKE_{PK_B}(CPO, Sign_{SK_S}(CPO))$.

At the end of this protocol, Bob has a receipt to acknowledge that his CPO has been posted, and the Server is convinced that the holder of the bond certificate has just agreed to the CCPO, and has the arbiter's approval as well. After step 2, Bob knows he's talking to the Server in real-time. After step 3, the Server knows it's talking to Bob in real time.

3.8 Browsing the CPOs

Pam is a potential seller, with a bonding certificate (BC_P) of her own. Before she is allowed to browse the CPOs in real time (with the ability to bind them), she must go through a protocol. (The CPOs may be available to people who aren't browsing, but nobody is allowed to bind a CPO until they go through this protocol.) The purpose of this protocol is to prove that she is guaranteed by a bonding agency, and also to decrease the computational load on the server by establishing a secret authentication key, K_p. All of this radically decreases the computational expense of allowing Pam to browse the CPOs.

1. Pam forms

   $R_0 =$ a random 160-bit number
   $T =$ a time range
   $U_0 = 3,1$
   $X_0 = U_0, R_0, T, BC_P$

   and sends to the Server

   $M_0 = PKE_{PK_S}(X_0, \text{Sign}_{SK_P}(X_0))$.

2. The Server decides whether to grant Pam access. If so, it forms

   $R_1 =$ a random 160-bit number
   $U_1 = 3,2$
   $X_1 = U_1, \text{hash}(X_0), R_1$

   and sends to Pam,

   $M_1 = PKE_{PK_P}(X_1, \text{Sign}_{SK_S}(X_1))$.

3. Pam responds by forming

   $U_2 = 3,3$
   $X_2 = U_2, \text{hash}(X_1)$,

   and sends to the Server

   $M_2 = PKE_{PK_S}(X_2, \text{Sign}_{SK_P}(X_2))$.

4. The server verifies the signature, and then responds by forming

   $U_3 = 3,4$
   $K_p =$ a random secret key for binding CPOs.
   $T =$ a time range (from first protocol message)
   $X_3 = U_3, \text{hash}(X_2), T, K_p$

   and sends to Pam

   $M_3 = PKE_{PK_P}(X_3, \text{Sign}_{SK_S}(X_3))$.

At the end of this protocol, Pam holds the secret shared key with which she is allowed to bind a CPO, within the time limits specified in the last message. Pam and the Server are both convinced that they have interacted with one another in real-time, and the Server knows that Pam's attempts to bind CPOs are guaranteed by her bonding agency.

3.9 Binding the CPO

As Pam browses the CPOs, each is sent to her by the Server, authenticated under K_p, and including a random challenge to prevent replay attacks. When Pam wants to bind one, she forms an offer to bind the CPO, and sends it, along with the hash of the authenticated CPO, authenticated under K_p. The Server is convinced that this is a valid offer to bind the CPO, and that it's happening in real time. It responds by sending her $BOUND\_CPO$.

1. The Server forms

   $U_0 = 4, 1$
   $R_0 =$ a random 160-bit number,
   $X_0 = U_0, R_0,$ CPO description

   and sends Pam

   $M_0 = PKE_{PK_P}(X_0, \text{Auth}_{K_p}(X_0))$.

   This step is repeated for each CPO browsed.

2. Pam forms

   $U_1 = 4, 2$
   $R_1 =$ a random 160-bit number
   $X_1 = U_1, \text{hash}(X_0), R_1,$ Offer Details

   and encrypts and sends to the Server

   $M_1 = PKE_{PK_S}(X_1, Auth_{K_p}(X_1))$.

3. If the offer is acceptable to the Server, then it forms

   $U_2 = 4, 3$
   $T =$ timestamp
   $X_2 = U_2, \text{hash}(X_1), BC_P, T, CPO,$ Offer Details

   and encrypts and sends to Pam

   $M_2 = PKE_{PK_P}(X_2, \text{Sign}_{SK_S}(X_2))$.

4. Pam stores $X_2, \text{Sign}_{SK_S}(X_2)$ as $BOUND\_CPO$.

After step 2, the Server knows it's received a message from Pam in real time. After step 3, Pam knows the same thing, and immediately has an answer. Note that the symmetric operations here are meant to speed up the process of verifying whether or not a single user has just bound the CPO. This also reduces the computational load on the server significantly.

3.10 Variation: Accepting Bids

If we want to accept bids instead of allowing the first willing seller take the CPO, then the protocol is almost unchanged.

As Pam browses the CPOs, each is sent to her by the Server, authenticated under K_p, and including a random challenge to prevent replay attacks. When Pam wants to bid on one, she forms an bid, and sends it, along with the hash of the authenticated CPO, authenticated under K_p. The Server is convinced that this is a valid offer to bind the CPO, and that it's happening in real time. It responds by sending her $BID\_RECEIPT$.

1. The Server forms

   $U_0 = 6, 1$
   $R_0 =$ a random 160-bit number,
   $X_0 = U_0, R_0,$ CPO description

   and sends Pam

   $M_0 = PKE_{PK_P}(X_0, \text{Auth}_{K_p}(X_0))$.

   This step is repeated for each CPO browsed.

2. Pam forms

   $U_1 = 6, 2$
   $R_1 =$ a random 160-bit number
   $X_1 = U_1, \text{hash}(X_0), R_1,$ Bid Details

   and encrypts and sends to the Server

   $M_1 = PKE_{PK_S}(X_1, Auth_{K_p}(X_1))$.

3. If the bid is acceptable to the Server, then it forms

   $U_2 = 6, 3$
   $T =$ timestamp
   $T_{Response} =$ Time by which bid will get a response.
   $X_2 = U_2, \text{hash}(X_1), BC_P, T, T_{Response}, CPO,$
   Bid Details

   and encrypts and sends to Pam

   $M_2 = PKE_{PK_P}(X_2, \text{Sign}_{SK_S}(X_2))$.

4. Pam stores $X_2, \text{Sign}_{SK_S}(X_2)$ as $BID\_RECEIPT$.

5. When the time for accepting bids is over, the Server presents these to the Arbiter, who selects one as being the best bid. The winning bid is notified, as are all losing bids.

Note that the Arbiter is used in this variation, instead of the Buyer, to resolve which bid is best. This gives the sellers some assurance that the Buyer isn't going to cheat, either by changing his criteria halfway through the process, or by soliciting bids, and then making a bid himself, which he then can always take to avoid going through with the deal.

3.11 Delivering the CPO

The "Offer Details" field of $BOUND\_CPO$ specify how Pam and Bob are to finish the deal. In most cases, this will simply involve delivering some goods in exchange for some money, possibly in the presence of the Arbiter. In some cases, however, this will involve intermediaries, to preserve anonymity for Pam, Bob, or both. The only crucial points of this are that Pam has the $BOUND\_CPO$, and since that also includes BC_P, she can prove her identity to Bob or an intermediary with a simple challenge-response protocol.

4 Security of the CPO Protocols

4.1 Black-Box Attacks

A *Black-Box* attack is an attack on a system that can be carried out even if the system functions exactly as it should. These attacks must be dealt with externally–the CPO system we have described cannot resist them. The value in knowing about these attacks is that it lets us know the maximum strength of the system.

1. *Seller collusion*—Nothing can prevent sellers from colluding among themselves, in order to come to a higher offering price than might otherwise have been made possible. However, there are significant incentives for cheating among colluding sellers.

2. *Jurisdictional Problems*—A user in one country may post a CPO whose eventual seller is in another country, with server and arbiters in still other countries. This could raise some interesting jurisdictional problems. A CPO might be posted from a French citizen offering to buy Cuban cigars. Even if the seller is in Cuba, if the arbiter or server is in the US, there may be some legal problems with this. The solution we use is simply to give the server and arbiter each a chance to refuse to take part, once they've seen the proposed CPO.

3. *Failure to Deliver*—It is possible that a would-be seller binds the CPO, and then refuses to deliver the agreed-upon goods. This cannot be prevented, though his bonding agency will probably wind up paying the would-be buyer something for his trouble. It will be necessary to design the payoffs for this carefully, to avoid making it profitable for a buyer and seller in collusion to decide ahead of time to have the seller refuse to deliver the agreed-upon goods.

4.2 End-Run Attacks

An *End-Run* attack is an attack that bypasses the system entirely, attacking some of the system's basic assumptions.

1. *False Certifications*—If an attacker can produce sufficiently convincing false identifying documents, then they can enter into CPOs on either side in someone else's name.

2. *Hard-to-Interpret Conditions*—A buyer may put some hard-to-interpret conditions into the CPO, hoping to be able to refuse many or most sellers (perhaps collecting some kind of fee for frivolous binding of the CPO each time). This should be caught by the Arbiter, whose job it is to judge whether or not these conditions can be reasonably met. We expect that most arbiters would require use of "fill-in-the-blank" style conditions, which would minimize ambiguity.

3. *Key Compromises*—If a user has his private signing key stolen, then there is no longer any protection from having CPOs posted and bound in his name. Perhaps a partial countermeasure for this would be to require that the CPO server checks CRLs each day to determine whether this user is on one. If so, then the CPO is taken down immediately.

4. *Repudiation*—If a user has posted or bound a CPO, and now wishes he had not, he may simply claim to have had his key compromised. In some cases, it will be relatively easy to prove, i.e., by phone records, that the user himself posted or bound the CPO. In other cases, however, the only evidence will be the claim of the user. This is not generally solvable by cryptography. However, intelligent policies for handling certifications and future CPOs will help.

4.3 Direct Attacks

A *Direct* attack is an attack on the protocols and algorithms within the system itself. These attacks should generally be dealt with by repairing the system.

1. *Forged Bonding Certificates*—An attacker that could forge bonding certificates for himself could get away with all kinds of frivolous CPO posts and bindings. However, forging these requires either a successful internal attack on a recognized bonding agency, or a method to defeat the signature scheme used to sign the certificate. Even extending a previously-valid certificate to cover another few days is equivalent to defeating the signature scheme, by changing the signed data without detection.

2. *Forged Arbiter's Acceptance*—An attacker might try to get a forged Arbiter's Acceptance, to allow him to post a CPO that the arbiter would not care to try to interpret or enforce. The protocol for getting Arbiter's Approval should prevent replay of old arbiters' approvals. The digital signatures involved should prevent alteration of any fields (including the timestamp field).

3. *Forged Post of a CPO*—An attacker might try to forge a post of a CPO from someone else. The posting protocol should prevent simply replays of old CPO posts. Forging a post should thus require breaking the signature scheme, or learning the user's private signing key.

4. *Forged Binding of the CPO*—An attacker might try to forge a binding of a CPO in someone else's name. However, the protocol that allows a user to browse the CPOs, and that provides a binding key to each such user, should prevent anyone from being able to do this unless they know that user's private signing key. This does point out, however, that binding keys are important and should be stored and handled as carefully as other key material. It might also be a good idea to add a step to the browsing protocol to take a given binding key out of action once the user decides to end her session.

5. *Learning Details of a CPO in Advance*—An attacker might want to monitor some arbiter (say, one who specializes in CPOs for antique cars), in order to learn details of a proposed CPO in advance. Similarly, an attacker might want to monitor a bonding agency, or a server. Two things prevent this:

   (a) All communications about the CPO between the Buyer, Server, and Arbiter are encrypted under the respective parties' public keys.

(b) All such communications include random numbers, so an attacker cannot under almost any conceivable circumstances try comparing previous CPOs whose contents he knows with current proposed CPOs.

6. *Interference in Protocols*—Every important value in all these protocols is protected by signature or MAC. However, there is no real defense against an attacker altering bits in early protocol messages to prevent a given buyer from being able to use the CPO system. There is no cryptographic solution to this problem, but there may be ways of designing communications networks so that it is quite difficult to interrupt these transmissions.

5 Conclusions

This set of protocols describes one possible implementation of an infrastructure to support CPOs. It is important to note that the bonding agency, arbiter, and server can conceivably be the same entity. In this case, these protocols can be dramatically simplified.

6 Acknowledgments

The authors would like to thank Chris Hall, James Jorasch, Jay Walker, and the anonymous referees for their helpful comments. This work is patent pending in the United States and other countries.

References

[And93] R. Anderson, "Why Cryptosystems Fail," *Communications of the ACM,* v. 37, n. 11, Nov 1994, pp. 32-40.

[And94] R. Anderson, "Liability and Computer Security: Nine Principles," *Computer Security — ESORICS '94,* Springer-Verlag, 1994, pp. 231-245.

[And95] R. Anderson, "Robustness Principles for Public Key Protocols," *Advances in Cryptology — CRYPTO '95,* Springer-Verlag, 1995, pp. 236-247.

[BGH+95] M. Bellare, J.A. Garay, R. Hauser, A. Herzberg, H, Krawczyk, M. Steiner, G. Tsudik, and M. Waidner, "iKP - A Family of Secure Electronic Payment Protocols", *The First USENIX Workshop on Electronic Commerce,* USENIX Association, 1995, pp. 89–106.

[Bra93a] S. Brands, "Untraceable Off-line Cash in Wallets with Observers," *Advances in Cryptology—CRYPTO '93 Proceedings,* Springer-Verlag, 1994 pp. 302–318.

[Bra93b] S. Brands, "An Efficient Off-line Electronic Cash Systems Based on the Representation Problem," C.W.I. Technical Report CS-T9323, 1993.

[CR93] CitiBank and S. S. Rosen, "Electronic-Monetary System," International Publication Number WO 93/10503; May 27 1993.

[DP89] D.W. Davies and W.L. Price, *Security for Computer Networks,* Second Edition, John Wiley & Sons, 1989.

[Fer94] N. Ferguson, "Extensions of Single-Term Coins," *Advances in Cryptology—CRYPTO '93 Proceedings*, Springer-Verlag, 1994, pp. 292–301.

[FY92] M. Franklin and M. Yung, "Towards Provably Secure Efficient Electronic Cash," Columbia Univ. Dept of C.S. TR CUCS-018-92, April 24, 1992. (Also in Icalp-93, July 93, Lund Sweden, LNCS Springer-Verlag).

[LMP94] S. H. Low, N. F. Maxemchuk and S. Paul, "Anonymous Credit Cards," *The Second ACM Conference on Computer and Communications Security,* ACM Press, 1994, pp. 108–117.

[MN93] G. Medvinsky and B. C. Neuman, "Netcash: A Design for Practical Electronic Currency on the Internet," *The First ACM Conference on Computer and Communications Security,* ACM Press, 1993, pp. 102–106.

[NBS77] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard," National Bureau of Standards, U.S. Department of Commerce, Jan 1977.

[NIST93] National Institute of Standards and Technology, NIST FIPS PUB 180, "Secure Hash Standard," U.S. Department of Commerce, May 93.

[NM95] B. C. Neuman and G. Medvinsky, "Requirements for Network Payment: The NetCheque^{TM} Perspective," Compcon '95, pp. 32–36

[Oka95] T Okamoto, "An Efficient Divisible Electronic Cash Scheme," *Advances in Cryptology—CRYPTO '95 Proceedings*, Springer-Verlag, 1995, pp. 438–451.

[OO90] T. Okamoto and K. Ohta, "Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash," *Advances in Cryptology—CRYPTO '89 Proceedings*, Springer-Verlag, 1990, pp. 481–496

[OO92] T. Okamoto and K. Ohta, "Universal Electronic Cash," *Advances in Cryptology—CRYPTO '91 Proceedings*, Springer-Verlag, 1992, pp. 324–337.

[RSA93] RSA Laboratories, "Public Key Cryptography Standards #1: RSA Encryption Standard," version 1.5, 1 November 1993.

[Sch96] B. Schneier, *Applied Cryptography, 2nd Edition,* John Wiley & Sons, 1996.

[ST95] M. Sirbu and J. D. Tygar, "NetBill: An Internet Commerce System Optimized for Network Delivered Services," Compcon '95, pp. 20–25.

[SK96] B. Schneier and J. Kelsey, A Peer-to-Peer Software Metering System, in The Second USENIX Workshop on Electronic Commerce (USENIX Association, 1996), pp. 279–286.

[TMSW95] J. M. Tenenbaum, C. Medich, A. M. Schiffman, and W. T. Wong, "CommerceNet: Spontaneous Electronic Commerce on the Internet," Compcon '95, pp. 38–43