CERIAS Tech Report 2006-65
Secure Collaborative Planning, Forecasting, and Replenishment
by Mikhail J. Atallah
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Submitted to *Management Science*
manuscript

Secure Collaborative Planning, Forecasting, and Replenishment (SCPFR)

Vinayak Deshpande, Leroy B. Schwarz
Krannert School of Management, Purdue University {vinayak,lee}@mgmt.purdue.edu

Mikhail J. Atallah, Marina Blanton, Keith B. Frikken, Jiangtao Li
Department of Computer Science, Purdue University {mja,mbykova,kbf,jtli}@cs.purdue.edu

Although the benefits of information sharing between supply-chain partners are well known, many companies are averse to share their "private" information due to fear of adverse impact of information leakage. This paper uses techniques from Secure Multiparty Computation (SMC) to develop "secure protocols" for the CPFR® [†] (Collaborative Planning, Forecasting, and Replenishment) business process. The result is a process that permits supply-chain partners to capture *all* of the benefits of information-sharing and collaborative decision-making, but *without* disclosing their "private" demand-signal (e.g., promotions) and cost information to one another. In our collaborative (CPFR) scenario, the retailer and supplier engage in SMC protocols that result in: (1) a forecast that uses both the retailers and the suppliers observed demand signals to better forecast demand; and (2) prescribed order/shipment quantities based on system-wide costs and inventory levels (and on the joint forecasts) that minimize supply-chain expected cost/period. Our contributions are as follows: (1) we demonstrate that CPFR can be securely implemented without disclosing the private information of either partner; (2) we show that the CPFR business process is not incentive compatible without transfer payments and develop an incentive-compatible linear transfer-payment scheme for collaborative forecasting; (3) we demonstrate that our protocols are not only secure (i.e., privacy preserving), but that neither partner is able to make accurate inferences about the others future demand signals from the outputs of the protocols; and (4) we illustrate the benefits of secure collaboration using simulation.

Key words: privacy; security; secure computations; supply-chains; incentives; information sharing; CPFR
History: September 2005

1. Introduction

It is well known that information-sharing about inventory levels, sales, order-status, demand forecasts, production/delivery schedules, etc. can dramatically improve supply-chain performance. Lee and Whang (2000) describe several real-world examples. Of course, the reason for this improvement, isn't *information-sharing*, per se, but, rather, because shared information improves *decision-making*.

In Vendor-Managed Inventory (VMI) systems, for example, the buyer delegates inventory-replenishment decisions to its supplier, who, presumably, makes more system-oriented decisions

[†] CPFR® is a Registered Trademark of the Voluntary Interindustry Commerce Standards (VICS) Association

based on the SKU-level inventory-status information that the buyer shares (plus its own information). In other scenarios, the information of its partner is combined with each company's own information to make improved, but independent, decisions. Collaborative Planning, Forecasting and Replenishment (CPFR), for example, involves information-sharing between multiple trading partners (Retailers and Suppliers) to facilitate the planning and fulfillment of customer demand (see http://www.vics.org/committees/cpfr/ for details). Key CPFR activities include projecting consumer demand forecasts, and order and shipment forecasts (Demand and Supply Management), as well as placement of orders and delivery (Execution).

Despite its well-known benefits, however, many companies are averse to sharing their so-called "private" information. There are several reasons for this lack of information sharing in practice. Lee and Whang (2000) observe that supply-chain partners seldom share information that relates to sensitive cost data. One reason for not sharing such information is fear that a supply-chain partner will take advantage of this information by driving down future prices. Another concern is the confidentiality of the information to be shared. For example, a retailer may be unwilling to share sales-promotion information with its supplier because of fear this information will leak to a competitor. Walmart recently announced that it will no longer share its sales data with outside companies such as Information Resources, Inc and AC Nielson due to fear of information leakage (Hays 2004). Lack of trust between supply-chain partners can arise from each partner being wary of the possibility of other partners abusing shared information and reaping all the benefits of information sharing. Finally, companies may be unwilling to share information due to the fear of violating anti-trust regulations.

Secure Multi-Party Computation (SMC) provides a framework that enables supply-chain partners to make decisions that achieve system-wide goals without revealing the private information of any of the parties, and without the aid of a "trusted third party", even though the jointly-computed decisions require the information of all the parties. SMC accomplishes this through the use of so-called "protocols". An SMC *protocol* involves theoretically-secure hiding of private information (e.g., encryption), transmission, and *processing* of hidden private data. Since private information is never available in its original form (e.g., if encryption is used to hide the data, it is never decrypted), any attempt to hack or misuse private information is literally impossible. More details on SMC are provided in the literature review section and in section 4.

In this paper we develop and apply SMC protocols to the CPFR business process between a single supplier and a single retailer. Hence, the acronym "SCPFR" for *Secure* CPFR. Our primary goal is to demonstrate that collaborative forecasting and inventory planning can be conducted *without disclosing* private information of any supply-chain partners.

1.1. Overview/Summary

The model we examine is a 2-stage, serial (supplier-retailer) supply chain facing periodic stochastic retailer demand. Our model is based on that of Clark and Scarf (1960), except customer demand is non-stationary, i.e., customer demand is generated by a state-dependent linear process, where state information is split between the retailer and the supplier.

The privacy concerns of the supplier and retailer are as follows: Each has private information (e.g. promotion plans) with respect to future retail-customer demand. Each partner's private information would improve the collaborative forecast, but neither wants to disclose that information to its partner. Each participant also has private cost information (e.g., retailer's inventory-holding cost, manufacturer's production cost), which, if centralized, could lead to a "coordinated" decision, but both partners want to maintain its privacy, too. Finally, both partners have private inventory level information (e.g. on-hand inventory, backorders, in-transit inventory) which can lead to better ordering decisions if this information is shared.

Under a non-collaborative policy, each company forecasts demand and places replenishment orders based only on its own private information. Our SCPFR protocols will determine a collaborative forecast of future customer demand using both the retailer's and supplier's private information, and, based on these, the target echelon base-stock levels of both partners that will minimize supply-chain expected cost. Based on the base-stock levels, and their private inventory status, our SCPFR protocols then prescribe shipment quantities.

The contributions of this paper are as follows:

- We address privacy/security issues in CPFR. In particular, we demonstrate that lead-time demand forecasts, echelon base-stock levels, and order quantities can be computed collaboratively without disclosing the private information of either the supplier or the retailer.
- We address incentive issues for collaborative forecasting. In particular, we: 1) show that the CPFR process is *not* incentive compatible in absence of transfer payments; i.e, the retailer and supplier have incentives to inflate or deflate their forecasts; and (2) construct an incentive-compatible linear transfer payment scheme for collaborative forecasting.
- We provide an "inverse optimization" analysis of the SCPFR process, i.e., demonstrate that it is difficult/impossible for either party to determine the private information of its partner using its own private information and the shipping decisions made by the SCPFR protocols. This is an important practical consideration; i.e., why use SMC techniques if participants can infer their partners' private inputs from the output of the protocol?
- Demonstrate the benefits of SCPFR over a non-collaborative policy.

• Provide computationally efficient SMC protocols. That is, given their privacy-preserving nature, certain simple mathematical processes become very complex. We reformulate such processes to make them computationally more efficient.

The outline of this paper is as follows. Section 2 provides a review of relevant literature. Section 3 introduces the demand model, and the collaborative forecasting and inventory-planning process. Incentive issues for collaborative forecasting are also addressed in this section. Secure protocols for the collaborative forecasting and planning process are provided in section 4. Section 5 discusses practical issues that can arise in implementing SCPFR protocols and also provides a simulation study highlighting the benefits of SCPFR. Concluding remarks are provided in section 6.

2. Literature Review

Our research is based on three streams of literature: (i) The Value of Information in Managing Supply Chains, (ii) Secure Multiparty Computation, and (iii) Algorithmic Mechanism Design.

2.1. The Value of Information in Managing Supply Chains

Our supply-chain model consists of a 2-stage, serial (supplier-retailer) supply chain facing periodic stochastic retailer demand. There exists a large body of literature on multi-echelon inventory theory which we draw upon. Clark and Scarf (1960) established optimal policies for a serial multi-echelon inventory system for a finite-horizon problem with stationary demand, and showed that the optimal base-stock levels can be computed in a recursive fashion. Federgruen and Zipkin (1984) extended these results to the infinite-horizon model and also achieved further simplifications for the normal-demand case.

Traditionally, supply-chain management research has focused on centralized policies under the assumption that all the information about the system (e.g., costs capacity, inventory status) is available to a central planner. However, most real-world supply chains are managed by several decision-makers, each with their own, sometimes incompatible, objective functions, and each using her/his own private information. This has led to more recent research on the management of decentralized supply-chains, like the retailer-supplier scenario we examine. Chen (2004) provides a review on the role of information-sharing in achieving supply-chain collaboration and the consequences of failing to share (or imperfectly transmitting) it.

Several research papers have focused on showing the value of information sharing in a variety of settings. Lee et al. (1997a,b) have shown that lack of sharing information such as retailer demand can lead to a phenomenon known as the "bullwhip" effect. Examples of sharing downstream information such as inventory levels (leading to echelon-stock policies) include Chen (1998), Moinzadeh

(2002), Cachon and Fisher (2000), and Gavirneni et al. (1999). Lee et al. (2000) quantify the value of sharing demand information in a supply-chain model with a non-stationary demand process. Other examples of information sharing in supply-chains include Chen (2001) (Cost information), Chen and Yu (2001) (Lead-time information), and Deshpande and Schwarz (2005) (Capacity and Market potential information). Our model involves sharing, but not disclosing, of three kinds of information between the retailer and supplier: demand forecast signals, inventory holding and backlogging costs, and inventory level information.

For the CPFR business scenario, Aviv (2001) was the first to analyze the effect of collaborative forecasting on supply-chain performance. Aviv (2002) extends this CPFR model to auto-correlated demand processes, while Aviv (2003) provides a time-series framework for supply-chain management. Our non-stationary demand model is the same as in Aviv (2001), except that demand signal information is split between the retailer and supplier, and both parties desire privacy.

There is extensive literature on constructing incentive schemes for supply-chain collaboration (Cachon 2003). For example, Corbett (2001) considers the impact of asymmetric information on supply-chain coordination. For a two-echelon setting, Lee and Whang (1999) construct a non-linear incentive scheme to align the retailer and supplier's incentives with that of a centralized system. In a model similar to ours, but with stationary demand, Cachon and Zipkin (1999) construct a linear payment scheme to achieve system optimal base-stock levels as a Nash equilibrium solution. For a single-period model, using a Stackelberg framework, Cachon and Lariviere (2001) and Mishra et al. (2005) have shown that there exist incentives for the retailer to inflate demand forecasts.

Literature on privacy concerns in a supply-chain due to fear of information leakage is quite scarce. Li (2003) shows that fear of information leakage may prevent a retailer from sharing demand information with the supplier. Recently, Anand and Goyal (2005) have formalized the impact of information leakage on the incentives to acquire and share demand information. A related stream of literature looks at how information can be gleaned from the orders placed by retailers. Tarantola (1987) and Ahuja and Orlin (2001) coined the term "inverse optimization" to represent the inverse problem of inferring the values of the model parameters from given observed values of the observable parameters. Raghunathan (2001) and Graves (1999) show that the supplier can infer demand from the retailers orders for an AR(1) demand process. Recently, Gaur et al. (2005) have provided a full characterization of conditions under which demand information sharing is useful, for general ARMA demand processes. Raghunathan (2001) also conjectures that shared information is valuable only when it cannot be deduced from other parameters.

The major contribution of this paper to the information-sharing literature is to demonstrate that *information-disclosure isn't necessary to achieve a collaborative policy.* In particular, using the seminal model of Clark and Scarf (1960), we demonstrate that supply-chain forecasting and inventory-replenishment can be done collaboratively without disclosing the private information of either the supplier or retailer.

2.2. Secure Multiparty Computation

Cryptographic techniques, once considered an esoteric subject, have revolutionized the way we interact, particularly on the Internet. Examples include the encryption of credit-card information and the use of digital certificates and signatures. Within cryptography, the sub-area of *secure multiparty computation* is most relevant to this work: Secure multi-party protocols are a form of cooperative distributed computing that preserves the privacy of the participants' data. This general class of computations typically takes the following form between two parties (usually called Alice and Bob): "Alice" and "Bob" each have private data (say, x_A for Alice and x_B for Bob), and they want to compute $f(x_A, x_B)$ where the function f is known to both Alice and Bob, and $f(x_A, x_B)$ is efficiently computable by someone who had both x_A and x_B. However, neither Alice nor Bob is willing to disclose his/her private data to the other or to a third party. Informally speaking, a protocol that involves only Alice and Bob, is said to be secure if, at its end, Alice and Bob have learned only $f(x_A, x_B)$.

The history of the multi-party computation problem is extensive since it was introduced by Yao (1982) and extended by Goldreich et al. (1987) and others. Broadly speaking, it has been established that there exists a secure protocol to evaluate any well-defined function, no matter how complex. However, Goldwasser (1997) states that although the general secure multi-party computation problem is solvable in theory, using the solutions derived by these general results for special cases can be impractical. In other words, efficiency dictates the development of special solutions for special cases. Therefore, for efficiency reasons we might need to either transform the computation into a different form or provide a customized solution.

There is a substantial volume of work on the application of SMC techniques to auctions. For example, Franklin and Reiter (1996) describe SMC protocols that ensure that an auctioneer will be able to extract the winning bid without learning anything about the losers' bids until after the bidding period. Subsequent work (e.g. Naor et al. 1999, Jakobsson and Juels 2000, Decker et al. 2001, Elkind and Lipmaa 2004, Brandt and Sandholm 2005) has extended the research on secure auctions.

Literature on applications of SMC techniques to operations management is very scarce. Atallah et al. (2003) were the first to apply SMC to an operations management problem. They develop secure protocols for allocating the fixed capacity of a supplier among N retailers. Their allocation protocols are both incentive compatible and privacy preserving with respect to the supplier's capacity and the retailers' demand drivers. Recently, Clifton et al. (2004) examined a problem faced by independent trucking companies that have separate pickup and delivery tasks. They describe a secure protocol that finds opportunities to swap loads without revealing any information except the loads to be swapped.

Our contributions to the SMC literature is the development of computationally-efficient protocols to perform the computations required by SCPFR. We also provide an incentive compatibility and inverse optimization analysis of our protocols which traditionally has not been analyzed in SMC.

2.3. Algorithmic Mechanism Design

Mechanism design (MD) studies how private information can be elicited from independent agents by providing incentives to the participants to report their information truthfully. Conitzer and Sandholm (2002) show that the general algorithmic MD problem is NP-complete. One significant new research direction within MD is the blending of economics with the traditional distributed-computing notions of computational complexity and algorithmics Feigenbaum et al. (2002): More specifically, the distributed algorithmic mechanism design (DAMD) model considers the agents participating in a distributed computation to be acting in their own selfish best interest (as in mechanisms), while also considering computational complexity and algorithmics (as in traditional distributed computing). One of the two open problems listed in (Feigenbaum et al. 2002) asks whether easy solutions can be shown for natural problems of interest. The present paper can be viewed as a step in that direction, for some specific cases of supply-chain interactions.

Recently, the privacy of the agents within the DAMD framework has also been considered in general terms (Feigenbaum and Shenker 2002, Brandt and Sandholm 2004). We label this subfield DAMDP. Specific forms and applications of DAMDP, such as the special case of online auctions have been analyzed. Naor et al. (1999) have developed an architecture for implementing the Groves-Clarke mechanisms. Our work can be viewed as a specific form of DAMDP applied to the supply-chain setting.

3. A Model of Collaborative Forecasting and Planning

Our model is based on that of Clark and Scarf (1960): a two-stage serial (supplier-retailer) supply chain facing periodic, stochastic retailer demand over an infinite time horizon. Our assumptions

are the same as those of Clark and Scarf except that customer demand in period t, d_t, is realized from a state-dependent linear process, as described below.

The retailer observes customer demand over time and places replenishment orders on the supplier. The supplier receives orders placed by the retailer but does not observe customer demand. The retailer and the supplier observe independent "signals" about future customer demand. For example, the retailer might have private information about future promotions which can affect his forecast of demand. Similarly, the supplier may observe signals about overall market trends which can influence future demand.

In the non-collaborative scenario, retail demand is seen only by the retailer and not transmitted to the supplier, and the retailer's and supplier's future demand signals are private information and not shared with the other. As a result, the retailer forecasts demand based on his past demand observations and (only) his signals about future demand. Correspondingly, the supplier only observes the retailer's replenishment orders; and must forecast the retailer's future orders without the knowledge of past customer demand or the retailer's private signals.

In the collaborative scenario, a joint forecast is created by incorporating past observations of demand as well the retailer's and supplier's signals about future demand. Under SCPFR, these forecasts are computed without disclosing either party's private demand signals and without revealing customer demand to the supplier.

3.1. Demand Model and Forecasting Process

Customer demand follows a linear process given by the following equation:

$$d_t = \mu + \theta^r \sum_{i=1}^{T} \delta^r_{t,i} + \theta^s \sum_{i=1}^{T} \delta^s_{t,i} + \epsilon_t \tag{1}$$

Here, d_t denotes the demand realization in period t, while $\delta^j_{t,i}$ indicates the signal observed by player j (=retailer or supplier) about period t demand in period $t-i$. For example, $\delta^r_{t,i}$ may represent the impact of promotion that the retailer plans to run in period t as assessed in period $t-i$. Similarly, $\delta^s_{t,i}$ may represent the impact of new product introductions by the supplier in period t as estimated in period $t-i$. This demand model is similar to the one proposed by Aviv (2001, 2002), except that we do not capture intertemporal correlation between demands in consecutive periods or the correlation between the retailer and the supplier's signals. The key difference between our model and Aviv's is that information is split between the retailer and the supplier, and each player desires its privacy. Also, we assume that the parameters of the demand process, i.e., μ, θ^r, and θ^s are *not* known to either the supplier or the retailer and, hence, must be estimated from past observations.

As in Aviv (2001), we assume that the signals and the error term are normally distributed with known parameters, i.e., $\delta^r_{t,i} \sim N(0,\sigma^r_i)$, $\delta^s_{t,i} \sim N(0,\sigma^s_i)$, $\epsilon_t \sim N(0,\sigma_0)$.

Hence, at the beginning of time-period t, demand over periods $[i, i+L]$ (where $i \geq t, i+L-t+1 \leq T$) is normally distributed with mean and standard deviation given by:

$$\mu_{t,[i,i+L]} = (L+1)\mu + \theta^r \sum_{j=i}^{i+L} \sum_{k=j-t+1}^{T} \delta^r_{j,k} + \theta^s \sum_{j=i}^{i+L} \sum_{k=j-t+1}^{T} \delta^s_{j,k} \tag{2}$$

$$\sigma_{t,[i,i+L]} = \sqrt{(L+1)\sigma_0^2 + {\theta^r}^2 \sum_{j=i}^{i+L} \sum_{k=1}^{j-t} {\sigma^r_k}^2 + {\theta^s}^2 \sum_{j=i}^{i+L} \sum_{k=1}^{j-t} {\sigma^s_k}^2} \tag{3}$$

In each period t the retailer observes the demand, d_t, and demand signals up to T periods in the future, $\delta^r_{j,j-t}, j = t+1,\ldots,t+T$, but these observations are not known to the supplier. Similarly, in each period t, the supplier observes signals about demand up to T periods in the future, $\delta^s_{j,j-t}, j = t+1,\ldots,t+T$, but these observations are not known to the retailer.

In the collaborative forecasting scenario, the forecast is based on both the retailer's and supplier's observations. Hence the forecast is determined as follows:

1. At the beginning of each period t, estimate $\hat{\mu}$, $\hat{\theta}^r$, and $\hat{\theta}^s$ by regressing the observations d_j versus the observed signals $\delta^r_{j,i}$ and $\delta^s_{j,i}$ for all $j < t$.

2. For the forecast horizon (T periods) construct the forecast using the following equation:

$$\hat{d}_{j,t} = \hat{\mu} + \hat{\theta}^r \sum_{i=j-t+1}^{T} \delta^r_{j,i} + \hat{\theta}^s \sum_{i=j-t+1}^{T} \delta^s_{j,i}, \quad j = t,...,t+T-1 \tag{4}$$

where $\hat{d}_{j,t}$ is the forecast of the mean demand in period j, as observed at the beginning of period t. Thus, at time t the estimate of mean demand over a lead-time of L periods beginning in period $i \geq t, i+L-t+1 \leq T$ (i.e. periods $[i, i+L]$) is given by

$$\hat{\mu}_{t,[i,i+L]} = \sum_{j=i}^{i+L} \hat{d}_{j,t} = (L+1)\hat{\mu} + \hat{\theta}^r \sum_{j=i}^{i+L} \sum_{k=j-t+1}^{T} \delta^r_{j,k} + \hat{\theta}^s \sum_{j=i}^{i+L} \sum_{k=j-t+1}^{T} \delta^s_{j,k} \tag{5}$$

The estimate the standard deviation of demand over a lead-time of L periods $[i, i+L]$ is:

$$\hat{\sigma}_{t,[i,i+L]} = \sqrt{(L+1)\sigma_0^2 + {\hat{\theta}^r}^2 \sum_{j=i}^{i+L} \sum_{k=1}^{j-t} {\sigma^r_k}^2 + {\hat{\theta}^s}^2 \sum_{j=i}^{i+L} \sum_{k=1}^{j-t} {\sigma^s_k}^2} \tag{6}$$

3.2. Inventory-Planning Process

We assume the following sequence of events during a period:

1. Shipments corresponding to orders placed a lead-time earlier are received.

2. New orders are placed and shipments corresponding to these orders are released.
3. Signals about future demands are observed by the supplier and retailer, respectively.
4. Demand for the period is realized.
5. Holding and backorder costs are incurred.

The only inventory available to satisfy customer demand is the retailer's on-hand inventory at the beginning of each time period. Excess customer demand is backordered. Retailer end-of-period backorders incur a penalty cost of $\$p^R$/unit. Retailer end-of-period inventory is charged a holding cost of $\$h^R$/unit. The supplier incurs an inventory-holding cost of $\$h^S$/unit on its end-of-period inventory each period ($h^S < h^R$). In addition, the supplier incurs a backorder-penalty cost of $\$p^S$/unit on customer backorders at the retailer. At the beginning of each time period, after receiving any units delivered that time period, but before demand occurs, the retailer has the opportunity to place an order on the supplier. There is no fixed order cost. If the supplier's inventory is inadequate to fill the retailer's order entirely, the supplier will ship only a partial order, backordering the remainder until its own inventory is adequate to fill it. The supplier-to-retailer delivery leadtime is a fixed L_R periods. Like the retailer, the supplier can place an order (internally or externally) at the beginning of each period. The supplier's source of supply is infinite. The leadtime on supplier orders is a fixed L_S periods.

By definition, the retailer's echelon inventory is the same as its local inventory; while the supplier's echelon inventory equals the total supply-chain inventory; i.e., inventory at the retailer, plus inventory at the supplier, plus any inventory in transit between the supplier and the retailer. Correspondingly, from a supply-chain perspective, an echelon inventory-holding cost, $\$h^S$/unit, is charged on the supplier's end-of-period echelon inventory; in addition, end-of-period retailer inventory is charged an additional $\$(h^R - h^S)$/unit. Define $h_1 = h^R - h^S$ and $h_2 = h^S$. Customer backorders at the end of any period are charged $\$p = (p^R + p^S)$/unit.Let $p_1 = p^R, p_2 = p^S$. Also, define y^S and y^R to be the echelon base stocks of the supplier and retailer, respectively. The goal of the collaborative inventory-planning process is to determine the echelon base-stock levels in each period that minimize the total supply chain costs.

3.3. The Collaborative Inventory Planning Policy

For the stationary-demand case, Clark and Scarf (1960) prove that the optimal (y^S, y^R) can be determined sequentially: first, by finding the y^{R*} and then y^{S*}. For the non-stationary demand case, several researchers have proposed a myopic policy for an L period lead-time with non-stationary base-stock levels (e.g. Graves 1999). This myopic policy is optimal under the assumption that if the desired inventory at either the supplier or retailer is smaller than its existing inventory, then

units can be returned (by the retailer to the supplier, or by the supplier to its outside supplier) instantaneously and without penalty. This assumption is typically made in non-stationary demand models (e.g. Lee et al. 2000). Note that, if the mean of lead-time demand is much larger than its standard deviation, then one can expect the probability of returns (or negative orders) to be small. See Graves (1999) and Miyaoka and Hausman (2004) for additional justification.

Our collaborative inventory policy is the myopic policy described above. Following the convention in Chen and Zheng (1994), we use an accounting scheme that shifts costs across periods. Since the retailer's base-stock level y^R in period t affects costs incurred at the end of period $t+L_R$, all costs resulting from the y^R decision incurred at the end of period $t+L_R$ are charged in period t, assuming that the supplier can deliver this order quantity. Similarly, since the supplier's base-stock level y^S in period t affects costs incurred in period $t+L_R+L_S$ through his ability to deliver on the retailer's order placed in period $t+L_R$, all costs resulting from the y^S decision, which include the penalty for not satisfying the retailer's order in period $t+L_S$ and the echelon-2 inventory at the beginning of period $t+L_S$, are charged in period t.

We determine the optimal state-dependent base-stock levels (y^S, y^R) following Clark and Scarf (1960).

In period t, First y_t^{R*} is chosen to minimize:

$$G_t^1(y_t^R) = (h_1) \cdot E(\text{Leftovers}_{t+L_R}|y_t^R) + (h_2+p) \cdot E(\text{Backorders}_{t+L_R}|y_t^R)$$

where

$$E(\text{Leftovers}_{t+L_R}|y_t^R) = \int_0^{y_t^R} (y-x) \cdot f_{[t,t+L_R]}(x)\,\mathrm{d}x$$

$$E(\text{Backorders}_{t+L_R}|y_t^R) = \int_{y_t^R}^{\infty} (x-y) \cdot f_{[t,t+L_R]}(x)\,\mathrm{d}x$$

where $f_{[t,t+L_R]}(x)$ is the pdf of the demand distribution over periods $[t,t+L_R]$. Since demand is normally distributed, $G_t^1(y_t^R)$ reduces to:

$$G_t^1(y_t^R) = h_1\sigma_{t,[t,t+L_R]}z_1 + (h_1+h_2+p)\sigma_{t,[t,t+L_R]}LF(z_1)$$

where $z_1 = (y_t^R - \mu_{t,[t,t+L_R]})/\sigma_{t,[t,t+L_R]}$, $\mu_{t,[t,t+L_R]}$ is the mean of the demand over periods $[t,t+L_R]$ and $\sigma_{t,[t,t+L_R]}$ is the standard deviation of the demand over periods $[t,t+L_R]$ given by equations (2) and (3). $LF()$ is the standard loss function for normally-distributed demand.

Proposition 1. *The retailer's optimal base-stock level is given by:*

$$y_t^{R*} = \mu_{t,[t,t+L_R]} + z_1^* \cdot \sigma_{t,[t,t+L_R]}$$

where z_1^* *is the solution to*

$$(h_1+h_2+p)\Phi(z_1^*)-(h_2+p)=0 \tag{7}$$

where, $\Phi(\cdot)$ *is the cdf for the standard normal distribution.*

Note that, z_1^* is easily determined given (h_1+h_2+p) and (h_2+p) and a standard normal table. Also note that, since costs are stationary in our model, z_1^* has to be computed only *once*.

Second, y^{R*} is used to define an implied penalty cost, $P(y_t^S)$, on the supplier's echelon inventory, as follows:

$$\begin{aligned} P(y_t^S|y_{t+L_S}^{R*}, d_{[t,t+L_S-1]}) &= 0 && \text{if } \; y_t^S - d_{[t,t+L_S-1]} \geq y_{t+L_S}^{R*} \\ &= G_{t+L_S}^1(y_t^S - d_{[t,t+L_S-1]}) - G_{t+L_S}^1(y_{t+L_S}^{R*}) && \text{if } \; y_t^S - d_{[t,t+L_S-1]} \leq y_{t+L_S}^{R*} \end{aligned}$$

Here, $d_{[t,t+L_S-1]}$ is demand over the supplier's lead-time and is a normally-distributed random variable with mean $\mu_{t,[t,t+L_S-1]}$ and standard deviation $\sigma_{t,[t,t+L_S-1]}$, as calculated from equations (2) and (3), respectively. Note that $y_{t+L_S}^{R*}$ is the retailer's base-stock level at time $t+L_S$ and is a normally distributed random variable at time t. Hence, this penalty function is defined conditioned on the realization of $y_{t+L_S}^{R*}$ and $d_{[t,t+L_S-1]}$. Thus, $U_t = d_{[t,t+L_S-1]} + y_{t+L_S}^{R*}$ is also a normally distributed random variable with mean μ_U and standard deviation σ_U, where $\mu_U = \mu_{t,[t,t+L_S+L_R]} + z_1^*\sigma_{t+L_S,[t+L_S,t+L_S+L_R]}$ and $\sigma_U^2 = \sigma_{t,[t,t+L_S+L_R]}^2 - \sigma_{t+L_S,[t+L_S,t+L_S+L_R]}^2$. Also, define the random variable $V_t = d_{[t,t+L_S+L_R]}$. Note that V_t is also a normal random variable with a mean $\mu_V = \mu_{t,[t,t+L_S+L_R]}$ and standard deviation $\sigma_V = \sigma_{t,[t,t+L_S+L_R]}$ as given by equations (2) and (3), respectively.

Hence, the optimal echelon base stock for the supplier, y_t^{S*}, is the y^S that minimizes

$$G_t^2(y_t^S) = h_2(y_t^S - \mu_{t,[t,t+L_S-1]}) + E_{U>y_t^S}[G_{t+L_S}^1(y_t^S - d_{[t,t+L_S-1]}) - G_{t+L_S}^1(y_{t+L_S}^{R*})]$$

Since $G^1(y^R)$ is convex, $P(y^S)$ is also convex. Consequently, $G^2(y^S)$ is convex.

PROPOSITION 2. *The supplier's optimal base-stock level* y_t^{S*} *is a solution to:*

$$h_2 - (h_2+p_1+p_2)\Phi(z_2(y_t^S)) + (h_1+h_2+p_1+p_2)\Phi(z_2(y_t^S); z_3(y_t^S); \frac{-\sigma_U}{\sigma_V}) = 0 \tag{8}$$

where $\Phi(x_1; x_2; \rho)$ *denotes the cdf of the standard* bi-variate *normal distribution with correlation* ρ*,* $z_2(y_t^S) = -(y_t^S - \mu_U)/\sigma_U$*, and* $z_3(y_t^S) = (y_t^S - \mu_V)/\sigma_V$*.*

Note that, given the value of their parameters, equations (7) and (8) are easily determined using the standard normal and bi-variate normal tables. However, as we will see in section 4, these determinations are much more difficult if the process is to be secure. Section 4 will describe efficient secure protocols for solving (7) and (8).

3.4. Incentives for Collaborative Forecasting

The previous sub-sections described a collaborative forecasting and inventory-planning policy which minimizes total supply-chain expected costs. However, in a decentralized setting, such a policy may not be incentive compatible, since either the supplier or the retailer might reduce his individual cost by distorting demand signal information to influence the centralized policy. In this section we address incentives for each of the participants to truthfully participate in the collaborative forecasting process.

We begin by formulating the supplier and retailer's individual cost functions. Following the convention in Chen and Zheng (1994), we use an accounting scheme that shifts costs across periods. Since the retailer's base-stock level y^R in period t affects costs incurred in period $t+L_R$, all costs related to the y^R decision incurred in period $t+L_R$ are charged in period t. Similarly, all costs resulting from the y_t^S decision are charged in period t.

Thus, in period t, the cost G_t^1 charged due to the y_t^R decision, is split between the retailer and the supplier as follows:

$$G_t^1(y_t^R) = G_t^{1,R}(y_t^R) + G_t^{1,S}(y_t^R)$$

where $G_t^{1,R}(y_t^R)$ and $G_t^{1,S}(y_t^R)$ are the retailer and supplier's share of the cost G_t^1, respectively, as given below.

$$\begin{aligned} G_t^{1,R}(y_t^R) &= (h_1+h_2)\cdot E(\text{Leftovers}_{t+L_R}|y_t^R) + (h_1+h_2+p_1)\cdot E(\text{Backorders}_{t+L_R}|y_t^R) \\ &= (h_1+h_2)\sigma_{t,[t,t+L_R]}z_1 + (h_1+h_2+p_1)\sigma_{t,[t,t+L_R]}LF(z_1) \end{aligned} \tag{9}$$

and

$$G_t^{1,S}(y_t^R) = -h_2\sigma_{t,[t,t+L_R]}z_1 + p_2\sigma_{t,[t,t+L_R]}LF(z_1) \tag{10}$$

Similarly, G_t^2, charged due to the y_t^S decision, is split between the retailer and the supplier as follows:

$$G_t^2(y_t^S) = G_t^{2,R}(y_t^S) + G_t^{2,S}(y_t^S)$$

where $G_t^{2,R}(y_t^S)$ and $G_t^{2,S}(y_t^S)$ are the retailer and supplier's share of the cost G_t^2 respectively, as given below.

$$G_t^{2,R}(y_t^S) = E_{U>y_t^S}[G_{t+L_S}^{1,R}(y_t^S - d_{[t,t+L_S-1]}) - G_{t+L_S}^{1,R}(y_{t+L_S}^{R*})]$$

$$G_t^{2,S}(y_t^S) = h_2(y_t^S - \mu_{t,[t,t+L_S-1]}) + E_{U>y_t^S}[G_{t+L_S}^{1,S}(y_t^S - d_{[t,t+L_S-1]}) - G_{t+L_S}^{1,S}(y_{t+L_S}^{R*})]$$

Since each player cares about his individual expected cost, we next examine if the players can decrease them by inflating or deflating the forecasts. The following theorems establish the incentives for each player.

THEOREM 1. *If the retailer's base-stock level is determined by the collaborative inventory planning policy (described in section 3.3), the retailer (supplier) can lower his individual cost by deflating (inflating) the lead-time demand forecast* $\mu_{t,[t,t+L_R]}$*, while keeping the forecast* $\mu_{t,[t,t+L_R+L_S]}$ *at its true value.*

It is interesting to note that Cachon and Lariviere (2001) and Mishra et al. (2005), for a single-period model in a Stackelberg setting, have shown that the retailer has an incentive to inflate forecasts. In our multi-period, non-stationary demand setting, Theorem 1 shows that the retailer has an incentive to deflate short-term (over retail lead-time) forecast. This is because the service-level that minimizes the supply-chain costs is higher than the one that minimizes the retailer's cost. Hence, the solution that minimizes supply chain costs prescribes a higher base-stock level than what the retailer desires. As a result, if the base-stock level is set using equation (7), the retailer can lower his base-stock level by deflating the forecast over his lead-time, and hence lower his cost. The incentives for the supplier go in the opposite direction.

THEOREM 2. *If the retailer's and supplier's base-stock levels are determined by the collaborative inventory planning policy (described in section 3.3), the retailer/supplier can lower his individual cost by distorting the lead-time demand forecast* $\mu_{t,[t,t+L_R+L_S]}$*, while keeping the forecast* $\mu_{t,[t,t+L_R]}$ *at its true value.*

The retailer's incentive for reporting his long-term forecast (over supply-chain lead-time), $\mu_{t,[t,t+L_R+L_S]}$, has two components. Since the retailer does not incur inventory-holding costs on inventory leftover at the supplier (at time $t+L_S$), the retailer can improve his inventory availability from the supplier by inflating the forecast $\mu_{t,[t,t+L_R+L_S]}$, which results in an increase in the supplier's base-stock level. But, since at time $t+L_S$, the collaborative inventory policy prescribes a higher shipment from the supplier than what the retailer desires, the retailer can lower the shipment he receives from the supplier by lowering his forecast $\mu_{t,[t,t+L_R+L_S]}$. If $p_2 = 0$, then the centralized solution prescribed retailer base-stock level is close to what the retailer desires at time $t+L_S$, and, hence, the retailer will inflate the demand forecast. However, if the supplier's stock-out cost is very high, the centralized retailer base-stock level is much higher than what the retailer desires, and, hence, the retailer will deflate the demand forecast. Hence, depending on the values of h_2 and p_2, the retailer may either inflate or deflate the forecast $\mu_{t,[t,t+L_R+L_S]}$. Theorems (1) and (2) illustrate that the collaborative forecasting process is not incentive compatible. We next show that by using a transfer-payment scheme between the retailer and the supplier, the collaborative forecasting process can be made incentive compatible.

THEOREM 3. *If in each period t, a transfer payment scheme (from the retailer to the supplier) of the form $\alpha + \beta\mu_{t,[t,t+L_R]} + \gamma\mu_{t,[t,t+L_R+L_S]}$ is used, then the collaborative forecasting process is incentive compatible. Here α is any pre-negotiated constant (depending on the bargaining powers of the retailer and supplier), $\beta = -(h_2 + \frac{h_1 p_2}{h_1+h_2+p_1+p_2})$, and $\gamma = h_2(1-\Phi(z_2^*)) - p_2(\Phi(z_2^*) - \Phi(z_2^*; z_3^*; \rho))$.*

Note that this incentive-compatible transfer-payment scheme is linear in the lead-time demand forecasts. It does not use performance measures such as retailer or supplier on-hand inventory or backorders, or any other information observable by one participant but not by the other. This is important from a trust perspective, because a participant does not need to trust the other supply-chain partner to report private performance measures truthfully, in order to compute the transfer payments each period. In the next sub-section, we summarize our secure and incentive-compatible process for collaborative forecasting and planning.

3.5. Secure Process for Forecasting and Inventory Planning

We now describe the process for secure collaboration between the retailer and the supplier; i.e., a process which does not reveal the private information of either party. This is a 6-step process:

1. Retailer and supplier input their (private) cost parameters, h^R, p^R, h^S, p^S to the protocol.

2. In period t, the retailer inputs (private) information $d_{t'}$ (where $t' = 1, \ldots, t$), $\delta^r_{j,i}$ (where $j = 0, \ldots, t+T$, and $i = j-t, \ldots, T$), and also inventory status information OH^R_t (on-hand inventory) and BO^R_t (backorders). The supplier inputs his (private) information $\delta^s_{j,i}$ (where $j = 0, \ldots, t+T$ and $i = j-t, \ldots, T$) and inventory status OH^S_t, and IT^R_t (in-transit from supplier to retailer).

3. The secure forecasting protocol (described in Section 4.3) is run to compute the demand forecasts $\hat{\mu}_{t,[t,t+L_R]}$, $\hat{\mu}_{t,[t,t+L_S+L_R]}$, and the standard deviations $\hat{\sigma}_{t,[t,t+L_R]}, \sigma_U$ and σ_V. These forecasts are computed in a split fashion (see section 4) and, hence, kept private.

4. The secure inventory-planning protocol (described in Section 4.4) is run to compute the retailer and supplier's optimal base-stock levels y^{R*} and y^{S*}. The retailer's inventory position is computed as $IP^R_t = OH^R_t - BO^R_t + IT^R_t$. Also, the supplier's inventory position is computed as $IP^S_t = IP^R_t + OH^S_t + IT^S_t$. All this information is also computed in split fashion and, hence, kept private. This serves as an input to the next step.

5. The secure replenishment protocol (described in Section 4.5) is run to compute the retailer and supplier's ordering decisions. This protocol computes the order quantity $q^R_t = y^{R*}_t - IP^R_t$ and $q^S_t = y^{S*}_t - IP^S_t$. The supplier's shipment to the retailer is computed as $SH^R_t = \min\{q^R_t, OH^S_t\}$. The supplier is instructed to release this shipment to the retailer in period t. The retailer is informed of this shipment quantity in period $t + L_R$ so that he can match it with the shipment received from

the supplier. Similarly, in period t, the supplier's outside source is instructed to release a shipment of q_t^S to the supplier in period t. The supplier is informed of this shipment quantity in period $t+L_S$ so that he can match it with the shipment received from the supplier. Each player learns nothing else from the protocol. See section 5.2 for additional discussion on this.

6. The transfer payment from the supplier to the retailer is computed as $TP_t = \alpha + \beta\hat{\mu}_{t,[t,t+L_R]} + \gamma\hat{\mu}_{t,[t,t+L_R+L_S]}$. This information is released to both the supplier and the retailer.

4. Secure Protocols for Forecasting and Planning

In this section we provide a definition of our security model and the secure protocols for forecasting and inventory planning, along with some important building blocks used to achieve security.

4.1. Security Model

Informally, a protocol is defined to be "secure" provided that, upon completion, none of the parties involved know any more than they would know had the computation been performed by a hypothetical trusted third party. See Appendix A.1 for a more technical discussion on the definition of security.

4.1.1. Hiding by adding a value In our protocols we often hide a value by adding a large random value to it. That is, we hide a value x by revealing $x+r$ for some unrevealed value r. It is our goal to choose the value r such that little or no information is revealed about x by revealing $x+r$.

Conventional addition by adding a random value potentially leaks information. For example, suppose Bob has a value x that is in the range $[0,10]$ and that he hides x by adding a value r that is in the range $[0,100]$. If the realized value of $r \in [10,90]$, then $x+r \in [10,100]$, and $x+r$ reveals no information about x. On the other hand, if the realized value of $r \notin [10,90]$, then $x+r$ may reveal information about x. For example, if $x+r=0$, then $x+r$ reveals that $x=0$. Hence, this scheme reveals no information about x with an 80% probability, but may leak some information about x with a 20% probability. By increasing the the range that r is chosen from, one can reduce the probability of a leakage to a negligible level.

Another way of hiding a value is by adding a random value using *modular arithmetic*. In modular arithmetic, numbers "wrap around" upon reaching a given fixed (typically very large) quantity N. Hence, one can hide the value of x by adding a random value $r \in [0, N-1]$ to it $\bmod N$. Because all values of random r are equally likely, knowing $x+r \bmod N$ does not reveal any information about x; i.e., all values of x are equally likely.

Boolean values are typically hidden by XORing them with a random Boolean value (a Boolean variable takes the value from the set $\{0,1\}$). The Boolean operator $\oplus$, known as the XOR operator, returns 1 if exactly one of its operands is 1 and returns 0 otherwise.

4.1.2. Split values In the rest of this section, we use the following notation: Any items superscripted with (s) are known to the supplier but not to the retailer, those superscripted with (r) are known to the retailer but not to the supplier. In what follows, we often *additively split* an item x between the supplier and the retailer for the purpose of hiding it from either party.

DEFINITION 1. A value x is said to be *additively split* between players supplier and retailer if supplier has $x^{(s)}$ and retailer has $x^{(r)}$ such that $x = x^{(s)} + x^{(r)}$, but the value of x is not known to either.

Often numbers are additively split using modular arithmetic. Recall that in modular arithmetic numbers "wrap around" upon reaching a given fixed quantity. Hence, one can hide x using large random numbers $x^{(s)}$ and $x^{(r)}$ such that $(x^{(s)} + x^{(r)}) \bmod N = x$, where $x^{(s)} + x^{(r)} \geq N$ and typically $x^{(s)}$, $x^{(r)}$, and N are all significantly larger than x.

If arithmetic is modular, then this kind of additive splitting hides x in an *information-theoretic* sense from both the supplier and the retailer. This means that even if the supplier or retailer were given unlimited computational capabilities, neither of them could independently learn any information about x given its share. If, on the other hand, arithmetic is not modular, then "hiding" x is secure in a practical sense (that is, there is a negligible chance that information about x may be revealed).

In some instances, prior to engaging in a protocol, the data may already be additively split between the players. If not, then the player without input has its input share set to 0. Note that there is no danger in doing this, because all subsequent computation is secure.

Similarly, we often *XOR split* a Boolean value x between the supplier and the retailer for the purpose of hiding it from either party. This form of hiding is secure in an information-theoretic sense.

DEFINITION 2. A Boolean value x is said to be *XOR split* between players Supplier and Retailer if Supplier has a Boolean value $x^{(s)}$ and Retailer has a Boolean value $x^{(r)}$ such that $x = x^{(s)} \oplus x^{(r)}$, but the value of x is not known to either party.

In this paper we introduce protocols that produce additively or XOR split values. By this we mean that the output of the protocol will be split between the retailer and the supplier. Furthermore, to ensure that this splitting is done in a secure manner, we require that each party's output could be

simulated from that party's input alone. This requirement prevents protocols that set one party's split value of the output to 0 and the other party's split value of the output to the actual result.

Since our protocols produce additively-split or XOR-split values, the reader may wonder how the final result is obtained. If the supplier (retailer) is to learn the result, then the retailer (supplier) can send its final split value to the supplier (retailer). If both parties are to learn the result, and there is concern that one party may tamper with the result, then fair simultaneous exchange (e.g., Asokan et al. 1997, 2000) can be used.

4.1.3. Our Security Model In this section, we give protocols for *honest-but-curious* (a.k.a. *semi-honest* or *passive*) adversaries, i.e., those that will follow the protocol, but will try to learn additional information. Later in paper we show how this computation can be made resistant to deviations from the protocols through the use of incentive compatibility of SCPFR and cryptographic techniques. But in the current discussion, adversaries that try to influence the result of the protocol are outside of our threat model.

In proving the security of our protocols, we use the composition theorem of Canetti (2000) extensively. This theorem states that if a function g is computed by invoking functions $f_1, \ldots, f_n$ and is proven secure if these functions are "perfectly" implemented in a secure manner (i.e., by using a trusted third party), then a protocol that computes g by invoking secure protocols for $f_1, \ldots, f_n$, each of which is proven secure, securely computes g. A consequence of this theorem is that to prove a protocol that sequentially invokes functions $f_1, \ldots, f_n$ is secure in the semi-honest model, all that needs to be shown is that: i) the intermediate results of the protocol do not leak information and ii) the individual functions are secure. Since all of our protocols produce additively-split or XOR-split outputs, the first constraint is trivially true for all of our protocols. We formalize this notion in the following lemma:

LEMMA 1. *If a protocol* Π *invokes only functions that use additively- or XOR-split input and produce additively- or XOR-split output, and all of these functions are individually secure in the honest-but-curious (semi-honest) model and independent (i.e., they do not share randomness), then* Π *is secure in the honest-but-curious (semi-honest) model.*

PROOF. Follows directly from the composition theorem in Canetti (2000). □

4.1.4. A Note on the Efficiency of our Protocols In what follows, we describe the complexity of our protocols in terms of the amount of resources that each requires. The resources we consider are communication, computation (where we mostly pay attention to modular exponentiations due to high computational overhead of these operations), and the number of rounds that

the protocol requires. For simplicity, we consider all numbers to be representable by a constant number of bits.

4.2. Building Blocks

Next we present two basic building blocks that are used in our secure forecasting, planning, and replenishment protocols: secure split addition and secure multiplication. Other building blocks used in SCPFR are split division, secure scalar product, secure polynomial evaluation, secure matrix multiplication, secure matrix inversion, secure square root, secure comparison, and secure binary search. These building blocks are presented in Appendix C.

4.2.1. Split Addition and Subtraction Suppose x and y are additively split between the supplier and the retailer, i.e., the supplier has $x^{(s)}$ and $y^{(s)}$, and the retailer has $x^{(r)}$ and $y^{(r)}$ such that $x = x^{(s)} + x^{(r)}$ and $y = y^{(s)} + y^{(r)}$. A split addition protocol allows the supplier and the retailer to compute $x + y$ in additively-split fashion. If the supplier computes $x^{(s)} + y^{(s)}$ locally and the retailer computes $x^{(r)} + y^{(r)}$ locally, then $x + y$ is naturally split by the supplier and the retailer.

4.2.2. Public-Key Cryptography In a public-key encryption scheme, there are two keys, a public key and a private key, for encryption/decryption. The public key is used to encrypt messages, and can be publicly known. The private key is used to decrypt messages, and must be known only to the owner of the keys. This means that anyone can encrypt messages, but only the owner can decrypt them. Typically, such schemes are more computationally expensive than symmetric encryption schemes (i.e., where there is a single key for encryption/decryption). Normally, both encryption and decryption operations involve modular exponentiation, where the size of the modulus is 1024 bits or longer.

4.2.3. Homomorphic Encryption In what follows, we use E to denote a homomorphic encryption function.

Definition 3. A cryptographic scheme is said to be homomorphic if for its encryption function E the following holds: $E(x) \cdot E(y) = E(x + y)$.

Some homomorphic schemes were proposed by Damgård and Jurik (2001), Paillier (1999), and Okamoto and Uchiyama (1998). A homomorphic encryption scheme is *semantically secure* if $E(x)$ reveals no information about x. Hence $x = y$ does not imply $E(x) = E(y)$.

Homomorphic encryption schemes are public-key encryption schemes. Their homomorphic property together with key asymmetry makes them extremely useful in secure multiparty computations. This is because someone without a private decryption key can directly perform computations on the encrypted value without learning anything about that value.

Homomorphic encryption is an efficient way of performing addition, subtraction, and multiplication operations in secure manner. Assume the supplier holds x and the retailer holds y. Then they can add the numbers if the supplier computes $E(x)$ and sends it to the retailer. All the retailer needs to do is to compute $E(y)$, and then $E(x) \cdot E(y) = E(x+y)$ to obtain the encryption of $x+y$ (assuming that the retailer cannot decrypt encrypted values, e.g., obtain x from $E(x)$). Subtraction is very similar, but now the retailer computes $E(-y)$ and then $E(x) \cdot E(-y) = E(x-y)$. Multiplication can be performed if the retailer computes $E(x)^y = E(xy)$.

4.2.4. Split Multiplication If x and y are additively split between the supplier and the retailer (i.e., the supplier has $x^{(s)}$ and $y^{(s)}$, and the retailer has $x^{(r)}$ and $y^{(r)}$), then homomorphic encryption can be used to perform secure split multiplication. Let z be the desired answer to be obtained additively split as $xy = z = z^{(s)} + z^{(r)}$. Then

$$xy = x^{(s)}y^{(r)} + x^{(r)}y^{(s)} + x^{(s)}y^{(s)} + x^{(r)}y^{(r)}$$

The third and fourth terms above can be computed locally by the supplier and the retailer, respectively. The first two terms are computed by having the supplier send to the retailer both $E(x^{(s)})$ and $E(y^{(s)})$, then the retailer (who can encrypt but does not have the private decryption key) chooses a random r and computes:

$$v = E(x^{(s)})^{y^{(r)}} E(y^{(s)})^{x^{(r)}} E(-r) = E(x^{(s)}y^{(r)} + x^{(r)}y^{(s)} - r)$$

and sends v to the supplier who decrypts it and sets $z^{(s)}$ equal to:

$$z^{(s)} = D(v) + x^{(s)}y^{(s)} = x^{(s)}y^{(r)} + x^{(r)}y^{(s)} - r + x^{(s)}y^{(s)}$$

where $D(x)$ denotes decryption of x. The retailer then sets $z^{(r)}$ equal to $z^{(r)} = r + x^{(r)}y^{(r)}$. Note that $z^{(s)} + z^{(r)} = xy$, as required.

Complexity Analysis: Secure split multiplication protocol involves a constant number of rounds and requires a constant number of modular exponentiations. Communication complexity is also $O(1)$.

4.3. Secure forecasting protocol

We begin by describing the protocol for securely computing parameters $\hat{\mu}$, $\hat{\theta}^r$, and $\hat{\theta}^s$, which are used in equation (4). Then we describe a protocol for computing $\hat{\mu}_{t,[i,i+L]}$ according to equation (5), and, finally, a protocol for computing $\hat{\sigma}_{t,[i,i+L]}$ according to equation (6).

4.3.1. Parameter regression protocol Let $\vec{\theta}$ denote the vector whose transpose is $(\hat{\mu}, \hat{\theta}^r, \hat{\theta}^s)$. The $\vec{\theta}$ vector is computed so as to minimize the sum of squared differences between the predicted values and the observed values. The sum of squared differences is given by:

$$\sum_{j=t}^{t+T-1} (\hat{d}_{j,t} - \hat{\mu} - \hat{\theta}^r \sum_{i=j-t+1}^{T} \delta_{j,i}^r - \hat{\theta}^s \sum_{i=j-t+1}^{T} \delta_{j,i}^s)^2.$$

Let us denote

$$\vec{Y} = \begin{pmatrix} \hat{d}_{t,t} \\ \hat{d}_{t+1,t} \\ \vdots \\ \hat{d}_{t+T-1,t} \end{pmatrix}, \quad \text{and } X = \begin{pmatrix} 1 & \sum_{i=1}^{T} \delta_{t,i}^r & \sum_{i=1}^{T} \delta_{t,i}^s \\ 1 & \sum_{i=2}^{T} \delta_{t+1,i}^r & \sum_{i=2}^{T} \delta_{t+1,i}^s \\ \vdots & \vdots & \vdots \\ 1 & \sum_{i=T}^{T} \delta_{t+T-1,i}^r & \sum_{i=T}^{T} \delta_{t+T-1,i}^s \end{pmatrix}.$$

The equation for obtaining the updated $\vec{\theta}$ is then

$$\vec{\theta} = \begin{pmatrix} \hat{\mu} \\ \hat{\theta}^r \\ \hat{\theta}^s \end{pmatrix} = [X^T X]^{-1} X^T \vec{Y}.$$

The protocol is described in Figure 1.

Input: X is split between supplier and retailer (supplier has the third column $\{\sum_{i=1}^{T} \delta_{t,i}^s, \ldots, \sum_{i=T}^{T} \delta_{t+T-1,i}^s\}$ and retailer has the second column $\{\sum_{i=1}^{T} \delta_{t,i}^r, \ldots, \sum_{i=T}^{T} \delta_{t+T-1,i}^r\}$); $\vec{Y}$ is known to retailer only.

Output: $\vec{\theta}$ is additively split between supplier and retailer, so that $\vec{\theta} = \vec{\theta}^{(s)} + \vec{\theta}^{(r)}$.

Protocol Steps:

1. Supplier and retailer compute the 3×3 matrix $V = X^T X$ in additively split fashion, by using a split scalar product protocol 3 times (each involving two vectors of t entries).
2. Supplier and retailer compute V^{-1} in additively split fashion using the protocol described in Section C.5. This requires using the division protocol of Atallah et al. (2004) for division by the determinant (the computation of the determinant and the other entries are, as Step 1 was, split scalar product computations).
3. Supplier and retailer compute $\vec{W} = X^T \vec{Y}$ in additively split fashion, by using a split scalar product protocol 3 times (one for each row of X^T).
4. Supplier and retailer compute $\vec{\theta} = V^{-1} \vec{W}$ in additively split fashion, by using a split scalar product protocol 3 times (one for each row of V^{-1}).

Figure 1 Secure parameter regression protocol.

Complexity Analysis: Step 1 requires 3 split scalar products, step 2 (the 3×3 matrix inversion) requires 10 secure scalar products and 9 split divisions, step 3 requires 3 secure scalar products, and step 4 requires 3 secure scalar products. Therefore, the protocol needs $O(1)$ secure scalar products and $O(1)$ secure divisions.

Security: By Lemma 1, this protocol does not reveal private information as long as modular arithmetic is used, because additive splitting in that case hides information perfectly. One of the division protocols in Atallah et al. (2004) uses modular arithmetic, but some of the more practical division protocols in Atallah et al. (2004) do not use modular arithmetic.

4.3.2. Mean Demand Forecasting Protocol Figure 2 describes a protocol that securely computes the estimate for mean demand $\hat{\mu}_{t,[i,i+L]}$ at time t over a lead-time of L periods from equation (5), given additively split estimates ($\hat{\mu}$, $\hat{\theta}^r$, and $\hat{\theta}^s$).

Input: Supplier knows the $\delta^s_{j,k}$'s and retailer knows the $\delta^r_{j,k}$'s, for all j,k such that $j = i,\ldots,i+L$ and $k = j-t+1,\ldots,T$. The parameters $\hat{\mu}$, $\hat{\theta}^r$, and $\hat{\theta}^s$ are available in additively split form, i.e., for each $x \in \{\hat{\mu},\hat{\theta}^r,\hat{\theta}^s\}$ Supplier (retailer) has a random $x^{(s)}$ (resp., $x^{(r)}$) such that $x = x^{(s)} + x^{(r)}$.

Output: Supplier and retailer obtain $\hat{\mu}^{(s)}_t$ and $\hat{\mu}^{(r)}_t$, respectively, where $\hat{\mu}_{t,[i,i+L]} = \hat{\mu}^{(s)}_t + \hat{\mu}^{(r)}_t$.

Protocol Steps:

1. First, Supplier and retailer compute $v_s = \sum_{j=i}^{i+L}\sum_{k=j-t+1}^{T}\delta^s_{j,k}$ and $v_r = \sum_{j=i}^{i+L}\sum_{k=j-t+1}^{T}\delta^r_{j,k}$. To do so, supplier locally computes $v^{(s)}_s = \sum_{j=i}^{i+L}\sum_{k=j-t+1}^{T}\delta^s_{j,k}$ (since all of $\delta^s_{j,k}$'s are known to supplier only), and retailer sets $v^{(r)}_s = 0$. Similarly, retailer locally computes $v^{(r)}_r = \sum_{j=i}^{i+L}\sum_{k=j-t+1}^{T}\delta^r_{j,k}$, supplier sets $v^{(s)}_r = 0$.
2. Supplier and retailer run a split multiplication protocol twice, once to compute $w_s = \hat{\theta}^s v_s$ and once to compute $w_r = \hat{\theta}^r v_r$ (both in split fashion).
3. Supplier locally computes $\hat{\mu}^{(s)}_t = (L+1)\hat{\mu} + w^{(s)}_s + w^{(s)}_r$, and retailer computes $\hat{\mu}^{(r)}_t = w^{(r)}_s + w^{(r)}_r$.

Figure 2 Secure mean demand forecasting protocol.

Complexity Analysis: The complexity of the secure demand forecasting protocol is dominated by the 2 executions of a secure multiplication protocol (in step 2).

Security: Correctness of the answer produced by the protocol follows from equations (4) and (5), which it directly implements. As long as the split multiplication protocol and the split addition protocol are secure, by Lemma 1, the secure demand forecasting protocol is secure.

4.3.3. Standard Deviation of Demand Forecasting Protocol Figure 7 (see Appendix D.1) gives a protocol that securely computes the standard deviation of demand $\hat{\sigma}_{t,[i,i+L]}$ over a lead-time of L periods $[i,i+L]$ according to equation (6).

Complexity Analysis: The main overhead of the protocol is associated with a constant number of invocation of the multiplication protocol and a single invocation of the square root protocol.

Security: Correctness of the protocol follows from the fact the it directly performs computations

required to compute the standard deviation of demand given by Equation (6). Security of this protocol follows from Lemma 1, as long as modular arithmetics is used for addition, and multiplication and square root protocols are secure.

4.4. Secure Inventory Planning Protocol

In this subsection, we first give a protocol for computing y_t^{R*}, then a protocol for computing y_t^{S*}.

4.4.1. Secure Protocol for Computing y_t^{R*} The goal is to find an optimal collaborative base-stock for the retailer, y_t^{R*}, which can be expressed as: $y_t^{R*} = \mu_{t,[t,t+L_R]} + z_1^* \cdot \sigma_{t,[t,t+L_R]}$, where z_1^* is found by solving equation (7): $(h_1+h_2+p)\Phi(z_1^*) - (h_2+p) = 0$

Since z_1^* needs to be computed only *once*, computing y_t^{R*} securely in each period t is straightforward by using the SMC primitives previously described. Hence, we focus on determining z_1^* securely in a split fashion.

Since the standard normal cdf Φ is well tabulated, one could determine y^{R*} as follows. The supplier and the retailer first compute: $x_1 = \frac{h_2+p}{h_1+h_2+p}$ in split fashion using split division, and then compute $z_1^* = \Phi^{-1}(x_1)$ by looking up the Φ^{-1} table in a secure fashion. Let T be the table that represent Φ^{-1}. Table T contains m tuples $\langle a_i, \Phi^{-1}(a_i)\rangle$, for $i = 1, \ldots, m$, where m is presumably a large integer so that T gives a good estimate of $\Phi^{-1}(\cdot)$ with enough precisions. Suppose the supplier and the retailer have x_1 split between them; they can jointly lookup the table T and obtain the value of $\Phi^{-1}(a)$ in split fashion where a is the closest to x_1 in the table. The value $\Phi^{-1}(a)$ can be used as an estimate of z_1^*. This, however, is hugely inefficient because of the necessity to do secure "table lookups" which takes m modular exponentiations. This approach requires one secure division and one secure table lookup, thus overall it requires $O(m)$ modular exponentiations, where m is the size of the table. We next propose an efficient solution that avoids the table lookup operation.

Our solution to securely computing $\Phi(z)$ is to use an approximation based on Abramowitz and Stegun (1972):

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n! 2^n (2n+1)} \approx \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N} \frac{(-1)^n x^{2n+1}}{n! 2^n (2n+1)}$$

where N is a large integer. The protocol for computing $\Phi(z)$ is given in Appendix D.2.

Given a procedure for computing $\Phi(z)$, what now remains is the search for z_1^* to satisfy equation (7). Let $f(z) = (h_1+h_2+p)\Phi(z) - (h_2+p)$. Our goal is to find z_1^* such that $f(z_1^*) = 0$. Since $f(z)$ is a monotone function, one can determine z_1^* by the binary search protocol described in Figure 6 (see Appendix C.8). As the evaluation of $f(z)$ takes $O(n)$ modular exponentiations, the overall binary search that computes z_1^* needs $O(k \cdot (\ell + n))$ modular exponentiations.

Compared to the table lookup approach which takes $O(m)$ modular exponentiations, this binary search approach is better in practice because the size of the table is huge, i.e., $m > k \cdot (\ell + N)$.

4.4.2. Secure Protocol for Computing y_t^{S*} To compute y_t^{S*}, one must solve

$$h_2 - (h_2 + p_1 + p_2)\Phi(z_2(y_t^S)) + (h_1 + h_2 + p_1 + p_2)\Phi(z_2(y_t^S); z_3(y_t^S); \frac{-\sigma_U}{\sigma_V}) = 0$$

where $z_2(y_t^S) = -(y_t^S - \mu_U)/\sigma_U$ and $z_3(y_t^S) = (y_t^S - \mu_V)/\sigma_V$. Hence $z_3 = \frac{\mu_U - \mu_V - z_2\sigma_U}{\sigma_V}$. To compute y_t^{S*}, we can first find z_2 such that

$$h_2 - (h_2 + p_1 + p_2)\Phi(z_2) + (h_1 + h_2 + p_1 + p_2)\Phi(z_2; \frac{\mu_U - \mu_V - z_2\sigma_U}{\sigma_V}; \frac{-\sigma_U}{\sigma_V}) = 0,$$

then we compute $y_t^{S*} = \mu_U - z_2\sigma_U$. We determine z_2 using the same technique as above, i.e., using approximation and binary search.

As above, we compute $\Phi(z_2; \frac{\mu_U - \mu_V - z_2\sigma_U}{\sigma_V}; \frac{-\sigma_U}{\sigma_V})$ in split fashion using approximations based on Abramowitz and Stegun (1972):

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \approx \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N} \frac{(-1)^n x^{2n}}{n! 2^n}$$

$$\Phi(x; y; \rho) \approx \Phi(x)\Phi(y) + \phi(x)\phi(y) \sum_{n=0}^{N} \frac{1}{(n+1)!} \mathrm{He}_n(x)\mathrm{He}_n(y)\rho^{n+1}$$

where

$$\mathrm{He}_n(x) = (-1)^n \frac{\phi^{(n)}(x)}{\phi(x)} = n! \sum_{m=0}^{[\frac{n}{2}]} \frac{(-1)^m}{2^m m!(n-2m)!} x^{n-2m}$$

are the Hermite polynomials. The protocol for computing $\Phi(x, y, \rho)$ is given in Appendix D.3. Let $f(z_2) = h_2 - (h_2 + p_1 + p_2)\Phi(z_2) + (h_1 + h_2 + p_1 + p_2)\Phi(z_2; \frac{\mu_U - \mu_V - z_2\sigma_U}{\sigma_V}; \frac{-\sigma_U}{\sigma_V})$. Our goal is to find z_2^* such that $f(z_2^*) = 0$. One can determine z_2^* by the binary search protocol described in Figure 6 (see Appendix C.8).

4.5. Secure Replenishment Protocol

Once the supplier and the retailer have y^{R*}, y^{S*}, IP_R, and IP_S computed in split fashion, computation of the order quantity is easy, and is shown in Appendix D.4. In the protocol, we use a cryptographic building block called *commitment scheme* (e.g., Cramer and Damgård 1998, Fujisaki and Okamoto 1997). In a commitment scheme, a prover chooses a secret s, computes the corresponding commitment c, and reveals c to a verifier with the following two properties: (1) the verifier cannot compute anything about s from the commitment c, and (2) the prover cannot change his committed value once it is committed, but he can open the commitment to reveal s and convince the verifier that s is indeed the value committed under c.

4.6. Secure Transfer Payment Protocol

Before describing a protocol for computing the transfer payment, we first give protocols for computing the parameters β and γ. Note that the value of β is fixed and, therefore, its computation must be performed only once. The value of γ, on the other hand, must be re-computed every period.

Secure Protocol for Computing β:

From Theorem 3, $\beta = -(h_2 + \frac{h_1 p_2}{h_1+h_2+p_1+p2}) = -h^S - \frac{(h^R-h^S)p^R}{h^R+p^R+p^S}$. Using this equation, a secure protocol for β is given in Appendix D.5.

Secure Protocol for Computing γ:

From Theorem 3, γ is computed as $\gamma = h_2(1-\Phi(z_2^*)) - p_2(\Phi(z_2^*) - \Phi(z_2^*; z_3^*; \rho))$. Since the value of $z_2^* = z_2(y_t^{S*})$ and z_3^* is computed during step 4 of the SCPFR process (see Section 3.5), we can directly use that value without re-calculating it. Furthermore, to minimize the possibility of cheating by dishonest players, this step (i.e., computation of transfer payment) should be conducted in parallel with computation of y_t^{S*} in step 4 of the SCPFR process (see Section 5.2 for further discussion). Given z_2^*, secure computation of γ becomes rather simple; our protocol for γ is given in see Appendix D.6.

Overall Protocol for Transfer Payment:

We use sub-protocols for computing β and γ to securely calculate the transfer payment. The protocol is given in Appendix D.7. Note that the values of $\hat{\mu}_{t,[t,t+L_R]}$ and $\hat{\mu}_{t,[t,t+L_S+L_R]}$ are computed as a part of step 3 of the SCPFR process (see Section 3.5) and thus are available in additively split form.

4.7. Alternative Boolean Circuit Approach

In this section, we briefly describe an alternative approach for secure computation. This alternative is based on the idea that one can securely compute a function by securely simulating a Boolean circuit for that function.

Suppose we have constructed a logical circuit C for computing some function f (note that we are not describing a physical circuit). As before, assume that the participants of the protocol will be honest-but-curious adversaries. It is possible to securely evaluate the circuit C with communication proportional to the size of C. This scheme also requires an oblivious transfer operation for each input wire (which requires some form of public key cryptography), and for each gate of the circuit it requires that a more efficient cryptographic function (e.g., a secure encryption scheme, such as AES) be evaluated a constant number of times. This circuit simulation is possible in a constant number of rounds using the technique identified by Yao (1986).

In summary, if one has a circuit C for evaluating a function, then one can securely evaluate the function with the same efficiency as evaluating C insecurely within a constant multiplicative factor (albeit a very large constant). Furthermore, it is possible to build such a circuit for each of the above-mentioned computations. We discuss how this is done in more detail in Appendix D.

The primary advantage of such an approach is that the rounds required to compute the function is constant, whereas many of the protocols outlined above require $O(k)$ rounds where k is the required precision of the answer. The disadvantage of is that operations such as multiplication are much more expensive in terms of communication than the previous approach (i.e., the common circuit for multiplying two k-bit numbers requires $O(k^2)$ gates). Thus, it is not clear to us which of these approaches will be more efficient, and we conjecture that this depends upon the environment in which the protocols are used.

5. Implementation Considerations and Effectiveness of CPFR

Secure CPFR (SCPFR) is designed to support the *Demand & Supply Management* activity of CPFR, which involves collaborative customer-demand forecasting and order planning. Specifically, within the context of our supplier-retailer supply chain, SCPFR: (1) updates the estimates of μ, θ^r and θ^s, using the private demand signals of both parties; (2) forecasts customer demand for future periods using these updated estimates; (3) computes the optimal collaborative base stocks, and (4) prescribes shipment quantities, based on the base-stock levels and each party's private inventory position. In addition, in order to assure incentive compatibility, SCPFR assesses transfer payments as contracted for in CPFR's Strategy-and-Planning activity.

5.1. Improved performance: Simulation Study

We address the following questions: How much benefit can the supply chain expect by implementing SCPFR? How much does each party (retailer and supplier) benefit due to SCPFR? To answer these questions, we compute costs/period for two policies: (i) SCPFR policy, described in sections 3 and 4, which minimizes total supply-chain costs, and (ii) a non-collaborative policy, where the retailers and suppliers do not share information, and, as a result, focus on minimizing individual costs based on their own information.

In the non-collaborative policy, the retailer constructs lead-time demand forecasts in the following fashion:

$$\mu^R_{t,[i,i+L]} = (L+1)\mu + \theta^r \sum_{j=i}^{i+L} \sum_{k=j-t+1}^{T} \delta^r_{j,k} \tag{11}$$

Note that (11) contains only signals observed by the retailer and not the supplier. Thus, at the beginning of each period t, the retailer estimates $\hat{\mu}$, and $\hat{\theta}^r$ by regressing its observed d_j values

on its observed signals $\delta^r_{j,i}$ for all $j < t$. Then, using these estimates, he constructs a lead-time demand forecast from equation (11). The retailer uses a myopic state-dependent base-stock policy, for deciding his order quantity, given by the following equation:

$$y^R_t = \mu^R_{t,[t,t+L_R]} + z^R_1 \cdot \sigma^R_{t,[t,t+L_R]} \tag{12}$$

Where $z^R_1 = \Phi^{-1}(h^R/(h^R + p^R))$ and $\sigma^R_{t,[t,t+L_R]}$ is his estimate of the standard deviation of the unobservable (error) terms of demand. The supplier observes the orders placed by the retailer, q^R_t, but does not observe customer demand or the retailer's signals. Hence, the supplier regresses the retailer's orders, q^R_t, on his demand signals δ^s to obtain estimates of μ and θ^s. The supplier uses the following equations to estimate his demand forecast and set his (installation) base-stock level.

$$\mu^S_{t,[i,i+L]} = (L+1)\mu + \theta^s \sum_{j=i}^{i+L} \sum_{k=j-t+1}^{T} \delta^s_{j,k} \tag{13}$$

$$y^S_t = \mu^S_{t,[t,t+L_S]} + z^S_2 \cdot \sigma^S_{t,[t,t+L_S]} \tag{14}$$

Where $z^S_2 = \Phi^{-1}(h^S/(h^S + p^S))$ and $\sigma^S_{t,[t,t+L_S]}$ be his estimate of the standard deviation of the unobservable (error) terms of retailer orders.

We compared the costs of the two policies for the following set of parameters: $\mu = 15, \theta^r = \theta_s = 1, L_R = L_S = 2, T = 5, \sigma^2_0 = 1, \sigma^{r2}_i = \sigma^{s2}_i = \sigma^2_i; \sigma^2_i = \sigma^2, 0.8\sigma^2, 0.6\sigma^2, 0.4\sigma^2, 0.2\sigma^2; i = 1,2,3,4,5; h^R = 1, h^S = 0.5, p^R = 12, p^S = 7$. These parameters are similar to those in Aviv (2002).

Each policy was simulated for 10,000 periods using the same customer demand. In the simulation, negative orders were not allowed, i.e., if the policy computed a negative order, an order of size zero was used. We observed that the SCPFR policy recommended negative orders in only 10 of the 10,000 periods (0.1%). We report the cost/period for each policy. Figure 3 plots the absolute and percentage benefits to the supply chain, the retailer, and the supplier as a function of σ. Note that, the benefit of secure CPFR over a non collaborative setting can be significant, ranging from 26% to 38%. These benefits are not equally split. The supplier gets most of the absolute cost savings when σ is low, whereas the retailers and suppliers savings are comparable at higher values of σ. The costs reported do not include the transfer payments derived in section 3.4. By adjusting the value of the prenegotiated constant α of the transfer payment scheme, an equitable distribution of the savings between the retailer and the supplier can be easily achieved.

Figure 4 displays the cost savings as a function of the split of the end-customer backorder costs between the supplier and the retailer. The graph shows that if most of the end-customer backorder costs accrue to the retailer, then the retailer gains the most by engaging in secure SCPFR. Note

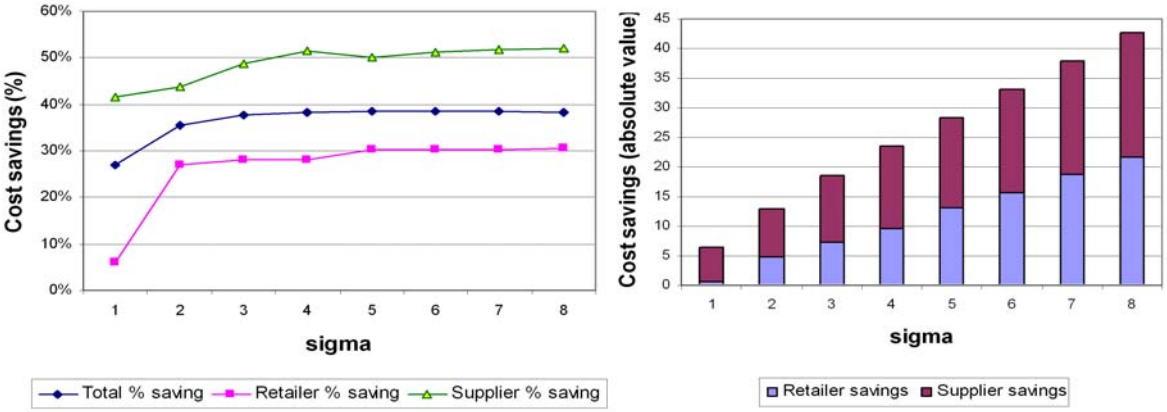

Figure 3 **Percentage and Absolute Cost Savings vs σ**

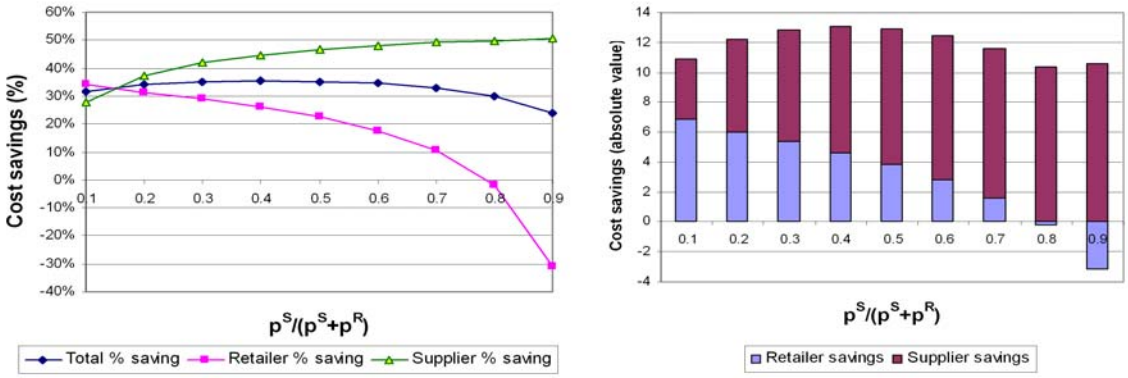

Figure 4 **Percentage and Absolute Cost Savings vs $p^S/(p^S+p^R)$**

that, although the supply chain benefits from secure CPFR under all scenarios, the retailer can actually be worse off if most of the end-customer backorder costs are attributed to the supplier. In this case, the retailer would like to stock very little inventory (due to his low backorder cost), but the collaborative policy forces him to stock much more. In this case, the transfer payments from the supplier to the retailer (α) are needed to ensure the retailer's participation.

5.2. Inverse Optimization and Practicality of protocols

As defined in section 4, a computation is said to be "secure" provided that, upon completion, none of the parties involved know anything more than the output provided by the protocol. It is important to note that, even in those circumstances when one or more parties involved could deduce the private inputs of another from the output of the protocol, such a protocol would be defined to be "secure". As a trivial example, note that if $f(x_A, x_B) = x_A + x_B$, both Alice and Bob could deduce the private input of the other simply by subtracting their private input from the sum provided by the secure protocol.

Hence, the issue of "inverse optimization" – which is not of concern within the area of SMC – is an important one when secure protocols are deployed in practice. Inverse optimization asks: Can the parties involved infer the input of other parties from the output of the protocol and their own private information? Specifically, for our SCPFR setting: Can the retailer or the supplier deduce its partners future demand signals from the output of the protocol? This issue is related to the *invertibility* and *inferability* of demand. Gaur et al. (2005) addressed the question of whether the supplier can infer the retailer's demand by observing the retailer's orders, under the assumption that the supplier knows the parameters of the demand process. In addition, we address the question of whether the retailer can infer the supplier's future demand signals from shipment information released by the protocol, assuming that the supply-chain partners do not know the parameters of the demand process.

We first summarize what information is available to each player in each step of the SCPFR process described in section 3.5. In step 1, the retailer and supplier provide their respective cost inputs, h and p, and these are kept private by the protocol. In step 2, for each period t, the retailer and supplier provide their (private) observations of future demand signals $\delta_{j,i}; t+T > j > t$ and their private inventory level information. Both of these are kept private by the protocol. In step 3, the mean and standard deviation of the demand are computed in a split fashion (and hence kept private). In steps 4 and 5, the base-stock levels and order quantities are also computed in a split fashion and are kept private. Finally, in step 5, the shipment quantities are computed. The period t's shipment from the supplier to the retailer is communicated to the supplier in period t, but is communicated to the retailer later in period $t+L_R$, when the retailer actually receives the shipment. Similarly, the shipment from the supplier's outside source to the supplier is communicated to the supplier in period $t+L_S$. Thus, at the beginning of each period t, the information available to the retailer from the protocol is the shipment observation $SH_j^R; j \leq t-L_R$ and his demand observations $d_j; j < t$. Similarly, at the beginning of each period t, information available to the supplier from the protocol is the shipment observation $SH_j^S; j \leq t-L_S$, and the shipments to the retailer $SH_j^R; j \leq t$. Theorem 4 establishes the non-inversability of future demand signal property of our protocols.

THEOREM 4. *In period t, the retailer cannot infer the supplier's future demand signals $\delta_{j,i}^s; j > t$ with probability one from the shipment information $SH_j^R; j \leq t-L_R$ provided by the protocol. Similarly, in period t, the supplier cannot infer the retailer's future demand signals $\delta_{j,i}^r; j > t$ with probability one from the shipment information $SH_j^R; j \leq t$ provided by the protocol.*

Theorem 4 shows that the protocol does not leak the values and the timing of future demand signals. A related question is whether the retailer can infer the supplier's past demand signals from the

shipment information. This question is complicated because the shipment quantity does not provide complete information on the retailer's desired order quantity (see footnote 5 in Chen (2004)). To answer this question, one needs to characterize the (stochastic) process followed by the shipment variable, $SH_t^R = \min\{q_t^R, OH_t^S\}$. If the supplier's probability of stocking out is non-negligible, then the shipment quantity observations provide censored data on the desired order quantities. We next characterize the invertibility of past demand signals from the shipment information received by the retailer, under the assumption of negligible probability of supplier stockouts.

THEOREM 5. *If the probability of a supplier stockout is negligible, then as $t \to \infty$, the retailer can infer the impact of the supplier's past demand signals, $\theta^s \sum_{i=t-L_R}^{t} \sum_{k=i-(t-L_R)+1}^{T} \delta_{i,k}^s$, with probability one from the shipment information $SH_j^R; j \leq t - L_R$ provided by the protocol.*

While the impact of the supplier's past demand signals may be invertible, if the supplier's stockout probability is negligible, we do not consider this as a breach of security. For example, if the demand signals δ signify promotion plans, then they typically can be observed directly by the other party when the promotion occurs during the demand period. Hence, the protocol leaks information that may be available to the supply-chain partner through other means. When the supplier's probability of stockout is non-negligible, it is significantly more difficult to infer the demand signals from the shipment information. We leave the complete characterization of information leakage under non-negligible supplier stock-out probability for future research.

Note that our single-supplier single-retailer model provides the most stringent test of inverse optimization. In practice, these protocols are likely to be implemented with several retailers. This results in "hiding in the crowd" phenomenon, making it almost impossible for one party to invert the private information of any other supply-chain partner. Also, in our model we assumed that the standard deviation of the demand signals and the white noise of the demand process is known to all parties. In practice, these are likely to be unknown, and hence required to be estimated from demand observations. While it would be straightforward to construct a secure protocol to do so, it would make inverse-optimization even more difficult. Finally, we assumed that the cost parameters to the protocol are stationary. If these parameters fluctuate (very likely in practice), then the protocols need to be adjusted so that the cost parameters are input every period and the critical fractiles will need to be recomputed every period. This feature will likely make inverse-optimization very difficult.

Our SCPFR protocols assumed the honest-but-curious model, i.e., the supplier and retailer follow the protocols, but may try to learn additional information about the other party's private input.

In real applications, however, a participant of the SCPFR protocols may not necessary follow the protocols if her malicious behavior cannot be detected by the other party. Recall that in section 4 a common technique for hiding of private data was to additively split it between the players. Because of the additive nature of data sharing, it is possible for a malicious party to gain advantage by distorting the data. For example, a malicious retailer may increase her share of the transfer payment value in order to receive a larger amount of payment from the supplier than prescribed by the protocol in the last step of the SCPFR process. In what follows, we propose two practical guidelines to address this issue. Note that it is possible to use general solutions to SMC to make the SCPFR protocols secure against malicious adversaries; such solutions, however, are expensive to be used in practice.

1. Instead of revealing the shipment quantity in step 5 and transfer payment in step 6 of the SCPFR process, these values are computed and used concurrently. Hence, tampering with the transfer payments will distort the order quantities and vice versa. Hence, the incentive compatibility of the transfer payment scheme will force the parties to follow the protocol.

2. In many business scenarios (including the SCPFR problem), information needs to be kept private for only a certain period of time. For example, the signals the supplier and retailer observe at time t may no longer be sensitive at time $t+L_R+L_S$, where L_R+L_S is the supply chain lead-time. A practical solution to enforce the participants to follow the protocols is then to make the protocols verifiable in the future (e.g., Schneier 1996). More specifically, the supplier and retailer commit, at the time of the protocol execution, to all of their private inputs, private keys, randomly generated numbers, and intermediate results of the protocols, and save the transcripts of the protocols. For efficiency, they can use an integer commitment scheme (e.g., Cramer and Damgård 1998, Fujisaki and Okamoto 1997) or even in some cases a cryptographic hash function (Schneier 1996). After a certain time, when the private inputs are no any longer sensitive, the supplier and retailer can open the commitments, reveal all their private inputs and intermediate results, and individually simulate the protocol. If one of them did not follow the protocols in the original execution, he/she will be caught during the second execution.

6. Conclusions

This paper analyzed privacy/security issues in supply-chains, and demonstrated that collaborative forecasting and inventory planning can be conducted by sharing, but without *disclosing* private information of any supply-chain partners. We examined a two-echelon supply-chain with one supplier and one retailer facing non-stationary demand. We constructed secure protocols for forecasting

demand based on private demand signals observed by the retailer and supplier. Using these forecasts, and the supplier's and retailer's private cost information, we securely computed echelon base-stock levels that minimize the total supply-chain cost/period. Using the retailer's and supplier's private inventory level information, and the echelon base-stock levels, the order/shipment quantities were computed securely. We constructed a linear transfer-payment scheme to enable incentive-compatibility of the collaborative forecasting process. We also provided an inverse-optimization analysis of the SCPFR process, and proved that the future demand signals observed by the retailer and the supplier cannot be inferred from the output of the protocol. A simulation study demonstrated that, depending on problem parameters, the savings of a SCPFR policy over a non-collaborative policy can be significant.

This paper is a first step in analyzing privacy/security issues in supply-chain management and this research can be extended in several ways. We assumed a linear demand model based on a Martingale model of forecast evolution. It will be interesting to extend our results to other demand models such as Bayesian demand models. More realistic supply-chain settings involving multiple suppliers, retailers and echelons should also be examined. We conjecture that such settings will enhance the privacy preserving nature of our protocols due to increased "hiding-in-the-crowd" phenomenon. More modeling effort is needed in understanding equilibrium issues in non-collaborative settings where information is not shared. This is useful in establishing benchmarks for the benefits of the SCPFR process. Many extensions are possible from a secure-multiparty-computation perspective. Our analysis can be extended to the "malicious" model of security, but the protocols will be more complex for these settings. Further experimentation needs to be carried out on comparing the computational efficiency of the split arithmetic versus boolean circuit approaches outlined in this paper. Tradeoffs between efficiency of protocols versus information leakage also need to be examined. Methods for checking the honesty of participants should be developed, particularly in situations where the value of information does not decay over time and hence commitment schemes cannot be used. Finally, we leave an analysis of the fairness of the division of SCPFR savings between the retailer and the supplier for future research.

Acknowledgments

Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center. We thank Xin Zhai for computational assistance on the simulation study. We also thank Joe Andraski, vice-chairman of the VICS CPFR committee, for supporting our research on secure supply-chain collaborations.

References

Abramowitz, M., I. Stegun. 1972. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. U.S. Department of Commerce.

Aho, A., J. Hopcrof, J. Ullman. 1974. *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Ahuja, P., J. Orlin. 2001. Inverse optimization. *Operations Research* **49(5)** 771–783.

Ajtai, M., J. Komlos, E. Szemeredi. 1983. An o((n log n)) sorting network. *15th annual ACM Symposium on Theory of Computing*. 1–9.

Alt, H. 1988. Comparing the combinational complexities of arithmetic functions. *Journal of the ACM* **35(2)** 447–460.

Anand, K., M. Goyal. 2005. Incentives for information acquisition and information dissemination in a supply-chain. Working Paper. The Wharton School, University of Pennsylvania.

Asokan, N., M. Schunter, M. Waidner. 1997. Optimistic protocols for fair exchange. *4th ACM Conference on Computer and Communications Security*. 7–17.

Asokan, N., V. Shoup, M. Waidner. 2000. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications* **18(4)** 593–610.

Atallah, M., M. Bykova, K. Frikken J. Li, M. Topkara. 2004. Private collaborative forecasting and benchmarking. *3rd ACM Workshop on Privacy in Electronic Society*.

Atallah, M., V. Deshpande, H. Elmongui, L. Schwarz. 2003. Secure supply chain protocols. *IEEE International Conference on E-Commerce*.

Aviv, Y. 2001. The effect of collaborative forecasting on supply chain performance. *Management Science* **47** 1326–1343.

Aviv, Y. 2002. Gaining benefits from joint forecasting and replenishment processes: The case of auto-correlated demand. *Manufacturing & Service Operations Management* **4** 55–74.

Aviv, Y. 2003. A time-series framework for supply-chain inventory managment. *Operations Research* **51** 210–227.

Bar-Ilan, J., D. Beaver. 1989. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. *8th Annual ACM Symposium on Principles of Distributed Computing*. 201–209.

Batcher, K. 1968. Sorting networks and their applications. *AFIPS Spring Joint Computer Conference*. 307–314.

Beame, P., S. Cook, H. Hoover. 1984. Log depth circuits for division and related problems. *16th Annual IEEE Symposium on Foundations of Computer Science*. 1–6.

Brandt, F., T. Sandholm. 2004. On correctness and privacy in distributed mechanisms. *Proceedings of the Agent-Mediated Electronic Commerce (AMEC) workshop*.

Brandt, F., T. Sandholm. 2005. Efficient privacy-preserving protocols for multi-unit auctions. *Proceedings of Financial Cryptography and Data Security (FC)*.

Cachon, G. 2003. Supply chain coordination with contracts. *Handbooks in Operations Research and Management Science: Supply Chain Management* .

Cachon, G., M. Fisher. 2000. Supply chain inventory management and the value of shared information. *Management Science* **46(8)** 1032–1048.

Cachon, G., M. Lariviere. 2001. Contracting to assure supply: how to share demand forecasts in a supply chain. *Management Science* **47(5)** 629–646.

Cachon, G., P. Zipkin. 1999. Competitive and cooperative inventory policies in a two-stage supply chain. *Management Science* **45(7)** 936–953.

Canetti, R. 2000. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology* **13(1)** 143–202.

Chen, F. 1998. Echelon reorder points, installation reorder points, and the value of centralized demand information. *Management Science* **44(12)** 221–234.

Chen, F. 2001. Auctioning supply contracts. Working paper, Columbia University.

Chen, F. 2004. Information sharing and supply chain coordination. *Handbook of Operations Research and Management Science: Supply Chain Management*. North-Holland, Amsterdam.

Chen, F., B. Yu. 2001. Quantifying the value of leadtime information in a single-location inventory system. Working paper, Columbia University.

Chen, F., Y. Zheng. 1994. Lower bounds for multi-echelon stochastic inventory systems. *Management Science* **40(11)** 1426–1443.

Clark, A., H. Scarf. 1960. Optimal policies for a multi-echelon inventory problem. *Management Science* **6(4)** 475–490.

Clifton, C., I. Iyer, R. Cho, W. Jiang, M. Kantarcioglu, J. Vaidya. 2004. An approach to identifying beneficial collaboration securely in decentralized logistics systems. Working Paper, Purdue University.

Conitzer, V., T. Sandholm. 2002. Complexity of mechanism design. *18th Conference on Uncertainty in Artificial Intelligence*. 103–110.

Corbett, C. 2001. Stochastic inventory systems in a supply chain with asymmetric information: Cycle stocks, safety stocks, and consignment stock. *Operations Research* **49(4)** 487–500.

Cramer, R., I. Damgård. 1998. Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? *Advances in Cryptology - CRYPTO 1998*. 424–441.

Damgård, I., M. Jurik. 2001. A generalization, a simplification and some applications of paillier's probabilistic public-key system. *4th International Workshop on Practice and Theory in Public Key Cryptography*. 119–136.

Decker, B. De, G. Neven, F. Piessens, E. Van Hoeymissen. 2001. Second price auctions, a case study of secure distributed computing. *3rd IFIP International Working Conference on Distributed Applications and Interoperable Systems*. 217–228.

Deshpande, V., L. Schwarz. 2005. Optimal capacity choice and allocation in decentralized supply chains. Working paper, Purdue University.

Elkind, E., H. Lipmaa. 2004. Interleaving cryptography and mechanism design: The case of online auctions. *8th Annual Conference on Financial Cryptography*. 117–131.

Federgruen, A., P. Zipkin. 1984. Computational issues in an infinite-horizon, multiechelon inventory model. *Operations Research* **32(4)** 818–836.

Feigenbaum, J., N. Nisan, V. Ramachandran, R. Sami, S. Shenker. 2002. Agents' privacy in distributed algorithmic mechanisms. *Workshop on Economics and Information Security*.

Feigenbaum, J., S. Shenker. 2002. Distributed algorithmic mechanism design: Recent results and future directions. *6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. 1–13.

Feldman, P., S. Micali. 1988. Optimal algorithms for byzantine agreement. *20th Annual ACM Symposium on Theory of Computing*. 148–161.

Franklin, M., M. Reiter. 1996. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering* **22(5)** 302–312.

Freedman, M., K. Nissim, B. Pinkas. 2004. Efficient private matching and set intersection. *Advances in Cryptology - EUROCRYPT 2004*. 1–19.

Frikken, K., M. Atallah. 2004. Privacy preserving route planning. *3rd ACM Workshop on Privacy in the Electronic Society*. 8–15.

Fujisaki, E., T. Okamoto. 1997. Statistical zero knowledge protocols to prove modular polynomial relations. *Advances in Cryptology - CRYPTO 1997*. 16–30.

Gaur, V., A. Giloni, S. Seshadri. 2005. Information sharing in a supply chain under arma demand. *Management Science* **51** 961–969.

Gavirneni, S., R. Kapuscinski, S. Tayur. 1999. Value of information in capacitated supply chains. *Management Science* **45(1)** 16–24.

Goethals, B., S. Laur, H. Lipmaa, T. Mielikainen. 2004. On secure scalar product computation for privacy-preserving data mining. *7th Annual International Conference on Information Security and Cryptology*. 104–120.

Goldreich, O. 2004. *The Foundations of Cryptography*. Cambridge University Press.

Goldreich, O., S. Micali, A. Wigderson. 1987. How to play any mental game. *19th Annual ACM Symposium on Theory of Computing*. 218–229.

Goldwasser, S. 1997. Multi-party computations: Past and present. *16th Annual ACM Symposium on Principles of Distributed Computing*. 1–6.

Graves, S. 1999. A single-item inventory model for a nonstationary demand process. *Manufacturing & Service Operations Management* **1(1)** 50–61.

Hays, C.L. 2004. What wal-mart knows about customers' habits. *The New York Times*, Nov. 14.

Jakobsson, M., A. Juels. 2000. Mix and match: Secure function evaluation via ciphertexts. *Advances in Cryptology - ASIACRYPT 2000*. 162–177.

Katz, J., R. Ostrovsky. 2004. Round optimal secure two-party computation. *Advances in Cryptology - CRYPTO 2004*. 335–354.

Lee, H., P. Padmanabhan, S. Whang. 1997a. The bullwhip effect in supply chains. *Sloan Management Review* **38** 93–102.

Lee, H., V. Padmanabhan, S. Whang. 1997b. Information distortion in a supply chain: The bullwhip effect. *Management Science* **43(4)** 546–558.

Lee, H., K. So, C. Tang. 2000. The value of information sharing in a two-level supply chain. *Management Science* **46(5)** 626–643.

Lee, H., S. Whang. 1999. Decentralized multi-echelon supply chains: Incentives and information. *Management Science* **45(5)** 633–640.

Lee, H., S. Whang. 2000. Information sharing in a supply chain. *Int. J. Technology Management* **20(3/4)** 373–387.

Li, L. 2003. Information sharing in a supply chain with horizontal competition. *Management Science* **48(9)** 1196–1212.

Malkhi, D., N. Nisan, B. Pinkas, Y. Sella. 2004. Fairplay - a secure two-party computation system. *13th USENIX Security Symposium*. 287–302.

Mishra, B., S. Raghunathan, X. Yue. 2005. Information sharing in supply chains: Incentives for information distortation. Working Paper, University of Wisconsin at Milwaukee.

Miyaoka, J., W. Hausman. 2004. How a base stock policy using "stale" forecasts provides supply chain benefits. *Manufacturing & Service Operations Management* **6** 149–162.

Moinzadeh, K. 2002. A multi-echelon inventory system with information exchange. *Management Science* **48** 414–426.

Naor, M., B. Pinkas. 1999. Oblivious transfer and polynomial evaluation. *31st Annual ACM Symposium on Theory of Computing*. 245–254.

Naor, M., B. Pinkas, R. Sumner. 1999. Privacy preserving auctions and mechanism design. *1st ACM Conference on Electronic Commerce*. 129–139.

Ofman, Y. 1963. On the algorithmic complexity of discrete functions. *Sov. Phys. Dokl.* **7** 589–591.

Okamoto, T., S. Uchiyama. 1998. A new public-key cryptosystem as secure as factoring. *Advances in Cryptology - EUROCRYPT 1998*. 308–318.

Paillier, P. 1999. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology - EUROCRYPT 1999*. 223–238.

Raghunathan, S. 2001. Information sharing in a supply chain: A note on its value when demand is nonstationary. *Management Science* **47(4)** 605–610.

Reif, J., S. Tate. 1989. Optimal size integer division circuits. *21st annual ACM Symposium on Theory of Computing*. 264–273.

Schneier, B. 1996. *Applied Cryptography*. 2nd ed. John Wiley & Sons.

Schoehage, A., V. Strassen. 1971. Schnelle multiplikation grosser zahlen. *Computing* **7** 281–292.

Tarantola, A. 1987. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation.* Elsevier, Amsterdam.

Yao, A. 1982. Protocols for secure computations. *23rd Annual IEEE Symposium on Foundations of Computer Science*.

Yao, A. 1986. How to generate and exchange secrets. *27th Annual IEEE Symposium on Foundations of Computer Science*.

Appendix A: A Brief Overview of SMC

Whereas Section 2.2 contained a high-level literature review of SMC, this section reviews in more technical detail the concepts and results from the SMC literature that are used in the paper. These include what is meant by "secure" computation, and general results from the simulation of circuits literature.

A.1. Security Model

We begin by discussing the security model, which is a standard security model (Canetti 2000, Goldreich 2004). At a high level, a protocol "securely" implements a function f if the information that can be learned by engaging in the protocol, could be learned in an ideal implementation of the protocol where the functionality was provided by a trusted oracle. We also consider two types of adversaries: (i) semi-honest adversaries that will follow the protocol exactly but will try to compute "extra" information; and (ii) malicious adversaries that will deviate from the protocol at any step in order to gain additional information or to control the outcome. The types of things that an adversary could do include: (i) substitute inputs (ideal, semi-honest, and malicious), (ii) deviate at an intermediate step (malicious), and (iii) terminate early (malicious ideal and malicious). Our protocols do not attempt to address item (iii); in a sense this is the fairness constraint (it may be possible for one party to learn the result and then terminate the protocol not allowing the other party to learn the result). The reason that such attacks are ignored is that the solutions for such tasks are complex and do not completely solve the problem. See discussion below.

We now formally define the notions above. We do this by defining the notion of an ideal-model adversary (one for the situation where there is an oracle) and a real-model adversary for the protocol Π, and then assert that a protocol is secure if the two executions are computationally indistinguishable. We focus on defining the case for two-party protocols (and refer the reader to Canetti (2000) for multiple parties). Assume the Π computes $f : \{0,1\}^\star \times \{0,1\}^\star \to \{0,1\}^\star$.

Definition of IDEAL model:

The ideal model can be viewed as two Probabilistic Polynomial Time (PPT) algorithms (A, B) each of which is composed of two parts A_I and A_O (and B_I and B_O), and the execution of the protocol is as follows:

1. Alice (Bob) sends $A_I(X_A, r_A)$ (resp., $B_I(X_B, r_B)$) to the oracle (where r_A and r_B are Alice and Bob's respective coin flips).

2. The oracle evaluates $f(A_I(X_A, r_A), B_I(X_B, r_B))$ obtaining output (Y_A, Y_B), and sending Y_A (Y_B) to Alice (resp., Bob).

3. Alice (Bob) outputs $A_O(X_A, r_A, Y_A)$ (resp., $B_O(X_B, r_B, Y_B)$).

Alice is said to be *honest* if $A_I(X_A, r_A) = X_A$ and $A_O(X_A, r_A, Y_A) = Y_A$ (a similar definition holds for Bob). We say that an adversary is *admissible* if at least one party is honest (i.e., we do not concern ourselves with adversaries that corrupt both parties).

We now define the ideal model's view in the case where Bob is honest (an analogous definition occurs when Alice is honest), there are two cases:

- Alice does not terminate, then $IDEAL_{A,B}(X_A, X_B) = (A_O(X_A, r_A, Y_A), f(A_I(X_A, r_A), X_B))$.

- Alice terminates, then $IDEAL_{A,B}(X_A, X_B) = (\perp, f(A_I(X_A, r_A), X_B))$.

We now define the actual execution for a protocol Π that implements the function f.

Definition of REAL model:

In a real model the parties are arbitrary PPT algorithms (A', B'), where for semi-honest adversaries the output function is arbitrary and for malicious adversaries the parties can behave arbitrarily. The adversaries are admissible if at least one party uses the algorithm specified by protocol Π. We define the interaction of protocol Π by $REAL_{\Pi,A',B'}(X_A, X_B)$ as the output from the interaction of $A'(X_A)$ and $B'(X_B)$ for protocol Π.

As is usual, we assert a protocol Π is secure if there is an ideal-model adversary that is as powerful as any real-model adversary up to a negligible degree. To define what is meant by this, we use the standard definition of computational indistinguishability (Goldreich 2004):

Definition of Computational Indistinguishability:

We say that two random variables X and Y are computationally indistinguishable if for any PPT algorithm D, any polynomial p, and large enough n, it holds that:

$$|(Pr(D(X) = 1)) - (Pr(D(Y) = 1))| < \frac{1}{p(n)}$$

Definition of security: We say that a protocol Π securely evaluates a function f if for any admissible adversary in the real model (A', B'), there exists an admissible ideal-model adversary (A, B) such that $IDEAL_{A,B}(X_A, X_B)$ and $REAL_{\Pi,A',B'}(X_A, X_B)$ are computationally indistinguishable.

Appendix B: Proofs of Theorems and Propositions

Proof of Proposition 1 y_t^R is chosen to minimize $G_t^1(y_t^R) = h_1\sigma_{t,[t,t+L_R]}z_1 + (h_1+h_2+p)\sigma_{t,[t,t+L_R]}LF(z_1)$ Therefore,

$$G_t^{1'}(y_t^R) = h_1\sigma_{t,[t,t+L_R]} + (h_1+h_2+p)\sigma_{t,[t,t+L_R]}(\Phi(z_1)-1)$$

Setting the derivative, w.r.t to y_t^R, equal to zero, we get the first order condition

$$(h_1+h_2+p)\Phi(z_1^*) - (h_2+p) = 0$$

From the convexity of the cost function, solution to the above equation is optimal. □

Proof of Proposition 2 Our proof follows from Federgruen and Zipkin (1984), who established the optimal base-stock levels for the stationary demand case. For non-stationary demand, y_t^S is chosen to minimize

$$G_t^2(y_t^S) = h_2(y_t^S - \mu_{t,[t,t+L_S-1]}) + E_{U>y_t^S}[G_{t+L_S}^1(y_t^S - d_{[t,t+L_S-1]}) - G_{t+L_S}^1(y_{t+L_S}^{R*})]$$

Taking derivative with respect to y_t^s, we get the first order condition

$$h_2 + \int_{y_t^S}^{\infty} G_{t+L_S}^{1'}(y_t^S - d_{[t,t+L_S-1]})f_U(u)du = 0$$

This can be written as

$$h_2 + \int_{y_t^S}^{\infty}\{-(h_2+p) + (h_1+h_2+p)F_{t+L_S,[t+L_S,t+L_S+L_R]}(y_t^S - d_{[t,t+L_S-1]})\}f_U(u)du = 0$$

Where, $F_{t+L_S,[t+L_S,t+L_S+L_R]}(\cdot)$ is the cdf of the retailer's lead-time demand distribution at $t+L_S$. Now $U = d_{[t,t+L_S-1]} + y_{t+L_S}^{R*}$ is a normally distributed random variable with mean μ_U and standard deviation σ_U, where $\mu_U = \mu_{t,[t,t+L_S+L_R]} + z_1^*\sigma_{t+L_S,[t+L_S,t+L_S+L_R]}$ and $\sigma_U^2 = \sigma_{t,[t,t+L_S+L_R]}^2 - \sigma_{t+L_S,[t+L_S,t+L_S+L_R]}^2$. Also, define the random variable $V_t = d_{[t,t+L_S+L_R]}$. Note that V_t is also a normal random variable with a mean $\mu_V = \mu_{t,[t,t+L_S+L_R]}$ and standard deviation $\sigma_V = \sigma_{t,[t,t+L_S+L_R]}$ as given by equations (2) and (3), respectively.

The above first order condition reduces to

$$h_2 - (h_2+p)\int_{y_t^S}^{\infty} f_U(u)du + (h_1+h_2+p)Prob(U \geq y_t^S; V \leq y_t^S) = 0$$

which can be rewritten as

$$h_2 - (h_2+p_1+p_2)\Phi(z_2(y_t^S)) + (h_1+h_2+p_1+p_2)\Phi(z_2(y_t^S); z_3(y_t^S); \frac{-\sigma_U}{\sigma_V}) = 0$$

where $\Phi(x_1; x_2; \rho)$ denotes the cdf of the standard *bi-variate* normal distribution with correlation ρ, $z_2(y_t^S) = -(y_t^S - \mu_U)/\sigma_U$, and $z_3(y_t^S) = (y_t^S - \mu_V)/\sigma_V$. This is because U and V have a joint-distribution which is bi-variate normal.

From the convexity of the G_t^2 cost function, solution to the above equation is optimal. □

Proof of Theorem 1 The retailer's cost, associated with the retailer's base-stock decision, y_t^R is given by

$$G_t^{1,R}(z_1) = (h_1+h_2)\sigma_{t,[t,t+L_R]}z_1 + (h_1+h_2+p_1)\sigma_{t,[t,t+L_R]}LF(z_1)$$

Where $z_1 = (y_t^R - \mu_{t,[t,t+L_R]})/\sigma_{t,[t,t+L_R]}$. Let $\tilde{\mu}_{t,[t,t+L_R]}$ be the retailer's reported mean lead-time demand forecast (which could be different than the true value $\mu_{t,[t,t+L_R]}$). Then, the effective z_1 value is given by $z_1 = (\tilde{\mu}_{t,[t,t+L_R]} - \mu_{t,[t,t+L_R]} + z_1^*\sigma_{t,[t,t+L_R]})/\sigma_{t,[t,t+L_R]}$, and z_1^* is computed from equation (7). Note that if there is no distortion of the lead-time forecast, i.e., $\tilde{\mu}_{t,[t,t+L_R]} = \mu_{t,[t,t+L_R]}$, then $z_1 = z_1^*$. We next establish the retailer's incentive to report the true lead-time demand forecast, by examining the derivative of the cost function at the true value of lead-time demand forecast.

$$\frac{\partial G_t^{1,R}}{\partial \tilde{\mu}_{t,[t,t+L_R]}} = \frac{\partial G_t^{1,R}}{\partial z_1}\frac{\partial z_1}{\partial \tilde{\mu}_{t,[t,t+L_R]}} = -p_1 + (h_1+h_2+p1)\Phi(z_1)$$

Hence

$$\frac{\partial G_t^{1,R}}{\partial \tilde{\mu}_{t,[t,t+L_R]}}\,|\tilde{\mu}_{t,[t,t+L_R]} = \mu_{t,[t,t+L_R]} = -p_1 + (h_1+h_2+p1)\Phi(z_1^*) = -p_1 + \frac{(h_1+h_2+p_1)(h_2+p_1+p_2)}{(h_1+h_2+p_1+p_2)} > 0$$

Thus, if the retailer's base-stock level is determined using proposition (1), the retailer can lower his cost by deflating the lead-time demand forecast $\mu_{t,[t,t+L_R]}$. Similarly, it can be shown that the supplier can lower his cost by inflating the lead-time demand forecast $\mu_{t,[t,t+L_R]}$. □

Proof of Theorem 2 The retailer's cost, associated with the retailer's base-stock decision, y_t^S is given by

$$G_t^{2,R}(y_t^S) = E_{U>y_t^S}\{G_{t+L_S}^{1,R}(y_t^S - d_{[t,t+L_S-1]}) - G_{t+L_S}^{1,R}(y_{t+L_S}^{R*})\}$$

Let $\tilde{\mu}_{t,[t,t+L_R+L_S]}$ be the retailer's reported mean lead-time demand forecast (which could be different than the true value $\mu_{t,[t,t+L_R+L_S]}$). Then, the effective z_3 value is given by $z_3 = (\tilde{\mu}_{t,[t,t+L_R+L_S]} - \mu_{t,[t,t+L_R+L_S]} + z_3^*\sigma_V)/\sigma_V$, and z_3^* is computed from equation (7). Note that if there is no distortion of the lead-time forecast, i.e., $\tilde{\mu}_{t,[t,t+L_R+L_S]} = \mu_{t,[t,t+L_R+L_S]}$, then $z_3 = z_3^*$ (and $z_2 = z_2^*$). We next establish the retailer's incentive to report the true lead-time demand forecast, by examining the derivative of the cost function at the true value of lead-time demand forecast.

$$\frac{\partial G_t^{2,R}}{\partial \tilde{\mu}_{t,[t,t+L_R+L_S]}} = \frac{\partial G_t^{2,R}}{\partial y_t^S}\frac{\partial y_t^S}{\partial \tilde{\mu}_{t,[t,t+L_R+L_S]}} = -p_1\Phi(z_2) + (h_1+h_2+p_1)\Phi(z_2;z_3;\rho)$$

Hence

$$\begin{aligned}\frac{\partial G_t^{2,R}}{\partial \tilde{\mu}_{t,[t,t+L_R+L_S]}}\,|\tilde{\mu}_{t,[t,t+L_R+L_S]} = \mu_{t,[t,t+L_R+L_S]} &= -p_1\Phi(z_2^*) + (h_1+h_2+p1)\Phi(z_2^*;z_3^*;\rho)\\ &= -h_2(1-\Phi(z_2^*)) + p_2(\Phi(z_2^*) - \Phi(z_2^*;z_3^*;\rho))\\ &\gtreqless 0\end{aligned}$$

Note that the above derivative is negative if $p_2 = 0$, but can be positive if p_2 is sufficiently large. Thus, if the supplier's base-stock level is determined using proposition (2), the retailer can lower his cost by distorting the lead-time demand forecast $\mu_{t,[t,t+L_R+L_S]}$, depending on the values of h_2 and p_2. Similarly, it can be shown that the supplier can lower his cost by distorting the lead-time demand forecast $\mu_{t,[t,t+L_R+L_S]}$. □

Proof of Theorem 3: The proof directly follows from the proofs of theorems (1) and (2). It is easy to verify that

$$\frac{\partial(G_t^{1,R} + TP_t)}{\partial \tilde{\mu}_{t,[t,t+L_R]}}\,|\tilde{\mu}_{t,[t,t+L_R]} = \mu_{t,[t,t+L_R]} = 0$$

and

$$\frac{\partial(G_t^{2,R}+TP_t)}{\partial\tilde{\mu}_{t,[t,t+L_R+L_S]}}\,|\tilde{\mu}_{t,[t,t+L_R+L_S]}=\mu_{t,[t,t+L_R+L_S]}=0 \qquad \square$$

Proof of Theorem 4 In each period t, the retailer's shipment quantity is based on the on-hand order quantity of the supplier and the retailer's order quantity, i.e. $SH_t^R=\min\{q_t^R,OH_t^S\}$. Now, the retailer's order quantity q_t^R is based on the retailer's base-stock level y_t^R, which contains information about demand signals observed, by both the retailer and the supplier, for the time interval $[t,t+L_R]$ as observed by time $t-1$. Now, the shipment received by the retailer in period t is the shipment that was released by the supplier in period $t-L_R$. This shipment contains information about demand signals over the time-interval $[t-L_R,t]$, but contains no information about demand signals about future periods. Hence, in any time period t, the protocol does not reveal any information to the retailer that contains the supplier's future observed demand signals. Hence, the retailer cannot infer the supplier's future demand signals from the output of the protocol.

The protocol does however reveal SH_t^R to the supplier, which contains information about the retailer's future demand signals. Under the assumption that the supplier's probability of stockout is negligible, the retailer's order process can be characterized as $q_t^R=y_t^{R*}-y_{t-1}^{R*}+d_{t-1}$. This can be rewritten as

$$q_t^R=\mu+\theta^r(\sum_{j=t}^{t+L_R}\delta_{j,j-t}^r+\sum_{i=L_R+1}^{T}\delta_{t+L_R,i}^r)+\theta^s(\sum_{j=t}^{t+L_R}\delta_{j,j-t}^s+\sum_{i=L_R+1}^{T}\delta_{t+L_R,i}^s)+\epsilon_{t-1}$$

However, since the protocol does not reveal retail demand to the supplier, the supplier does not know the white noise of the demand process, ϵ_{t-1}. Hence, even if the supplier get's the retailer order quantity information q_t^R, he cannot infer the retailer's future demand signals $\delta_j^r; j>t$. $\square$

Proof of Theorem 5 If the supplier's probability of stock-out is negligible, then the retailer, at any time t, knows $q_j^R; j\le t-L_R$, by setting $q_j^R=SH_j^R$. Hence, the retailer can infer the retailer's base-stock level by setting $y_j^{R*}=q_j^R+IP_j^R$ (since the retailer knows his echelon inventory-position from his on-hand inventory and backorders). As $t\to\infty$, the retailer can infer the mean of the retailer's base-stock level $(L_R+1)\mu+z_1^*\sigma_{L_R+1}$, by regressing y_j^{R*} versus his observed signals δ^r. Hence from each $y_j^{R*}; j\le t-L_R$, the retailer can infer the impact of supplier's past demand signals $\theta^s\sum_{i=j}^{j+L_R}\sum_{k=i-j+1}^{T}\delta_{i,k}^s$. $\square$

Appendix C: Building Blocks used in SCPFR

C.1. Split Division

Suppose x and y are additively split between Supplier and Retailer, and they want to compute x/y. A split division protocol enables them to carry out this computation in a secure, additively-split fashion. Supplier then obtains $z^{(s)}$ and Retailer obtains $z^{(r)}$ such that $z^{(s)} + z^{(r)} = x/y = (x^{(s)} + x^{(r)})/(y^{(s)} + y^{(r)})$. The division operation is different from other computations described so far because it must handle floating point arithmetic. Several efficient split division protocols were given by Atallah et al. (2004).

Complexity Analysis: All of the division protocols given in Atallah et al. (2004) for two players and constant size numbers will result in a constant number of rounds, constant communication and computational complexity (measured in modular exponentiations). Certain protocols are more efficient than others; this, however, of interest only for actual implementations and does not affect our analysis.

C.2. Secure Scalar Product

Suppose Supplier has a private vector $\vec{x} = (x_1, x_2, \ldots, x_n)$, and Retailer has a private vector $\vec{y} = (y_1, y_2, \ldots, y_n)$, a secure scalar product protocol enables them to compute $\vec{x} \cdot \vec{y}$ securely in additively-split fashion. That is, Supplier obtains $z^{(s)}$ and Retailer obtains $z^{(r)}$ such that $z^{(s)} + z^{(r)} = \vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i \cdot y_i$. Goethals et al. (2004) give a construction of a secure scalar product protocol using homomorphic encryption schemes.

Another variant of the scalar protocol is where two vectors $\vec{x} = (x_1, x_2, \ldots, x_n)$ and $\vec{y} = (y_1, y_2, \ldots, y_n)$ are additively split between Supplier and Retailer. A secure *split* scalar product protocol then enables them to compute $\vec{x} \cdot \vec{y}$ securely in split fashion. That is, Supplier obtains $z^{(s)}$ and Retailer obtains $z^{(r)}$ such that $z^{(s)} + z^{(r)} = \vec{x} \cdot \vec{y} = \sum_{i=1}^{n} (x_i^{(s)} + y_i^{(r)}) \cdot (y_i^{(s)} + y_i^{(r)})$. Similar to the above case, secure split scalar product can be implemented using a homomorphic encryption scheme since it relies on the use of only addition and multiplication operations.

Complexity Analysis: As can be seen from the above, secure scalar product involves a constant number of rounds and requires us to perform a $O(n)$ modular exponentiations. Communication complexity is also $O(n)$.

C.3. Secure Polynomial Evaluation

Suppose Supplier has a private polynomial P of degree k, and Retailer has a private integer x. Secure polynomial evaluation described in Freedman et al. (2004) enables Supplier and Retailer to compute $P(x)$ securely in additively-split fashion. Let $P(x) = a_k x^k + \cdots + a_2 x^2 + a_1 x + a_0$. Using homomorphic encryption, Supplier sends the encrypted coefficients $E(a_k), \ldots, E(a_1), E(a_0)$ to Retailer in order. Retailer computes $E(P(x)) = E(a_k)^{x^k} + \cdots + E(a_1)^x + E(a_0)$, then chooses a random r, and sends $E(P(x) - r)$ to Supplier. Now $P(x)$ is additively split between Supplier and Retailer.

Complexity Analysis: As can be seen from the above, secure polynomial evaluation involves a constant number of rounds and requires us to perform a $O(k)$ modular exponentiations. Communication complexity is also $O(k)$.

C.4. Secure Matrix Multiplication

Suppose an $\ell \times m$ matrix M_1 and an $m \times n$ matrix M_2 are additively split between Supplier and Retailer. Secure matrix multiplication enables them to compute $M_3 = M_1 \times M_2$ securely with the result in a split fashion. Let $m_k^{(i,j)}$ denote the value of the cell at position (i,j) in matrix M_k, where $k = 1,2,3$. Then for M_3's entries $m_3^{(i,j)}$, where $1 \le i \le \ell$ and $1 \le j \le n$, we have the following equation:

$$m_3^{(i,j)} = \sum_{k=1}^{m} m_1^{(i,k)} m_2^{(k,j)}$$

Therefore, $m_3^{(i,j)}$ can be computed in a split fashion using one secure split scalar product. M_3 then can be computed by Supplier and Retailer using the secure split scalar product protocol ℓn times.

Complexity Analysis: Secure matrix multiplication can be done with a single round of interaction. It requires $O(m\ell n)$ modular exponentiations and $O(\ell m + \ell n)$ communication.

C.5. Secure 3×3 Matrix Inversion

In this section we introduce a protocol for computing the inverse of a 3×3 matrix. There have been other schemes introduced for inverting matrices (Bar-Ilan and Beaver 1989). However, these schemes find the inverse when using modular arithmetic, which is not the same as the inverse when using standard arithmetic.

Let A be a 3×3 matrix, i.e.,

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

We can compute A's inverse as follows:

$$A^{-1} = \frac{1}{|A|} \begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}, \tag{15}$$

where $|A| = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$ and the values A_{ij}, $1 \le i,j \le 3$, are given by:

$$\begin{aligned} A_{11} &= \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} \quad A_{12} = -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} \quad A_{13} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ A_{21} &= -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} \quad A_{22} = \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} \quad A_{23} = -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ A_{31} &= \begin{vmatrix} a_{21} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \quad A_{32} = -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \quad A_{33} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}. \end{aligned}$$

The protocol that produces an inverse of a matrix A in an additively split form is given in Figure 5.

Complexity Analysis: In the matrix inversion protocol, step 1 requires 9 secure scalar products, step 2 requires one secure scalar product, step 3 requires 9 secure split divisions. Therefore, the protocol needs $O(1)$ secure scalar products and $O(1)$ secure divisions.

Security: Correctness of the protocol follows from Equation (15), which it faithfully implements. As long as the split multiplication protocol and the split division protocol are secure, by Lemma 1, the matrix inversion protocol is secure, i.e., does not reveal any private information.

Input: Supplier and Retailer input a 3×3 matrix A in additively split form, i.e., Supplier has $A^{(s)}$ and Retailer has $A^{(r)}$ such that $A^{(s)} + A^{(r)} = A$.

Output: Supplier learns $B^{(s)}$ and Retailer learns $B^{(r)}$ such that $B^{(s)} + B^{(r)} = A^{-1}$.

Protocol Steps:

1. For each $i \in \{1,2,3\}$ and $j \in \{1,2,3\}$, Supplier and Retailer run a split scalar product protocol to compute A_{ij} in additively split fashion.
2. Supplier and Retailer run another split scalar product protocol to compute $|A| = a_{11}A_{11} - a_{12}A_{12} + a_{13}A_{13}$ in additively split fashion.
3. For each $i \in \{1,2,3\}$ and $j \in \{1,2,3\}$, Supplier and Retailer run a split division protocol to compute $b_{ij} = A_{ji}/|A|$ according to Equation (15).

Figure 5 **Secure 3×3 matrix inversion.**

C.6. Secure Square Root

In some of our protocols we need the ability to compute the square root of a split value. To do this, we use a Taylor series for square root. Recall that the Taylor series is:

$$\sqrt{1+x} = \sum_{i=0}^{\infty} \frac{x^n \prod_{j=0}^{n} (\frac{1}{2} - j)}{n!}$$

Clearly, we can compute square root with the secure division, multiplication, and addition operations.

Complexity Analysis: The secure square root protocol requires $O(1)$ rounds, communication, and computation.

C.7. Secure Comparison

To compare two values we propose the usage of the technique described by Frikken and Atallah (2004), for comparing modularly additively split values. This scheme requires that the modulus be twice as large as the largest value that the unsplit data can take. This scheme requires communication and computation linear in the number of bits required to represent the modulus being used. However, protocols were also introduced to reduce the communication and computation to the number of bits required to represent the unsplit value.

Complexity Analysis: The above approaches require a constant number of rounds of interaction. The communication and computation are also $O(1)$.

C.8. Secure Binary Search

In some of our protocols we need the ability to compute x^* such that $f(x^*) = 0$, where f is a monotone function and can only be computed jointly by Supplier and Retailer. This can be done by binary search. Suppose x is in the range of $[s,t]$. Let k be the number of binary-search steps and $u = (t-s)/2^{k+1}$ be a precision number. The secure binary-search protocol is described in Figure 6. After the binary-search protocol, we ensure that x is close enough to x^* where $f(x^*) = 0$, i.e., it is guaranteed that $x - u < x^* < x + u$.

In the above protocol, we use a cryptographic building block called *oblivious transfer*. In oblivious transfer, a sender has two messages M_0 and M_1, a receiver has a bit $b \in \{0,1\}$; in the end, the receiver obtains M_b

Input: f is a function that can be securely computed by Supplier and Retailer. Let s and t be the lower and upper bound of x respectively. And let x initially be $(s+t)/2$.

Output: x is additively split between Supplier and Retailer, such that $f(x) \approx 0$.

Protocol Steps: For step $i = 1, 2, \ldots, k$:

1. Supplier and Retailer jointly compute $f(x)$ in split fashion.
2. If $f(x) > 0$, decrease x by $2^{k+1-i}u$; otherwise, increase x by $2^{k+1-i}u$. The decision of whether to increase x or decease x should be oblivious to both Supplier and Retailer. This is achieved as follows:

(a) Supplier and Retailer run a secure comparison protocol to compare $f(x)$ to 0 with the result being XOR split. Supplier obtains $b^{(s)}$ and Retailer obtains $b^{(r)}$ such that $b^{(s)} \oplus b^{(r)} = 1$ if $f(x) > 0$, $b^{(s)} \oplus b^{(r)} = 0$ otherwise.

(b) Supplier sends her encrypted share of x to Retailer. Retailer computes $E(x)$ by adding his share.

(c) Retailer chooses a random number r and sets his new share of $x = -r$. Retailer then prepares two encryptions $e_0 = E(x + 2^{k+1-i}u + r)$ and $e_1 = E(x - 2^{k+1-i}u + r)$, and sets $\theta_0 = e_{0 \oplus b^{(r)}}$ and $\theta_1 = e_{1 \oplus b^{(r)}}$.

(d) Supplier and Retailer run an oblivious transfer protocol in which Retailer inputs his two encrypted items θ_0 and θ_1 and Supplier inputs $b^{(s)}$. In the end, Supplier obtains $\theta_{b^{(s)}} = e_{b^{(s)} \oplus b^{(r)}} = e_b$; that is, she learns $x + 2^{k+1-i}u + r$ if $f(z_1)$ is negative and learns $x - 2^{k+1-i}u + r$ is positive. Supplier and Retailer obtain the new x in split fashion.

Figure 6 Secure binary search protocol.

without anything else and the sender learns nothing. The oblivious transfer protocol (e.g., Naor and Pinkas 1999) requires $O(1)$ modular exponentiations.

Complexity Analysis: The above protocol takes k steps. Each step requires one secure comparison, one evaluation of $f(x)$, and one oblivious transfer. Let ℓ be the number of bits required to represent the $f(x)$, o be the number of modular exponentiations needed to compute $f(x)$, then each step requires $O(\ell + o)$ modular exponentiations. Thus, the overall protocol needs $O(k \cdot (\ell + o))$ modular exponentiations.

Security: By Lemma 1, the secure binary search protocol is secure as each intermediate result is additively split between Supplier and Retailer.

Appendix D: Secure Protocols

D.1. Secure Standard Deviation Protocol

Input: Supplier knows the σ_k^s's and Retailer knows the σ_k^r's, for all $k = 1, \ldots, i+L-t$; the value of σ_0 is known to both of them. The parameters $\hat{\theta}^s$ and $\hat{\theta}^r$ are available in additively split form, i.e., Supplier has $\hat{\theta}^{s(s)}$ and $\hat{\theta}^{r(s)}$ and Retailer has $\hat{\theta}^{s(r)}$ and $\hat{\theta}^{r(r)}$.

Output: Supplier and Retailer obtain $\hat{\sigma}_t^{(s)}$ and $\hat{\sigma}_t^{(r)}$, respectively, where $\hat{\sigma}_{t,[i,i+L]} = \hat{\sigma}_t^{(s)} + \hat{\sigma}_t^{(r)}$.

Protocol Steps:

1. Supplier and Retailer engage in secure multiplication protocol twice to compute $(\hat{\theta}^s)^2$ and $(\hat{\theta}^r)^2$, respectively, and obtain the results in additively split form.
2. Supplier locally computes $v_s^{(s)} = \sum_{j=i}^{i+L} \sum_{k=1}^{j-t} (\sigma_k^s)^2$, and Retailer correspondingly computes $v_r^{(r)} = \sum_{j=i}^{i+L} \sum_{k=1}^{j-t} (\sigma_k^r)^2$. Supplier sets his value of $v_r^{(s)}$ to 0, and similarly Retailer sets $v_s^{(r)}$ to 0. Note that these computations should be performed only once (since the summations do not change),and therefore are pre-computed.
3. Supplier and Retailer execute secure multiplication protocol to compute $u_s = (\hat{\theta}^s)^2 \cdot v_s$, where Supplier inputs her share of $(\hat{\theta}^s)^2$ and $v_s^{(s)}$ and obtains $u_s^{(s)}$ and Retailer inputs his share of $(\hat{\theta}^r)^2$ and $v_s^{(r)}$ and obtains $u_s^{(r)}$ ($u_s = u_s^{(s)} + u_s^{(r)}$ and $v_s = v_s^{(s)} + v_s^{(r)}$). Similarly, they compute $u_r = (\hat{\theta}^r)^2 \cdot v_r$ and obtain the result in additively split form.
4. Supplier sets $w^{(s)} = (L+1)\sigma_0^2 + u_s^{(s)} + u_r^{(s)}$; Retailer sets $w^{(r)} = u_s^{(r)} + u_r^{(r)}$. Now the value of $w = w^{(s)} + w^{(r)}$ corresponds to the composite value under the square root in Equation (6).
5. Supplier and Retailer engage in a secure square root protocol providing $w^{(s)}$ and $w^{(r)}$, respectively, as their inputs. As a result, they obtain $\hat{\sigma}_{t,[i,i+L]}$ in additively split form.

Figure 7 Secure standard deviation of demand forecasting protocol.

D.2. Secure protocol for computing $\Phi(z)$

Input: The value of z is additively split between Supplier and Retailer.

Output: Supplier and Retailer additively share $\Phi(z)$.

Protocol Steps:

1. Supplier sends to Retailer $E(z^{(s)})$, the encryption of her share using her public key.
2. Retailer computes $E(z) = E(z^{(s)}) \cdot E(z^{(r)})$, and runs secure polynomial evaluation on z, and obtains $E(\Phi(z))$.
3. Retailer chooses a random r and sends to Supplier $E(\Phi(z) - r)$. In the end, $\Phi(z)$ is additively split between Supplier and Retailer.

Figure 8 Secure protocol for computing $\Phi(z)$.

Complexity Analysis: This protocol requires n modular exponentiations.

Security: By Lemma 1, this protocol is secure since each intermediate result is additively split between supplier and retailer.

D.3. Secure protocol for computing $\Phi(x, y, \rho)$

Input: The values of x, y, and ρ are additively split between Supplier and Retailer.

Output: $\Phi(x, y, \rho)$ is additively split between Supplier and Retailer.

Protocol Steps:

1. Supplier and Retailer compute $\Phi(x)$ and $\Phi(y)$ in additively split fashion using the technique in section 4.4.1. Then they compute $\Phi(x) \cdot \Phi(y)$ using split multiplication.
2. Supplier and Retailer compute $\phi(x)$ and $\phi(y)$ in split fashion using secure polynomial evaluation. Then they compute $\phi(x) \cdot \phi(y)$ using split multiplication.
3. Supplier and Retailer compute $\mathrm{He}_0(x), \mathrm{He}_1(x), \ldots, \mathrm{He}_n(x)$ in split fashion using secure polynomial evaluation. Note that the result of $\mathrm{He}_i(x)$ can be used to compute $\mathrm{He}_{i+1}(x)$.
4. Similarly, Supplier and Retailer compute $\mathrm{He}_0(y), \mathrm{He}_1(y), \ldots, \mathrm{He}_n(y)$ in split fashion using secure polynomial evaluation.
5. For $i = 0, \ldots, n$, Supplier and Retailer compute $\mathrm{He}_i(x)\mathrm{He}_i(y)\rho^{i+1}$ in split fashion.
6. Supplier and Retailer put the pieces together and obtain $\Phi(x; y, \rho)$ in split fashion.

Figure 9 **Secure protocol for computing $\Phi(x, y, \rho)$.**

Complexity Analysis: In this protocol, step 1 and 2 require $4n$ modular exponentiation, step 3 and 4 require $2n$ modular exponentiation, step 5 requires around $4n$ modular exponentiation. Therefore, overall the supplier and the retailer need to perform $10n$ modular exponentiation.

Security: By Lemma 1, the above protocol is secure as each intermediate result is additively split between the supplier and the retailer.

D.4. Secure replenishment protocol

Input: The values of y^{R*}, y^{S*}, IP_R, and IP_S are additively split between Supplier and Retailer.

Output: Supplier learns q_S and Retailer learns q_R, where $q_R = y^{R*} - IP_R$ and $q_S = y^{S*} - IP_S$.

Protocol Steps:

1. Supplier and Retailer compute q_S and q_R in split fashion using split subtraction twice.
2. Retailer sends $q_S^{(r)}$ to Supplier, Supplier obtains $q_S = q_S^{(s)} + q_S^{(r)}$.
3. Supplier sends a cryptographic commitment of $q_R^{(s)}$ (using any suitable commitment scheme) to Retailer (at current time t). At time $t + L_R$, Retailer obtains $q_R^{(s)}$ from Supplier and verifies that it corresponds to the committed value at time t. Supplier sets $q_R = q_R^{(r)} + q_R^{(s)}$.

Figure 10 **Secure replenishment protocol.**

Complexity Analysis: The cost of the secure replenishment protocol is the complexity of exchanging $q_S^{(r)}$ and $q_R^{(s)}$.

Security: By Lemma 1, the above protocol does not reveal private information as long as modular arithmetic is used.

D.5. Secure protocol for computing parameter β

Input: Supplier inputs private h^S and p^S and Retailer inputs private h^R and p^R.

Output: Supplier and Retailer additively share the value of β.

Protocol Steps:

1. Retailer sets $a^{(r)} = h^R$ and $b^{(r)} = p^R$, Supplier sets $a^{(s)} = -h^S$ and $b^{(s)} = 0$, and they engage in a secure multiplication protocol on the values of a and b. Let the resulting value be denoted as c, i.e., $c^{(r)} + c^{(s)} = c = a \cdot b$.
2. Retailer sets $d^{(r)} = h^R + p^R$, Supplier sets $d^{(s)} = p^S$, and they engage in a secure division protocol on the values of c and d. Let e denote the result of division, i.e., $e^r + e^{(s)} = e = \frac{c}{d}$.
3. Retailer sets $\beta^{(r)} = -e^{(r)}$ and Supplier sets $\beta^{(s)} = -h^S - e^{(s)}$.

Figure 11 **Secure protocol for computing parameter β.**

Complexity Analysis: The complexity of this protocol is dominated by one execution of the multiplication protocol and one execution of the division protocol.

Security: By Lemma 1.

D.6. Secure protocol for computing γ

Input: Supplier and Retailer hold the values of $\Phi(z_2^*)$ and $\Phi(z_2^*; z_3^*; \rho)$ in additively split form. Supplier also inputs h^S and p^S.

Output: Supplier and Retailer obtain γ in additively split form.

Protocol Steps:

1. Supplier sets $a^{(s)} = h^S$, Retailer sets $a^{(r)} = 0$, and they execute a secure multiplication protocol on values of a and $\Phi(z_2^*)$, and obtain $b = h_2 \cdot \Phi(z_2^*)$ in split form. Supplier and Retailer then engage a secure subtraction protocol to compute $c = a - b$ and obtain c in additively split form. Note that $c = h_2(1 - \Phi(z_2^*))$.
2. Supplier sets $d^{(s)} = p^S$, Retailer sets $d^{(r)} = 0$. Observe that $d = p_2$. Supplier and Retailer run a secure subtraction protocol to compute $\Phi(z_2^*) - \Phi(z_2^*; z_3^*; \rho)$, and then engage a secure multiplication protocol to compute $e = p_2(\Phi(z_2^*) - \Phi(z_2^*; z_3^*; \rho))$.
3. Supplier and Retailer run a secure subtraction protocol to compute $\gamma = c - e$. As a result, Supplier has $\gamma^{(s)}$ and Retailer has $\gamma^{(r)}$.

Figure 12 **Secure protocol for computing γ.**

Complexity Analysis: The computational and communication complexity of this protocol is dominated by secure polynomial evaluation and one execution of the multiplication protocol.

Security: By Lemma 1, the above protocol does not reveal private information as long as the underlying blocks are secure.

D.7. Secure protocol for computing transfer payment

Input: Supplier and Retailer hold the values of $\hat{\mu}_{t,[t,t+L_R]}$, $\hat{\mu}_{t,[t,t+L_S+L_R]}$, β, and γ in additively split form. The value of α is known to both of them.

Output: Supplier and Retailer learn the value of transfer payment TP_t.

Protocol Steps:

1. Supplier and Retailer execute a secure multiplication protocol twice: once to compute $a = \beta \cdot \hat{\mu}_{t,[t,t+L_R]}$, and another time to compute $b = \gamma \cdot \hat{\mu}_{t,[t,t+L_R+L_S]}$.
2. They exchange their shares of a and b (possible using a fair exchange protocol) and set $TR = \alpha + a + b$.

Figure 13 **Secure protocol for computing transfer payment.**

Complexity Analysis: The computational and communication complexity of this protocol is dominated by two executions of secure multiplication.

Security: By Lemma 1.

Appendix E: Known Results for Circuit Simulation

We summarize known results for secure circuit evaluation in this section. The main part of this is Theorem 6. As this paper is based on two party computations, we summarize the results for two party computation. For an excellent survey of this work see Goldreich (2004).

THEOREM 6. *Given a boolean circuit of binary gates with m gates, n inputs, and depth d, there exist techniques for evaluating the circuit in a private manner for non-adaptive computationally-bounded adversaries in the following cases:*

1. **Passive(semi-honest)***: (Yao 1986) showed that this can be done with n 1-out-of-2 Oblivious Transfers (OT), $O(m)$ communication, $O(m)$ evaluations of a pseudorandom function (such as AES), and $O(1)$ rounds.*
2. **Malicious:** *(Katz and Ostrovsky 2004) showed that this circuit can be evaluated in the malicious model with only 5 rounds of interaction assuming that early termination is not considered a violation of security. Furthermore, the computation and communication are polynomial in the size of the circuit.*

The above theorem states that any function that is efficiently evaluable (i.e., is in P) can be evaluated securely with polynomial time computation and polynomial size communication and a constant number of rounds. This holds for adversaries that are semi-honest or malicious. This is because for every function in P there is a polynomial size circuit that evaluates the function. There are a three remaining issues to discuss: (i) the notion of early termination, (ii) the practicality of these protocols, and (iii) the usage of circuits in the constructions.

First, as stated above, we do not consider early termination (i.e., one party stopping part way through the protocol) to be a violation of security. The primary reason for this is that it is impossible to provide protection against such deviation unless there is some third party helping with the computation. This follows from impossibility results of Byzantine Agreement (Feldman and Micali 1988).

The protocols in Theorem 6 for the semi-honest model are actually practical for some problems. In fact, there has recently been an implementation of these protocols which shows that it is useful for many problems (Malkhi et al. 2004). However, the protocol for the malicious model requires many expensive operations (such as zero-knowledge proofs) for each gate of the circuit. Thus while the round complexity of this scheme is quite good, it is unlikely that these techniques are tractable for many problems (including those in this paper).

The final remaining issue is the usage circuits. First, circuit complexity is not equivalent to computational complexity. One reason for this is that circuits have a size and a depth, and while this does not matter for the above mentioned secure protocol techniques, there are protocols that require the depth of the circuit. There is one other substantial difference, however, and that is that circuits do not have random access memory. Thus the circuit for indirectly indexing into a list of size n has $O(n)$ gates, where as this would be a single operation in computational complexity.

We now describe various binary circuit complexities:

1. *Adding two m-bit numbers*: Adding or subtracting two m-bit numbers can be done optimally using circuits of size $O(m)$ gates and depth $O(\log m)$ (Ofman 1963).

2. *Adding k m-bit numbers*: Requires $O(m \log k)$ gates and $O(\log k \log m)$ depth.

3. *Multiplying two m-bit numbers*: The practical circuits for multiplication have size $O(m^2)$ and depth $O(\log m)$. Although there are asymptotic improvements to these circuits, they come at the cost of huge constant factors; the asymptotically best of them (and the worst in terms of having impractically large constant factors) is a circuit of size $O(m \log m \log\log m)$ and depth $O(\log m)$ derived from the textbook Schoenhage-Strassen integer multiplication algorithm (Schoehage and Strassen 1971, Aho et al. 1974) (which is itself of mainly theoretical interest, and not used in practice).

4. *Integer division of two m-bit numbers*: Just as for multiplication, the practical circuits for multiplication have size $O(m^2)$ and depth $O(\log m)$. However, unlike multiplication, the "impractical" (but asymptotically better) circuits achieve either $O(m \log m \log\log m)$ using the Schoehage-Strassen technique (Alt 1988, Beame et al. 1984) or depth $O(\log m)$ (Beame et. al 1984) but not simultaneously. The division circuits that come closest to achieving "close to simultaneity" of these size and depth bounds have size $O(m \log m \log\log m)$ and depth $O(\log m \log\log m)$ (Reif and Tate 1989).

5. *Comparing m bit numbers:* Comparisons of the form $\geq, \leq, >, <, =, \neq$ requires $O(m)$ gates and $O(\log m)$ depth.

6. *Sorting n numbers with m bits:* A practical sorting network is one that implements Batcher's sort (Batcher 1968) and has size $O(mn \log^2 n)$ and depth $O(\log m \log^2 n)$. An asymptotically better (but impractical due to its large constant factors) sorting network is the AKS one (Ajtai et al. 1983) that has size $O(mn \log n)$ and depth $O(\log m \log n)$.