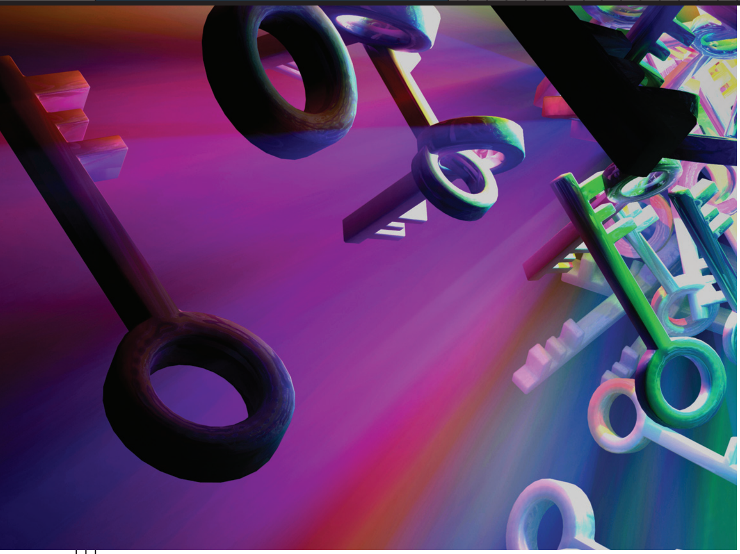

Secure Collaborative Supply-Chain Management

Florian Kerschbaum and Axel Schröpfer
SAP Research Karlsruhe, Germany

Antonio Zilli
University of Salento, Italy

Richard Pibernik
EBS Business School, Germany

Octavian Catrina
University of Mannheim, Germany

Sebastiaan de Hoogh and Berry Schoenmakers
Eindhoven University of Technology, Netherlands

Stelvio Cimato and Ernesto Damiani
Università degli studi di Milano, Italy

The SecureSCM project demonstrates the practical applicability of secure multiparty computation to online business collaboration. A prototype supply-chain management system protects the confidentiality of private data while rapidly adapting to changing business needs.

The Internet's ubiquity and the advent of cloud computing enable new forms of collaboration, while services such as Facebook and Twitter let people communicate and interact in novel ways. These trends impact businesses as well as consumers, but can companies safely operate in this new world?

It is well known that effective supply-chain collaboration reduces inefficiencies such as excess costs, inadequate service levels, and environmental pollution. Nevertheless, companies often do not collaborate due to a lack of trust: they fear that data they share could be used outside its intended purpose. For example, business partners need to know one another's production costs to optimally schedule warehousing and production, but they could also use such information to manipulate future price negotiations.

Researchers have long tried to reconcile these two conflicting goals of information sharing and protecting confidentiality. Cryptographers developed a theoretical solution more than 25 years ago: *secure multiparty computation*. According to this concept, multiple parties compute a function on their joint confidential input. Each party learns the result but nothing else about the other parties' inputs. The "Secure Multiparty Computation" sidebar describes this subfield of cryptography in more detail and provides an example of its use.

The adoption of secure multiparty computation for real-world business problems still faces four key obstacles. First, researchers must analyze observed inefficiencies in the supply chain and derive a formula that accurately addresses them. Second, researchers must develop a secure protocol to implement this computation. Designing such protocols is a challenging task, as their efficiency is low and performance optimizations often require manual security verification. Third, researchers must carry out practical experiments to evaluate the solution's computational performance. Finally, there must be a risk assessment to determine how adoption will impact the economic environment.

CASE STUDY: AEROENGINE SUPPLY CHAIN

The European research project SecureSCM (www.securescm.org) has completed all four of these steps in a case study using an aeroengine parts manufacturer's real supply chain.

Published by the IEEE Computer Society

0018-9162/11/$26.00 © 2011 IEEE

SECURE MULTIPARTY COMPUTATION

Secure multiparty computation is a cryptographic technique that allows several parties to compute any function without any party having to disclose its input to another. Each party's input remains private to that party, but the result can be made available to all, or only to a subset, of the other parties.

Consider the following simple example. Alice, Bob, and Charlie each have a number x_A, x_B, and x_C, respectively, as private input and want to compute $x = x_A + x_B + x_C$. However, they do not want to disclose their private data to one another. Alice chooses a random number r and privately sends $r + x_A$ to Bob. Bob adds his input and privately sends $r + x_A + x_B$ to Charlie. Charlie does the same with his input and sends $r + x_A + x_B + x_C$ back to Alice. Alice recalls r, subtracts it from the received value $r + x_A + x_B + x_C$, and announces $x = x_A + x_B + x_C$. Observing the messages exchanged between Alice, Bob, and Charlie, it is easy to see that none of them learns the input of the other parties—for example, Alice's random choice r blinds the value she sends to Bob, and Alice does not see the message (including $r + x_A + x_B$) Bob sends to Charlie.

Cryptography research has proven that such a protocol exists for any efficiently computable function for any finite number of parties. It is important to note that secure computation does not rely on a trusted third party to perform computations and ensure data privacy, but is based on decentralized computation implemented through cryptographic protocols.

In the aerospace industry, coordination among supply-chain actors is critical to success. During program execution, many parts flow from one stage of the supply chain to the next, and at each stage these parts are assembled into a new component and continue their flow. This movement of products forms a pyramidal supply chain: at the apex is the *program leader firm*; below and directly connected to it are three or more *prime partners*, each in charge of a main section of the aircraft such as the engine, airframe, and avionics.

Prime partners manage numerous suppliers that provide the most important component modules. For example, in the engine section of the supply chain, the prime partner decouples the engine into the low-pressure turbine, the high-pressure turbine, the combustion chamber, the gearbox, and so on, and assigns the detailed design and manufacturing of these modules to different suppliers. As in the previous stage, the module suppliers require their own parts and raw materials. Given this complex network of firms and relationships, synchronizing product flow is essential to efficiently managing stock.

Downstream companies leading the supply chain are linked by long-term partnerships because designing and assembling engines and other complete aircraft parts require both highly specialized skills and knowledge and considerable research investment. In contrast, upstream firms that produce nonaerospace-specific products exploit short-term business opportunities.

The SecureSCM case study focused on the supply chain of the shroud nozzle, which is part of the engine's low-pressure turbine. Figure 1 shows the supply chain's structure, which includes the shroud nozzle manufacturer, an external manufacturer that is an outsourcing partner, and five component suppliers. The final customer is the engine manufacturer.

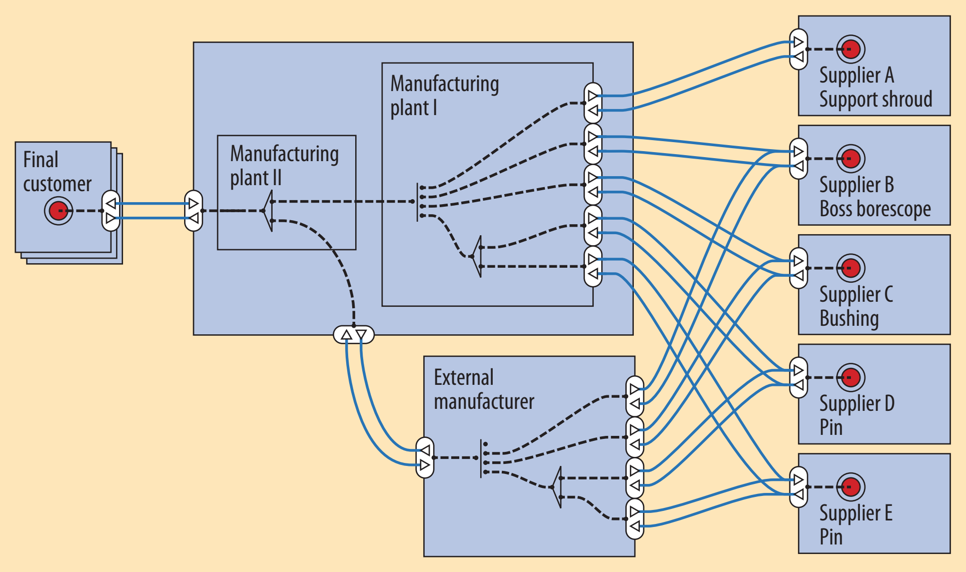

Figure 1. Supply chain of the shroud nozzle, part of an aeroengine's low-pressure turbine. The supply chain includes the shroud nozzle manufacturer, an external manufacturer that is an outsourcing partner, and multiple component suppliers. The final customer is the engine manufacturer.

We interviewed the various actors in the supply chain, who provided several reasons for protecting the confidentiality of their data. The manufacturer wanted to protect short-term market demand and its capacity by limiting the external manufacturer's awareness of its business power. The suppliers sought to protect their capacity and maintain low costs by not causing the manufacturer to look to other potential suppliers. They also wanted to conceal their production, warehousing, and shipping costs and their inventory status from the manufacturers to protect their business and production strategy. Moreover, and somewhat orthogonal to trust, all of the parties desired to protect their confidential data to meet their partners'

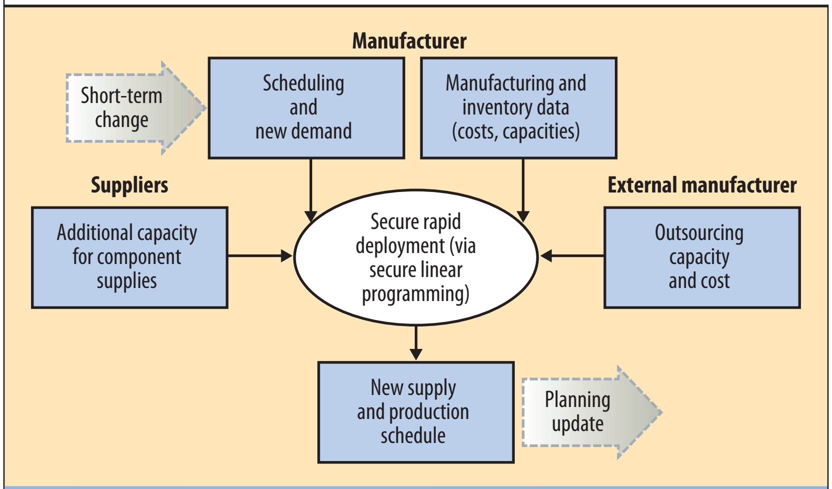
Manufacturer
Short-term change
Scheduling and new demand
Manufacturing and inventory data (costs, capacities)
Suppliers
External manufacturer
Additional capacity for component supplies
Secure rapid deployment (via secure linear programming)
Outsourcing capacity and cost
New supply and production schedule
Planning update

Figure 2. SecureSCM's rapid deployment system quickly and cost-effectively adapts manufacturing production schedules to component suppliers' short-term capacity information.

expectations: a firm unable to protect its confidential data is not a good partner.

PROBLEM ANALYSIS

In standard operating mode, the shroud nozzle manufacturer procures four components from different suppliers that are delivered to a raw materials and component warehouse. Depending on the production schedule, the manufacturer transports these components to a facility where they are preassembled. It then ships the preassembled components to a different warehouse, from which they are delivered to a different manufacturing plant for final assembly, testing, and quality control.

In general, this part of the supply chain is characterized by a stable production schedule that the program leader determines well in advance. Sometimes, however, the program leader must, on short notice, order spare parts, which can lead to an upsurge in demand that the current supply chain cannot easily handle. The manufacturer has no means to quickly change production schedules and obtain extra parts from its standard suppliers. To cover this additional demand and satisfy its customers' service-level requirements, the manufacturer employs, on a contract basis, an external manufacturer that procures the necessary components, assembles the shroud nozzle, and delivers it to the manufacturer.

The outsourcing partner charges a premium price that exceeds the standard supply-chain cost by an order of magnitude. The manufacturer would therefore like to implement a system that enables it to quickly reschedule production to accommodate short-term demand peaks. However, the manufacturer has no information about the suppliers' capacity and thus cannot quickly place orders with them. One reason for this is that the suppliers are not willing to share detailed real-time information about their capacity utilization and ability to provide additional components.

To remedy this problem, SecureSCM proposed a rapid deployment system, depicted in Figure 2, that utilizes the suppliers' short-term capacity data to quickly adapt manufacturing production schedules. The objective is to minimize total cost—including supply, manufacturing, inventory, and backordering costs—while respecting all relevant capacity constraints and fulfilling overall customer demand.

We formulated this optimization problem using *linear programming*, which minimizes an objective function subject to linear equality and inequality constraints. The objective function is the sum of all supply-chain costs; the equality constraints model the flow of goods, and the inequality constraints implement the capacity constraints. To protect sensitive and private data, the protocol relies on secure multiparty computation.

SECURE MULTIPARTY COMPUTATION PROTOCOL

Secure multiparty computation uses generic cryptographic protocols to protect data privacy and deliver correct outputs in the presence of an adversary—which can be a single entity or a coalition of parties—that controls communications. There are two basic models: in the *passive adversary* model, semi-honest parties reveal some secret data but otherwise continue to follow the protocol; in the *active adversary* model, parties deviate from the protocol to carry out malicious attacks.

Generic protocols for multiparty computation are roughly based on either *homomorphic encryption* or linear *secret sharing*. Both schemes represent a computable function as a Boolean or arithmetic circuit over a finite field or ring; linear functions of secret values are locally computed, while multiplication requires interaction between parties. With homomorphic encryption, the parties encrypt their secret inputs. With secret sharing, the parties randomly split their secret inputs into shares and give one share to each of the other parties. If enough parties combine their shares, they can reconstruct the secret; otherwise, they learn nothing about it. We used protocols based on linear secret sharing in the case study because they are more efficient than those based on homomorphic encryption.

The performance of generic protocols is insufficient for secure linear programming and similar large-scale applications. Our first task, therefore, was to develop a collection of efficient protocols based on linear secret sharing for operations on primitive data types and arrays.[1,2]

Current protocols provide all basic operations with binary, signed integer, and signed fixed-point data types as well as secret indexing for vectors and matrices. Floating-point protocols are still too complex for large-scale applications. We constructed the protocols using a small set of building blocks based on the same security model and cryptographic techniques, thus simplifying analysis and application development.

The protocols' effectiveness is primarily determined by the amount of exchanged data (communication complexity) and the number of sequential interactions (round complexity). Our pragmatic approach focused on achieving maximum efficiency while meeting rigorous but realistic security requirements. We obtained important performance and scalability improvements by precomputing secret random values and optimizing the building blocks for parallel computation of large batches of operations. In particular, we reduced communication complexity by encoding data in small fields that match the size of the data types; this is possible for protocols based on secret sharing because privacy does not rely on computational assumptions (as it does in the case of homomorphic encryption), and share conversions between different fields are efficient. Furthermore, families of building blocks that exploit different tradeoffs allow the application developer to select the variant that best suits the algorithm and execution environment.

Research on solving linear programs has produced two types of iterative algorithms: *interior-point* methods have polynomial complexity, but *simplex* techniques can require exponentially many iterations in the worst case. However, this is rarely an issue, and simplex algorithms are more suitable for secure computation due to their simpler iterations.

Previous secure linear programming protocols[3] solved only linear programs for which the null solution is feasible—that is, it does not violate any constraint. Because this condition is generally not satisfied, we developed a two-phase simplex protocol for solving any linear program. The inputs are shared-secret values representing the constraints (capacities) and the objective function (costs), while the outputs are the shared-secret final simplex data and a public value indicating the termination condition: optimal, unbounded, or infeasible. The termination condition proved useful in the case study for debugging the supply-chain model. The simplex data structure is protected throughout the computation through secret sharing as well as *secret indexing*, a cryptographic method that hides the indexes of changes in the simplex data. To avoid exponentially many iterations, the protocol also reveals the number of iterations.

The type of simplex algorithm a secure multiparty computation protocol uses has an important impact on performance and scalability. The algorithm must take into account the complexity of the secure operations: arithmetic, comparison, and indexing. After evaluating different variants, we obtained the most efficient protocol with a simplex algorithm using fixed-point arithmetic.[4] This algorithm reduces communication complexity by using more compact data types, more efficient data structures, and fewer secure comparisons (one of the main performance bottlenecks). It is thus more suitable than other algorithms, including some variants that are superior for nonsecret data.

IMPLEMENTATION

Implementing secure multiparty computation protocols is difficult: the programmer must manage complicated cryptographic routines and handle many auxiliary tasks, such as communication. To address this challenge, we designed L1, a small programming language that makes it easier to rapidly implement and evaluate such protocols during a project cycle.[5]

The type of simplex algorithm a secure multiparty computation protocol uses has an important impact on performance and scalability.

L1 is a procedural language similar to C or Java that also supports parallel execution. It offers primitives for cryptographic operations, such as homomorphic encryption and secret sharing, as well as for synchronous and asynchronous communication. All communication occurs over secure and authenticated channels using the Secure Sockets Layer protocol. The programmer can combine different secure computation protocols and even partially reveal intermediate results, if deemed safe.

An L1 program represents the code run by a specific player, instead of the function implemented by all players, but also incorporates *player-specific code*, such that the same program can be used for all parties. Consider the following example:

```
1    send(1, “share_” + id(), share);
2    1: {
3      for (int32 i=1; i<=players; i++)
4        shares[i]=readInt(“share_”+i);
5      output(reconstruct(shares));
6    }
```

Line 1 sends all parties' shares to the first party. The player-specific code in lines 2-6 tells the first party to receive all shares and to reconstruct and output the secret.

After implementing a protocol in L1, we evaluate it using a testbed. Because protocol performance depends on the computer system as well as the network characteristics, we simulate two scenarios. In one scenario, the servers are connected via a local area network (LAN). This could be the case if the parties are cohosts in a common data center. In the other, more common, scenario, the servers are connected via a wide area network (WAN). In this case, we reduce the bandwidth and increase network latency.

For our case study, we connected eight servers representing the various supply-chain actors. Each server had a dual-core 2.6-GHz AMD Opteron 885 processor and 4 Gbytes of RAM. For brevity, we report here only the results of our best protocol variant. For the LAN scenario, with a bandwidth of 1,000 Mbits per second and a latency of 1 millisecond, the protocol runtime was 41 minutes and 4 seconds. For the WAN scenario, with a bandwidth of 10 Mbps and a latency of 20 ms, the runtime was 1 hour, 1 minute, and 38 seconds—a reasonable performance penalty for business purposes.

Secure multiparty computation can facilitate other forms of collaboration besides supply-chain management.

RISK ASSESSMENT

Secure multiparty computation prevents supply-chain partners from learning one another's input, but it does not prevent them from altering their own. Partners who cannot reconcile their objectives with the common good might be tempted to distort the optimization to yield results more favorable to themselves. It is therefore necessary to assess the risk of any conflicts of interest.

Supply-chain actors are characterized by strategic interdependence—the payoff (typically in revenue) for any action depends on what the other actors do. All actors attempt to maximize their own payoff vis-à-vis others' response strategies, and equilibrium is attained when no actors would obtain a higher payoff by unilaterally deviating from their course.

In such a scenario, it is possible to compute a risk profile for each actor based on several factors that could contribute to selfish behavior:

- *fairness*—the actor's perception of the supply chain's fairness, modeled as the distance between its fairness value (the average of the contributions that each actor gives to all possible coalition configurations) and the actor's actual profit obtained from the chain;
- *payoff*—the actor's perception of the potential outcome of a unilateral action; and
- *context*—the actor's role and position within the supply chain.

We translate these three factors into a probability value and then compute a risk profile for each actor in the supply chain by multiplying the probability of a unilateral attack by its impact. It is then an easy matter to set up an incentive scheme that defines the way benefits and penalties devolve to the actors, motivates collaborative behavior, and punishes disruptive behavior.

Researchers can use a prototype supply-chain risk simulator that emerged from the SecureSCM project[6] to simulate a particular attack in different scenarios and predict the supply chain's response (www.mathworks.com/matlabcentral).

SecureSCM demonstrates the practical applicability of secure multiparty computation to online business collaboration. Using part of an aeroengine manufacturer's supply chain, we designed and implemented a collaborative planning system that protects the confidentiality of private data while rapidly adapting to changing business needs. To our best knowledge, this is the first system to run secure multiparty protocols on such a large problem instance.

While data confidentiality is a clear prerequisite of online business collaboration, it is not sufficient. Because partners must put their trust in electronic systems, user acceptance is a critical issue. Secure multiparty computation raises the risk of selfish behavior, and all parties must feel confident that no one can game the system. Nevertheless, the companies we worked with in the case study have expressed an interest in ultimately adopting the system, which we continue to improve.

Secure multiparty computation can facilitate other forms of collaboration besides supply-chain management. Whenever parties working together are reluctant to exchange data due to confidentiality or privacy concerns, this approach can be beneficial. For example, law-enforcement agencies have successfully used secure multiparty computation in criminal investigations.[7]

References

1. O. Catrina and S. de Hoogh, "Improved Primitives for Secure Multiparty Integer Computation," *Proc. 7th Conf. Security and Cryptography for Networks* (SCN 10), LNCS 6280, Springer, 2010, pp. 182-199.
2. O. Catrina and A. Saxena, "Secure Computation with Fixed-Point Numbers," *Proc. 14th Int'l Conf. Financial Cryptography and Data Security* (FC 10), LNCS 6052, Springer, 2010, pp. 35-50.
3. T. Toft, "Solving Linear Programs Using Multiparty Computation," *Proc. 13th Int'l Conf. Financial Cryptography and Data Security* (FC 09), LNCS 5629, Springer, 2009, pp. 90-107.

4. O. Catrina and S. de Hoogh, “Secure Multiparty Linear Programming Using Fixed-Point Arithmetic,” *Proc. 15th European Symp. Research in Computer Security* (ESORICS 10), LNCS 6345, Springer, 2010, pp. 134-150.
5. A. Schröpfer, F. Kerschbaum, and G. Müller, “L1—An Intermediate Language for Mixed-Protocol Secure Computation,” *Proc. 35th Ann. IEEE Computer Software and Applications Conf.* (COMPSAC 11), IEEE CS Press, 2011; http://eprint.iacr.org/2010/578.pdf.
6. M. Anisetti et al., “Using Incentive Schemes to Alleviate Supply Chain Risks,” *Proc. Int’l Conf. Management of Emergent Digital EcoSystems* (MEDES 10), ACM Press, 2010, pp. 221-228.
7. F. Kerschbaum and A. Schaad, “Privacy-Preserving Social Network Analysis for Criminal Investigations,” *Proc. 7th ACM Workshop on Privacy in the Electronic Society* (WPES 08), ACM Press, 2008, pp. 9-14.

Florian Kerschbaum *is a research architect at SAP Research Karlsruhe, Germany. His research interests include security and privacy in distributed applications and applied cryptography. Kerschbaum received a PhD in computer science from Karlsruhe Institute of Technology. Contact him at florian.kerschbaum@sap.com.*

Axel Schröpfer *is a PhD student at SAP Research Karlsruhe. His research interests include applied cryptography and programming languages. Schröpfer received an MSc in computer science from Karlsruhe University of Applied Science. Contact him at axel.schroepfer@sap.com.*

Antonio Zilli *is a research fellow in the Center for Business Innovation at the University of Salento, Italy. His research focuses on secure knowledge-management practices and aerospace value-network systems. Zilli received a degree in physics from the University of Lecce, Italy. Contact him at zilli@unisalento.it.*

Richard Pibernik *is a professor of supply-chain management at EBS Business School in Wiesbaden, Germany, and an adjunct professor at the Zaragoza Logistics Center, Spain. His research interests include supply-chain collaboration, demand fulfillment, and supply-chain risk management. Pibernik received a PhD in business administration from Goethe University, Frankfurt, Germany. Contact him at richard.pibernik@ebs.edu.*

Octavian Catrina *is a senior research associate in the Department of Mathematics and Computer Science at the University of Mannheim, Germany. His research interests include cryptographic protocols, secure multiparty computation, and security of distributed systems. Catrina received a PhD in telecommunications from Politehnica University Bucharest, Romania. Contact him at octavian.catrina@uni-mannheim.de.*

Sebastiaan de Hoogh *is a PhD student at Eindhoven University of Technology, Netherlands. His research interests include applied cryptography and secure multiparty computation. De Hoogh received an MSc in mathematics from Eindhoven University of Technology. Contact him at s.j.a.d.hoogh@tue.nl.*

Berry Schoenmakers *is an associate professor of cryptography at Eindhoven University of Technology and an advisor on cryptography at Philips Research Labs. His research focuses on privacy-protecting protocols as applied to electronic voting, electronic payment systems, and secure multiparty computation, and on implementing systems based on such protocols. Contact him at berry@win.tue.nl.*

Stelvio Cimato *is an assistant professor of computer technology at Università degli Studi di Milano, Italy. His research interests include cryptography, network security, and Web applications. Cimato received a PhD in computer science from Università di Bologna, Italy. Contact him at stelvio.cimato@unimi.it.*

Ernesto Damiani *is a professor and heads the School of Computer Science at the Università degli Studi di Milano. His research interests include knowledge extraction and processing, system and network security, and software process engineering. Damiani received a PhD in computer science from the Università degli Studi di Milano. He is a senior member of IEEE and an ACM Distinguished Scientist. Contact him at ernesto.damiani@unimi.it.*