Fair Exchange with a Semi-Trusted Third Party

(extended abstract)

Matthew K. Franklin Michael K. Reiter

AT&T Laboratories—Research, Murray Hill, New Jersey, USA
{franklin,reiter}@research.att.com

Abstract

We present new protocols for two parties to exchange documents with *fairness*, i.e., such that no party can gain an advantage by quitting prematurely or otherwise misbehaving. We use a third party that is "semi-trusted", in the sense that it may misbehave on its own but will not conspire with either of the main parties. In our solutions, disruption by any one of the three parties will not allow the disrupter gain any useful new information about the documents. Our solutions are efficient and can be based on any of several cryptographic assumptions (e.g., factoring, discrete log, graph isomorphism). We also discuss the application of our techniques to electronic commerce protocols to achieve fair payment.

1 Introduction

A fair exchange protocol is a protocol by which two parties swap secrets without allowing either party to gain an advantage by quitting prematurely or otherwise misbehaving. Though already a well-studied problem, fair exchange has recently experienced a resurgence of activity due to its utility in a number of emerging applications, among them being electronic payment protocols (e.g., [Ket95, KG95, CTS95]) and certified electronic mail protocols (e.g., [BT94, ZG96, DGLW96]). In payment protocols, fair exchange can ensure that a customer receives a document from a vendor if and only if the vendor receives payment from the customer. Similarly, fair exchange can ensure that a certified electronic mail is delivered to its destination if and only if a proof of that delivery is delivered to its sender.

Early study yielded elegant but typically inefficient solutions to the fair exchange problem and the related "contract signing" problem [Blu81, Blu83, Yao86, LMR84, VV83, Cle89]. Also in this vein is recent work on "ripping" off-line electronic coins [Jak95]; this gives efficient two-party solutions to a problem related to fair purchase using off-line electronic cash, by removing the main financial incentive for cheating—but without guaranteeing fairness. Most recent, practical approaches employ one or more trusted parties to achieve fair exchange (e.g., [Ket95, KG95, BT94, CTS95, ZG96, DGLW96]). This work, however, does not consider possible misbehavior by the trusted party, and thus may enable the third party to learn the contents of documents being exchanged.

In this paper we propose a different approach to fair exchange. We use a third party that is "semi-trusted," in the sense that it may misbehave on its own but does not conspire with either of the main parties. In our solutions, disruption by any one of the three parties will not allow the disrupter (or anyone else) gain any useful new information about the documents. Since the third party need not be fully trusted, the third party could even be a *random* member of the network. This option, fair exchange by "kindness of strangers", provides a novel type of security that may be appealing for protocols on large public networks. Since misbehavior by the randomly chosen third party is unprofitable and detectable, the only drawback is that the protocol may need to be repeated. Our solutions are efficient, requiring only four messages in the case of no disruptions.

Fair exchange is not useful if a document holder can substitute an arbitrary (and worthless) document for the one that is expected. In our solutions, we assume that each document holder possesses a one-way hash of the document it desires (or an encryption of the document and a one-way hash of the key). A protocol is considered fair if the documents (or keys) that are swapped are consistent with the known hash values. In some cases, this property yields stronger guarantees than those provided by existing systems that implement forms of fair exchange, such as NetBill [CTS95]: NetBill guarantees only that a vendor that provides a worthless document can be detected after the exchange occurs, using mediation outside the scope of the protocol. Our protocol, in contrast, verifies the consistency of each document with the hash value requested by the other party, and so the exchange will succeed only with the requested documents.

Our assumption that a party knows the one-way hash of the document it desires is justified in protocols and applications in which one party is responsible for revealing the input that produces a known output, already validated as part of the protocol or application, from a one-way function. Examples include the S/KEY user authentication system [Hal94], the PayWord electronic payment scheme [RS95], on-line/off-line digital signatures [EGM96], and applications of digital timestamping [HS91]. More generally, consider a public database of tuples of the form $\langle \text{desc}_i, \text{enc}_i, f(K_i), \sigma_i \rangle$ where desc_i is a description of the contents of a data file (e.g., the title of a movie); enc_i is an encryption under a secret key K_i of the data file (e.g., a movie); $f(K_i)$ is a one-way hash of K_i; and σ_i is an authority's signature on the preceding

CCS 97, Zurich, Switzerland

information, which serves as the authority's appraisal that the decryption of enc_i using K_i will indeed produce the described item. Then any documents in the database can be exchanged fairly.

The rest of the paper is organized as follows. Models and definitions are given in Sections 2–3. We present our protocol in Section 4 and an optimized version in Section 5. We discuss some variations in Section 6, and applications to electronic payment protocols in Section 7 (and Appendix A).

2 Properties of fair exchange

For a fair exchange, we assume an initial state in which party X holds a secret key K_X and party Y initially holds a secret key K_Y. They wish to fairly exchange the two keys, with the help of a "semi-trusted" third party Z. We assume that all three parties know a one-way function f on the keyspace. We further assume that initially X knows $f(K_Y)$ and Y knows $f(K_X)$. A party is *honest* if it follows the fair exchange protocol. At the end of the fair exchange, the following will be true:

1. If all three parties are honest, then X learns K_Y and Y learns K_X.
2. If X and Z are honest, then Y learns nothing useful about K_X unless X learns K_Y.
3. If Y and Z are honest, then X learns nothing useful about K_Y unless Y learns K_X.
4. If X and Y are honest, then Z learns nothing useful about K_X or K_Y.

Throughout this paper, we assume that at most one of X, Y and Z misbehaves ("1-resilience"). The properties above require nothing otherwise, and in fact our solutions give little protection when two parties conspire against a third.

There are a number of notable omissions from the above properties. For example, it might seem that Properties 2 and 3 should require that Z learn nothing useful about K_X and K_Y, respectively, and perhaps Property 4 should require that X learns K_Y if and only if Y learns K_X. Such requirements, however, dictate that an *honest* party not learn certain things from a misbehaving party, which is difficult to enforce if the misbehaving party "conspires" with the honest party without the honest party's consent. For example, X could misbehave and send information to (an honest) Z that would enable Z to recover K_Y, or a misbehaving Z might disrupt the exchange so that (an honest) X learns K_Y without Y learning K_X. Beaver [Bea90] has pointed out similar "passive conspiracies" for more general multiparty protocols. It is debatable whether protection against passive conspiracies should be included in our basic model. Fortunately, it is easy to modify any fair exchange protocol to at least prevent an honest X from "accidentally" learning K_Y without Y learning K_X: At the end of the protocol, if X has learned K_Y, then X sends K_X to Y; and Y does similarly. This will complete any half-completed exchange, since under our assumptions X and Y must be honest when Z is misbehaving.

Other forms of disruption, unprofitable to the disruptor, are not covered by our basic model. Suppose Z disrupts the protocol so that at the end neither X learns K_Y nor Y learns K_X. Then, by assumption, X and Y are honest, so they could simply exchange K_X and K_Y by themselves. Alternatively, they could agree to restart the protocol, perhaps with a different Z. Similarly, X or Y could disrupt the protocol so that neither key was exchanged, but this is neither profitable nor preventable.

3 The one-way function

The one-way function used in our protocols must be of the form $f : G \to G$ where G is a group in which testing membership, computing the group operation and inverse, and sampling from a nearly uniform distribution are efficient. Moreover, f is required to have the additional property that there exists an efficiently computable function $F : G \times G \to G$ such that $F(x, f(y)) = f(xy)$. A few examples of proposed one-way functions with this property are listed below.

- Suppose that $G = Z_N^*$ where N is a product of two large distinct primes, and that f is defined by $f(x) = x^2 \bmod N$. Then $F(x, y) = x^2 y \bmod N$ has the property that $F(x, f(y)) = f(xy)$. f is one-way under the assumption that N is hard to factor.
- Suppose that $G = Z_p^*$ where p is a prime such that $p - 1$ has a large prime factor q, and that f is defined by $f(x) = g^x \bmod p$ where g has order q in Z_p^*. Then $F(x, y) = y^x \bmod p$ has the property that $F(x, f(y)) = f(xy)$. f is one-way if it is difficult to compute discrete logarithms with respect to g, p.
- Consider the group G of bijective functions $x : \{1, \ldots, n\} \to \{1, \ldots, n\}$ (i.e., the group of all permutations of $\{1, \ldots, n\}$) with a group operation of composition (i.e., $xy = x \circ y$). Fix a set $E \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$, and for any group member x, let $x(E) = \{\langle x(i), x(j)\rangle : \langle i, j\rangle \in E\}$ and $f(x) = x(E)$. Then, $F(x, y) = x(y(E)) = \{\langle x(y(i)), x(y(j))\rangle : \langle i, j\rangle \in E\}$ has the property that $F(x, f(y)) = f(xy)$. f is one-way under the assumption that $\langle \{1, \ldots, n\}, E\rangle$ is a "hard graph" [BC86], i.e., that it is computationally infeasible to determine an isomorphism between it and a random isomorphic copy of it.

More generally, it is possible to construct a one-way function f of the necessary form from any (not necessarily certified) one-way group action [BY90]. The first and third constructions above are reasonably efficient; e.g., the first, with a 768-bit N, is only roughly one order of magnitude slower than the Secure Hash Algorithm [NBS93], based on tests performed with the Cryptolib library [LMS93]. The construction based on discrete logarithms with a 768-bit p, however, is another two orders of magnitude slower still, i.e., roughly 1000 times slower than SHA.

Our protocols constrain X to exchange a preimage of the hash value $f(K_X)$ held by Y before the start of the protocol, and Y is similarly constrained. Without knowing anything more about the protocols, it is easy to see that X can attack the protocols if it can find a $\hat{K}_X \neq K_X$ such that $f(\hat{K}_X) = f(K_X)$; it can then simply run the protocol with $\hat{K}_X$. One defense is for f to be determined *after* K_X, e.g., as an appropriate function of the document encrypted with K_X, provided that it is difficult to determine a $\hat{K}_X$ after the fact such that $f(\hat{K}_X) = f(K_X)$ (a stronger requirement than one-wayness, but a weaker requirement than collision-freedom). This is possible because our protocols place no

requirement that the same f is used for both X and Y. Another defense is simply to require f to be collision-free, as it is in the construction based on discrete logarithms. For the construction based on factoring, only trivial collisions of the form $\{x, -x\}$ can be found, but these can be easily detected and overcome in our protocols. The construction based on graph isomorphism may not be collision-free, as a collision could be constructed from an automorphism of $<\{1,\ldots,n\}, E>$, i.e., a nontrivial permutation x such that $x(E) = E$. For simplicity, we will present our protocols using the same function f for both X and Y, and when convenient we will refer to f as being collision-free. The reader should be aware, however, that both of these assumptions can be relaxed in some cases.

4 The basic fair exchange protocol

Intuitively, our exchange protocol works by sharing K_X between Y and Z using a 2-out-of-2 verifiable secret sharing scheme [CGMA85, Fel87], and similarly sharing K_Y between X and Z. The properties of f described in Section 2 enable Z to verify that this sharing has been performed correctly (without revealing K_X or K_Y to Z) and, if so, Z sends its shares of K_X and K_Y to Y and X respectively.

Below we describe the protocol as a sequence of rounds, each consisting of multiple messages. We use the notation $X \rightarrow Y : m$ to denote that X sends m to Y. We assume that messages are private and authenticated, and that message replay and redirection attacks are countered with the usual techniques (e.g., identities of the sender and receiver in each message, time of transaction, nonces).

1. In the first round, X chooses x_1 at random (from the domain of f) and sends it to Y:

$$X \rightarrow Y \quad : \quad x_1 \tag{1}$$

Similarly, Y chooses y_1 at random at sends it to X:

$$Y \rightarrow X \quad : \quad y_1 \tag{2}$$

2. When X receives y_1, it sends:

$$X \rightarrow Z \quad : \quad f(K_X), f(K_Y), K_X x_1^{-1}, f(y_1) \tag{3}$$

Similarly, when Y receives x_1, it sends:

$$Y \rightarrow Z \quad : \quad f(K_Y), f(K_X), K_Y y_1^{-1}, f(x_1) \tag{4}$$

3. If Z receives

$$\alpha_X, \beta_X, \gamma_X, \delta_X$$

from X and

$$\alpha_Y, \beta_Y, \gamma_Y, \delta_Y$$

from Y, it verifies that

- $\alpha_X = \beta_Y = F(\gamma_X, \delta_Y)$, and
- $\alpha_Y = \beta_X = F(\gamma_Y, \delta_X)$.

If so, Z sends:

$$Z \rightarrow X \quad : \quad \gamma_Y \tag{5}$$
$$Z \rightarrow Y \quad : \quad \gamma_X \tag{6}$$

Correctness can be argued as follows. If all three parties are honest, then at the end of the protocol X can compute $K_Y = \gamma_Y y_1$, and Y can compute $K_X = \gamma_X x_1$. If X and Z are honest, then Y learns nothing useful from the execution of the protocol unless Z sends γ_X in the third round. But if this happens, then in the third round Z also sends γ_Y to X such that $f(\gamma_Y y_1) = f(K_Y)$. But then either X can compute K_Y from γ_Y, y_1 or Y has found a collision of f at $f(K_Y)$. Since the protocol is symmetric between X and Y, the case where Y and Z are honest is identical to the case where X and Z are honest. Finally, when X and Y are honest, then Z does not learn useful information about K_X or K_Y. More specifically, Z can simulate its view from $f(K_X), f(K_Y)$ alone, by choosing γ_X, γ_Y at random from the domain of f, and computing $\delta_Y = F(\gamma_X^{-1}, f(K_X)), \delta_X = F(\gamma_Y^{-1}, f(K_Y))$.

5 An optimized protocol

In this section, we describe the optimized version of our fair exchange protocol. The intuition behind the optimization is as follows. Flow (2) is eliminated by having X choose its own share y_1 of K_Y, and having X include this choice in flow (1). Flow (3) is eliminated by having X encrypt the relevant information in the public key of Z and then send it to Z through Y by including it in flow (1). In fact, it suffices to encrypt only some of this information, and to hash the rest using a cryptographically strong hash function. Flow (5) is replaced by a later flow from Y to X, although an exchange between X and Z is used as a fail-safe option if Y misbehaves.

In the description of the protocol below, we assume that the third party Z has a private key known only to itself, and that X knows the corresponding public key. We denote the encryption of m with Z's public key by $E_Z(m)$, and the decryption of m with Z's private key by $D_Z(m)$. The function h below is assumed to be a cryptographically strong hash function that is known to all parties.

1. X chooses x_1 and y_1 at random (from the domain of f), computes $x_2 = K_X x_1^{-1}$, and sends

$$X \rightarrow Y \quad : \quad x_1, y_1, E_Z(x_2), h(f(y_1)||f(K_X)||f(K_Y)||x_2)$$

2. When Y receives

$$x_1, y_1, \alpha, \beta$$

it computes $y_2 = K_Y y_1^{-1}$ and sends

$$Y \rightarrow Z \quad : \quad y_2, \alpha, \beta, f(y_1), f(x_1), f(K_X), f(K_Y)$$

3. When Z receives

$$y_2, \alpha, \beta, \gamma, \delta, \epsilon, \zeta$$

it verifies that

- $\epsilon = F(D_Z(\alpha), \delta)$
- $\zeta = F(y_2, \gamma)$
- $\beta = h(\gamma||\epsilon||\zeta||D_Z(\alpha))$

If so, Z accepts the exchange, sends

$$Z \rightarrow Y \quad : \quad D_Z(\alpha)$$

and subsequently will give y_2 to any party that can present $D_Z(\alpha)$. If Z does not accept, Z sends a message to Y reporting that it rejected.

4. If Y receives a message containing a value η such that $F(\eta, f(x_1)) = f(K_X)$, then it sends

$$Y \rightarrow X \quad : \quad y_2$$

Otherwise, Y informs X that the exchange failed.

5. If X receives from Y a value θ such that $F(\theta, f(y_1)) = f(K_Y)$, then it terminates the protocol. Otherwise, it sends x_2 to Z in an effort to retrieve y_2.

Note that there is a benefit of partial anonymity for X, since X and Z have no contact unless the exchange is disrupted.

The function h should be "like a random oracle". In particular, $h(f(y_1)||f(K_X)||f(K_Y)||x_2)$ should not leak information about x_2, or else a cheating Y might learn K_X after Step 1. If X has a signing key and Z knows the corresponding verifying key, and if we did not care about the anonymity of X, then $h(f(y_1)||f(K_X)||f(K_Y)||x_2)$ could be replaced by the signature of X on $f(y_1)||f(K_X)||f(K_Y)$. Then Step 5 would be modified so that Z gives y_2 to any party that demonstrates knowledge of the signing key of X.

The correctness of the protocol can be argued as follows. If all parties behave honestly, then K_X and K_Y are exchanged correctly. If X and Z behave honestly, then Y learns K_X only if Z sends x_2 in Step 3. Z does this only if its tests in Step 3 succeed and thus only if Z holds the missing share y_2 that is needed by the party that computed $h(f(y_1)||f(K_X)||f(K_Y)||x_2)$. Then this party will be able to present x_2 to Z to retrieve y_2 and reconstruct K_Y. If the party that computed $h(f(y_1)||f(K_X)||f(K_Y)||x_2)$ is Y itself, and Z accepts, then Y knew both x_2 and y_2 beforehand, and so receives no new information from Z. If Y and Z behave honestly, then X learns K_Y only if it receives y_2 from Y or Z. This will happen only if Z's tests in Step 3 succeed. If these tests succeed, then Z sends x_2 to Y in Step 3, and this will be the missing share of the value K_X that Y wanted. Lastly, if X and Y behave honestly, then Z does not learn useful information about K_X or K_Y, since it does not see shares x_1 or y_1.

6 Protocol variations

In this section we explore a number of variations to our protocols described in Sections 4 and 5.

6.1 Kindness of strangers

An interesting variation of the protocol of Section 5 is for Z to be chosen at random by X and/or Y. This gives a fast fair exchange between X and Y unless X or Y can collude with a random party on the network. If Z is unavailable, or uncooperative, then X and Y will discover this and can repeat the protocol. For example, X could choose Z based on a hash of elements of the first message from X to Y (e.g., all except $E_Z(x_2)$, which is dependent on Z). By including a nonce or a timestamp in the hash, it is likely that a different Z will be chosen for each repetition on a large network.

It is even possible to reward the third party when the exchange is successful. A simple method is for X and Y to each send a "tip" to Z after a successful exchange. This solution guarantees Z one or two coins when at most one of X and Y misbehave. More elaborate methods can guarantee Z a tip of constant amount if and only if the exchange is successful, and whether or not one of the parties misbehaves. For example, X can "rip" an off-line coin [Jak95] into two pieces, encrypt one of them using Z's public key, and then send both pieces (one encrypted, one not) to Y. Y will pass on to Z only the encrypted piece. Z verifies that it got a good half of a coin (else she rejects the exchange). Then either X or Y or both will send the other piece to Z at the end of a successful exchange. Other solutions can be based on a "fair transfer" of Z's tip using the techniques of this paper. We note that it seems to be difficult for X to maintain her anonymity from Z without risking Z's tip when Y misbehaves.

6.2 Unlinkability

In our protocol, the third party Z learns no useful information about the exchanged keys. However, if the same key K is exchanged more than once through the same Z, then Z can link these exchanges (since $f(K)$ is the same) and possibly determine a set of parties that are accessing the same (unknown) key. Such "linkability problems" are addressed in some electronic commerce protocols by adding random "salt" values [BGH+95]. We can address it by having X and Y blind [Cha81] the keys that are exchanged. Specifically, X can choose random values r_X, r_Y at the start of the protocol, and compute $f(r_X K_X), f(r_Y K_Y)$. X and Y can then exchange $\hat{K}_X = r_X K_X$ and $\hat{K}_Y = r_Y K_Y$ (if X includes r_X, r_Y in the first message to Y) with the same guarantees as in the original protocol (assuming f is collision-free). Since $\hat{K}_X, \hat{K}_Y$ are drawn from a nearly uniform distribution, common exchanges cannot be linked by Z.

6.3 Generalizations

As discussed in Section 4, our protocol uses a 2-out-of-2 verifiable secret sharing scheme to share K_X between Y and Z, and similarly for K_Y. Straightforward generalizations can be based on n-out-of-n secret sharing (using any of the constructions for f), and to t-out-of-n secret sharing (using the discrete log construction and a discrete log based verifiable secret sharing scheme [Ped92]). This can, for example, be used to solve variations of fair exchange by distributing the role of one or more of the parties. Other generalizations can address exchanges among multiple document holders along the lines of the problems considered by Ketchpel and Garcia-Molina [KG95].

7 Applications to electronic payment

As mentioned in Section 1, fair exchange is useful in electronic payment protocols. We discuss some of these applications in this section.

7.1 Micropayment schemes

Our fair exchange protocol can be easily integrated with payment schemes in which payment is made by the customer revealing the input that produces a known output from a one-way function, such as in PayWord [RS95] (or PayTree [JY96]). This allows for fair purchase of a digital document with a PayWord coin, or for the fair exchange of two PayWord coins. In such uses, the vendor receiving the

PayWord coin would not even require an independent "appraisal" of the hash value prior to the exchange, as the prior digital signature of the customer on that hash value (dictated by the PayWord protocol) suffices to enable the vendor to be compensated by presenting the input that produces that hash value. It remains to be seen, however, whether the overhead of our fair exchange protocol would be too excessive to meet the performance demands of a micropayment protocol like PayWord.

7.2 On-line payment schemes

Many electronic payment protocols have an on-line party that authorizes each sale: e.g., the "clearer" in iKP [BGH+95] and the "currency server" in NetCash [MN93]. In these schemes, the customer often pays for goods before knowing they will be received. At best, the customer gets a convincing "receipt" that can be used to complain when the vendor fails to deliver. However, this receipt could be useless if the vendor has disappeared.

It is often possible to incorporate a fair exchange into the payment protocol, using the on-line authority as our semi-trusted third party. In fact, only "half" of the fair exchange needs to be incorporated into the purchase protocol, so that the key of the vendor (party Y in our fair exchange) gets shared between the customer (party X) and the on-line authority (party Z); the protocol is presented in Appendix A. This can be done for many electronic payment protocols (e.g., iKP, NetCash) without increasing the number of flows among the participants. For the purchase of digital goods, this can give the customer a strong alternative to receipt-based protection. We caution that doing so does not magically decrease the trust that must be placed in the on-line authority, who is already fully trusted (in some sense) for the integrity of the purchase. However, although fully trusted for their financial duties, these on-line authorities need not be trusted with the contents of the digital goods themselves.

8 Conclusion

We have shown efficient solutions to fair exchange in the semi-trusted third-party setting, with applications to electronic payment schemes. It would be interesting to consider other models in which fairness can be achieved efficiently. It would also be interesting to find other applications for "kindness of stranger" protocols, which may be quite effective in practice on large public and semi-public networks.

References

[Bea90] D. Beaver. Security, fault tolerance, and communication complexity in distributed systems. Ph.D. Thesis, Harvard University, May 1990.

[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 (extended abstract). In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, July 1995.

[Blu81] M. Blum. Three applications of the oblivious transfer: Part I: Coin flipping by telephone; Part II: How to exchange secrets; Part III: How to send certified electronic mail. Department of EECS, University of California, Berkeley, CA, 1981.

[Blu83] M. Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems* 1:175–193, 1983.

[BC86] G. Brassard and C. Crepeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science* pages 188–195, 1986.

[BT94] A. Bahreman and J. D. Tygar. Certified electronic mail. In *Proceedings of the 1994 Internet Society Symposium on Network and Distributed System Security*, February 1994.

[BY90] G. Brassard and M. Yung. One-way group actions. In *Advances in Cryptology—CRYPTO '90 Proceedings* (Lecture Notes in Computer Science 537), pages 94–107, Springer-Verlag, 1991.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM* 24:84–88, 1981.

[Cha85] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM* 28(10), October 1985.

[CFN88] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology—CRYPTO '88 Proceedings* (Lecture Notes in Computer Science 403), pages 319–327, Springer-Verlag, 1990.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th IEEE Symposium on Foundations fo Computer Science*, pages 383–395, 1985.

[Cle89] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *Advances in Cryptology—CRYPTO '89 Proceedings* (Lecture Notes in Computer Science 435), pages 573–588, Springer-Verlag, 1990.

[CTS95] B. Cox, J. D. Tygar, and M. Sirbu. NetBill security and transaction protocol. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, July 1995.

[DGLW96] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical protocols for certified electronic mail. *Journal of Network and Systems Management* 4(3), 1996.

[EGM96] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology* 9(1):35–67, 1996.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 427–437, October 1987.

[Hal94] N. M. Haller. The S/Key™ one-time password system. In *Proceedings of the Internet Society Symposium on Network and Distributed Systems*, 1994.

[HS91] S. Haber and W. S. Stornetta. How to time-stamp a digital document. *Journal of Cryptology* 3(2):99–111, 1991.

[Jak95] M. Jakobsson. Ripping coins for a fair exchange. In *Advances in Cryptology—EUROCRYPT '95* (Lecture Notes in Computer Science 921), pages 220–230, 1995.

[JY96] C. Jutla and M. Yung. PayTree: Amortized-signature for flexible micropayments. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 213–221, 1996.

[Ket95] S. Ketchpel. Transaction protection for information buyers and sellers. In *Proceedings of the Dartmouth Institute for Advanced Graduate Studies '95*, 1995.

[KG95] S. Ketchpel and H. Garcia-Molina. Making trust explicit in distributed commerce transactions. Stanford Digital Library Project Working Paper SIDL-WP-1995-0018, October 12, 1995.

[LMS93] J. B. Lacy, D. P. Mitchell, and W. M. Schell. CryptoLib: Cryptography in software. In *Proceedings of the 4th USENIX Security Workshop*, pages 1–17, October 1993.

[LMR84] M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, pages 11–21, 1984.

[NBS93] *Secure Hash Standard.* National Bureau of Standards FIPS Publication 180, 1993.

[MN93] G. Medvinsky and C. Neuman. NetCash: A design for practical electronic currency on the Internet. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 102–106, 1993.

[Ped92] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO '91 Proceedings* (Lecture Notes in Computer Science 576), pages 129–140, Springer-Verlag, 1992.

[RS95] R. Rivest and A. Shamir. PayWord and MicroMint—Two simple micropayment schemes. Manuscript, 1995.

[VV83] U. Vazirani and V. Vazirani. Trapdoor pseudo-random number generators, with applications to protocol design. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 23–30, 1983.

[Yao86] A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[ZG96] J. Zhou and D. Gollman. A fair non-repudiation protocol. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 55–61, May 1996.

A Fair on-line purchase

In this appendix, we describe a variation of our exchange protocol for making electronic payment. To be consistent with the literature on payment protocols, we will adjust our terminology. A customer C wishes to purchase a secret key K_V initially held by a vendor V, using an electronic payment protocol with an on-line authority A. We assume that all three parties know a one-way function f on the keyspace (of the form described in Section 3), and that initially C knows $f(K_V)$. At the end of the fair purchase, in addition to the security properties required for basic electronic payment, the following will be true:

1. If all three parties are honest, then C learns K_V, and V is credited for the purchase.
2. If C and A are honest, then V will not be credited for the purchase unless C learns K_V.
3. If V and A are honest, then C learns nothing useful about K_V unless V is credited for the purchase.
4. If C and V are honest, then A learns nothing useful about K_V.

Again, we henceforth assume that at most one of C, V, and A misbehaves, as the properties above require nothing otherwise.

Our protocol requires that C be able to generate an authenticator $\sigma_C(m)$ for a message m such that on-line authority A can authenticate m as having come from C without receiving it directly from C. If C possesses a private key and A knows the corresponding public key, then $\sigma_C(m)$ could be C's digital signature on m. If A and C share a PIN that is unique to the customer, and if C possesses a public key for A, then $\sigma_C(m)$ could be the encryption of PIN$||m$ under A's public key.

The protocol operates as follows:

1. C chooses a random y (in the domain of f) and sends

$$C \rightarrow V \quad : \quad y, f(y), f(K_V), \sigma_C(f(y)||f(K_V))$$

2. When V receives

$$y, \alpha, \beta, \gamma$$

it computes $x = K_V y^{-1}$ and sends

$$V \rightarrow A \quad : \quad x, \alpha, \beta, \gamma$$

3. A verifies that
   - $\beta = F(x, \alpha)$, and
   - α and β came from C (using γ)

   If so, A sends an acceptance message to V and will subsequently give x to C upon direct request, e.g., after C identifies itself to A using its private key or PIN. Otherwise, A sends a rejection message to V.
4. V notifies C of A's decision or if V timed out on A. If A accepted, then V sends K_V to C.
5. If C does not receive K_V (i.e., a value consistent with $f(K_V)$) from V, it requests the missing share x from A, from which it can reconstruct K_V.

This can be incorporated into many electronic payment protocols without increasing the number of flows among the participants, e.g., iKP and NetCash. When incorporated, A would verify the conditions for ordinary acceptance of a purchase, in addition to the test in Step 3 above.

We now argue that this protocol meets our goals. If V misbehaves, then this will lead to rejection by A unless x is indeed the missing share of the key that the customer wants. In this case, C can claim this missing share x from A. If C misbehaves, and the purchase is rejected, then it learns no relevant information about K_V from either V (who only responds with the standard rejection of the underlying purchase protocol) or A (who will not reveal x after rejection). If C misbehaves and the purchase is accepted, then C will only learn information that it has paid for. Lastly, if A misbehaves, it will never learn anything useful about K_V, as it never receives y.