RWTH Aachen

Department of Computer Science
Technical Report

SMC-MuSe: A Framework for Secure Multi-Party Computation on MultiSets

Georg Neugebauer, Ulrike Meyer

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2012-16

RWTH Aachen · Department of Computer Science · December 2012

SMC-MuSe: A Framework for Secure Multi-Party Computation on MultiSets

Georg Neugebauer, Ulrike Meyer

Research Group IT-Security
RWTH Aachen, Germany
Email: {neugebauer, meyer}@umic.rwth-aachen.de

Abstract. Secure Multi-Party Computation (SMC) offers a theoretically well-founded way to enable applications that preserve their users' privacy. However, the practical use of SMC has often been questioned in the past. This is partly due to the fact that the system assumptions made in theory are hard to meet in practice and partly due to the potentially very high overhead general purpose SMC frameworks induce on clients. In this report, we aim at bringing SMC closer to regular Internet users. We introduce SMC-MuSe, a framework for **S**ecure **M**ulti-Party **C**omputation on **Mu**lti**Se**ts. SMC-MuSe is targeted at the efficient implementation of specific interesting functions rather then on computing arbitrary ones. It is generic in the sense that it allows to compute any composition of privacy-preserving set intersections, unions, and reductions on multisets. The system model used in SMC-MuSe is kept close to the one assumed in theory and supports asynchronous communications, resilient SMC computations, and fully automated key management.

1 Introduction

Today's Internet is full of applications via which users share potentially private information with their friends and business partners (e.g. Doodle, Google Calendar, or Flickr). As a side effect this information is shared with an (implicitly) trusted service provider. The majority of users may still be willing to trade the free service use for making their information available to the server. Recently, however, privacy concerns of users are rising [EY,SUN]. Users gradually become more suspicious with respect to the use of their (personal) information by service providers.

Secure Multi-Party Computation (SMC) offers a theoretically well-founded way to resolve this issue. In general it allows multiple parties to compute a function on their individual private inputs in a distributed fashion without revealing anything but the output of the function to each other or any server. SMC protocols have been widely studied in theory in the past. These theoretical studies typically assume that (1) all participating parties can directly communicate with each other, (2) secure channels between each pair of parties exist and (3) any other keying material required for the SMC protocol in question is predistributed. These assumptions are hard to achieve in practice. In addition, the practical use of SMC has often been questioned due to the large communication and computation overhead it puts on the clients. As a result, there exist two main directions in SMC. The first direction aims at the computation of arbitrary functions even if this may lead to some functions not being computable within a reasonable time frame. The second direction aims at developing protocols to more efficiently compute some specific interesting functions. Examples for such functions are (multi-)set operations.

Recently, several frameworks have been proposed that aim to bring SMC closer to practical applications [BDNP08,BLW08,BSMD10,DGKN09]. All of these frameworks aim at the computation of arbitrary functions, some with the restriction on Z_p or Z_{2^n}. The frameworks [BDNP08,BLW08,BSMD10] introduce multiple trusted servers that carry out all SMC-related computations on encrypted inputs on behalf of the users. These servers are assumed to be able to synchronously communicate with each other. The problem of establishing secure channels (2) and distributing additional keying material (3) is deferred from the clients to the trusted servers. The computing servers need to be trusted by the users to correctly follow the protocol and do nothing but the required computations. To our knowledge only two of the existing frameworks have been evaluated in the context of a realistic application [BSMD10,DGKN09] and none of them in the context of an end-user application for regular Internet users.

In this report we propose SMC-MuSe[1], a new framework for secure multi-party computation in which all SMC-related computations are carried out on the clients themselves instead of on a server. The framework is targeted at the efficient computation of arbitrary compositions of intersections, unions, and element reductions of private multiset inputs. As such it particularly enables the implementation of e-Voting schemes, auctions, distributed network monitoring, and other interesting applications [KS05,MNMW11]. SMC-MuSe offers a comprehensive support infrastructure to address the three assumptions made in theoretical SMC. In particular, SMC-MuSe introduces two (non-colluding) server components: one that is solely responsible for relaying messages between clients and thus addresses (1) and one that automatically generates and distributes keying material to the involved clients (3) and thus additionally automates the setup of the required secure channels (2).

We show the potential of SMC-MuSe by implementing the four Multi-Party Reconciliation on Ordered Sets (MPROS) protocols proposed in [NMW10]. These protocols allow multiple parties to find common inputs in their individually ordered input sets that maximize a common order on the intersection of their inputs. Each of the MPROS protocols is based on a composition of multiset operations and thus a canonical candidate for implementation in SMC-MuSe. Finally, we build a privacy-preserving scheduling application for regular Internet users on top of the MPROS protocols. We provide an extensive performance evaluation of our MPROS implementation and show that they are, e.g., well suited to support a scheduling application for a realistic number of parties and inputs. Like Doodle [DOO], this new scheduling application allows several parties to agree upon a common meeting time. However, unlike in Doodle, in our application each party can assign preferences to each one of its free time slots which are taken into account when the unbiasedly best meeting time is determined. Furthermore all parties keep their inputs private from each other and do not even reveal them to any server.

Outline: In Section 2 we briefly review the core components of SMC-MuSe, namely the privacy-preserving operations on multisets and the MPROS protocols. We present the design, implementation, and evaluation of our framework in Section 3 and 4. Section 5 discusses related work on privacy-preserving set

[1] SMC-MuSe in action: https://www.youtube.com/user/misterxyz42

operations, system model discussions, SMC frameworks, and privacy-preserving scheduling applications.

2 SMC on Multisets

In this section, we briefly describe the privacy-preserving set operations on multisets used as basic building blocks in SMC-MuSe. In addition, we sketch the four MPROS protocols as examples for more complex protocols that can be built on top of the multiset operations.

Assume there are n parties $P_1, ..., P_n$, each holding a private input multiset $S_i\,(1 \leq i \leq n)$ chosen from a common domain D, where each set element may occur more than once. The intersection ($\cap$) of two multisets contains elements in the multitude of the minimum of the multitudes of the two input sets. The union ($\cup$) of two multisets contains elements in the multitude of the sum of the multitudes of the two input sets. The element reduction by t (Rd_t) of a multiset contains elements with a multitude reduced by t.

Privacy-preserving protocols for the computation of these basic operations on multisets were first proposed by Kissner et al. [KS05]. In addition, they proofed that one can compute any function over multisets in a privacy-preserving way that can be expressed in the grammar

$$\Upsilon ::= S_i \mid Rd_t(\Upsilon) \mid \Upsilon \cap \Upsilon | S_i \cup \Upsilon | \Upsilon \cup S_i\,. \tag{1}$$

In SMC-MuSe we implemented these privacy-preserving multiset operations such that any function that can be expressed in Equation 1 can be computed with n parties and arbitrary input multisets S_i in SMC-MuSe.

Protocols for multi-party reconciliation on ordered sets (MPROS) were first described in [NMW10,NMW11]. The MPROS protocols involve n parties, where each party holds k distinct private inputs. Each party associates a *preference* (or *rank*) with each of its inputs such that his most preferred input is s_{i1} and is associated with a preference of k, his second most preferred input is s_{i2} and is associated with a preference of $k-1$, and so on until his k-th input element s_{ik} which is associated with a preference of 1. An example for three parties with three private inputs and preferences is illustrated in Part (1) of Figure 1.

In all MPROS protocols, input multisets are represented as the roots of polynomials. I.e. the multisets S_i are represented as polynomials $f_i(X) = \prod_{j=1}^{k}(X - s_{ij})^j$. All computations in the MPROS protocols are then performed on polynomials encrypted with a semantically secure homomorphic threshold cryptosystem, e. g., the Paillier cryptosystem [FP01,Pai99].

An additively homomorphic cryptosystem allows the encryption of the sum of two plaintexts $E(m_1 + m_2)$ with an operation $+_h$ in the ciphertext domain knowing only the ciphertexts c_1 and c_2. In addition, with a ciphertext $c = E(m)$ and a scalar value s, the encryption of the product $s \cdot m$ can be determined by applying the operation $+_h$ s times in the ciphertext domain using only the ciphertext c:

$$E(s \cdot m) = c \times_h s = \underbrace{c +_h \ldots +_h c}_{s \text{ times}}\,.$$

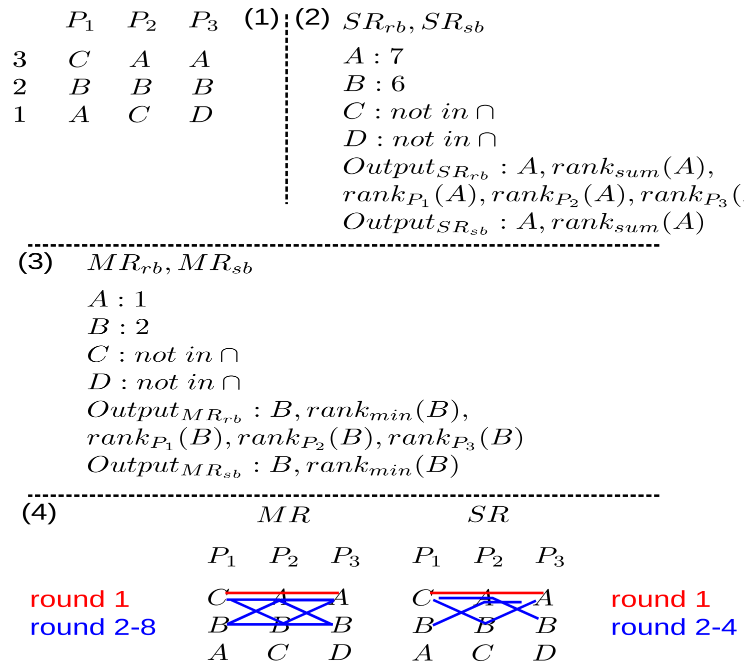

Fig. 1. (1) Example inputs and preferences (2) Output for SR-rb/SR-sb (3) Output for MR-rb/MR-sb (4) First rounds for MR-rb and SR-rb.

The output of a protocol run is determined by checking for the roots of the jointly decrypted output polynomial.

The two MPROS protocols SR-rb and SR-sb allow the n parties to determine (one or more) input(s) s that they all have in common and that maximizes the sum of the preferences each party assigned to its inputs. If no solution exists, the output is the empty set. The protocols are privacy-preserving with up to $c < n$ colluding parties in the semi-honest model. That is, apart from what can be deduced from the output of the protocols, the semi-honest parties do not learn anything about each others private inputs. A party is called semi-honest if it follows the protocol and performs all required computations yet it might store intermediate results and do additional computations.

Part (2) of Figure 1 illustrates the output of the SR protocols SR-rb and SR-sb. Note that in both protocols in addition to the common element with the highest sum of ranks A, the output contains the sum of ranks value for A which is seven. In the round-based protocol SR-rb the output additionally contains the exact rank assigned to A by each party. This is due to the fixed comparison order discussed below. The same holds for the set-based protocols in the special case of two parties P_1, P_2. I. e., each party knows the sum of ranks and the rank they assigned to a solution S. This uniquely determines the rank assigned by the other party as $rank_{P_2} = rank_{sum}(S) - rank_{P_1}(S)$. A similar observation can be made for the minimum of ranks protocols.

The protocols MR-rb and MR-sb enable the n parties to determine an unbiased solution that they all have in common and that maximizes the minimum of the preferences each party assigned to its inputs. If no solution exists, the output is the empty set. For up to $c < n$ colluding parties, the protocols are privacy-

Input

$$S_1 = \{C, C, C, B, B, A\}$$
$$S_2 = \{A, A, A, B, B, C\}$$
$$S_3 = \{A, A, A, B, B, D\}$$

MR

$$Rd_2(S_1 \cap S_2 \cap S_3) = Rd_2(\{B, B, A\}) = \emptyset$$
$$Rd_1(S_1 \cap S_2 \cap S_3) = Rd_1(\{B, B, A\}) = \{B\}$$
$$Output = B, rank_{min}(B)$$

SR

$$S'_1 = \{C^9, B^9, A^9\} \qquad S'_1 \cap S'_2 \cap S'_3 = \{A^9, B^9\}$$
$$S'_2 = \{A^9, B^9, C^9\} \qquad S_1 \cup S_2 \cup S_3 = \{A^7, B^6, C^4, D\}$$
$$S'_3 = \{A^9, B^9, D^9\}$$

$$Rd_8((S_1 \cup S_2 \cup S_3) \cap (S'_1 \cap S'_2 \cap S'_3)) = Rd_8(\{A^7, B^6\}) = \emptyset$$
$$Rd_7(...) = Rd_7(\{A^7, B^6\}) = \emptyset$$
$$Rd_6(...) = Rd_6(\{A^7, B^6\}) = \{A\}$$
$$Output = A, rank_{sum}(A)$$

Fig. 2. Example MR-sb and SR-sb

preserving in the semi-honest model. Part (3) of Figure 1 illustrates the output of MR-rb and MR-sb for three parties with three private inputs and preferences each. Similar to the sum of ranks protocols, the minimum of ranks of the optimal solution is part of the output of MR-rb and MR-sb. In addition, in MR-rb the individual ranks assigned by each party are revealed.

The protocols MR-rb and SR-rb are multi-round protocols, where each round consists of one or more private set intersection operations on sets of size 1. I. e., the input polynomials are all of degree 1. In the first round, party P_i's input set contains its most preferred element s_{i1}. In each of the following rounds, the parties participate in several private set intersections. The order in which the parties select their inputs in each round is fixed and determined such that in the first round in which the set intersections yield one or more non-empty set(s), these sets contain the common inputs that maximize the sum of ranks of all parties (in case of SR-rb) respectively the minimum of ranks assigned by all parties (in case of MR-rb).

The schedule exactly determines the input combination used in each round (compare Part (4), Figure 1) and therefore the individual ranks of the optimal solution are leaked. In the worst case, k^n rounds are required until the protocols SR-rb and MR-rb terminate. Part (4) of Figure 1 illustrates the operation of MR-rb and SR-rb for three parties with three inputs for the first eight respectively four rounds.

In the protocols MR-sb and SR-sb the preferences of each party are encoded as the multitude of the corresponding input in the party's input set. I.e. each party P_i ($i = 1, ..., n$) holds a private input set S_i which contains its k distinct inputs $s_{1i}, ..., s_{ik}$ with multitude $k, ..., 1$. The MR-sb protocol computes

$$Rd_t(S_1 \cap ... \cap S_n) \tag{2}$$

for $t = k-1, k-2, ..., 0$ until the resulting set is non-empty for the first time. All elements in this non-empty set then maximize the minimum of ranks. Due to the iterative reduction step, the minimum of ranks value of the optimal solution is revealed. As opposed to the round-based protocol MR-rb, the individual ranks of the other parties remain private.

Similarly, the protocol SR-sb computes

$$Rd_t((S_1 \cup ... \cup S_n) \cap S'_1 \cap ... \cap S'_n) \tag{3}$$

with $S'_i = \{(s_{i1})^{n \cdot k}, (s_{i2})^{n \cdot k}, ..., (s_{ik})^{n \cdot k}\}$ for $t = kn-1, kn-2, ..., n-1$ until the resulting set is non-empty for the first time. All elements in this non-empty set then maximize the sum of ranks. The auxiliary sets S'_i ensure that only inputs that are common to all parties are in the resulting set. The input polynomials with the highest degree occurring in the set-based approaches are $\frac{k(k+1)}{2}$ for MR-sb respectively $n \cdot k^2$ for SR-sb. Figure 2 illustrates the operation of the set-based protocols for three parties where the three inputs and preferences are as in Figure 1.

3 Framework Design

SMC-MuSe is a carefully designed framework for secure multi-party computation on multisets and as such supports application developers in implementing privacy-preserving applications. SMC-MuSe supports the developer in generating code for the distributed computation of any function expressible by a composition of set operations. We demonstrate the use of SMC-MuSe by an efficient implementation of the four MPROS protocols sketched in the previous section.

In addition, we show how SMC-MuSe allows an application developer to easily turn a protocol implementation into a user-friendly SMC application. The latter is demonstrated by the implementation of a scheduling application on top of the MPROS protocols. This scheduling application allows n parties to schedule a meeting in a privacy-preserving and fair manner, as illustrated in Figure 3. As such, SMC-MuSe provides for a proof-of-concept that general MPC applications can efficiently be implemented on top of privacy-preserving multiset operations. The SMC-MuSe framework was designed to exhibit a number of core properties.

- *Security and Privacy:* Building on privacy-preserving multiset operations, the framework is implemented in a way that provides for suitable security and privacy guarantees.
- *Usability:* SMC-MuSe is implemented in a way that makes SMC easily accessible to application developers and thus ultimately helps in bringing SMC applications to end-users.
- *Platform Support:* On the client side, our framework supports all major operating systems including Windows, Linux, MAC OS, and Android.
- *Scalability:* Our implementation supports large numbers of users simultaneously.

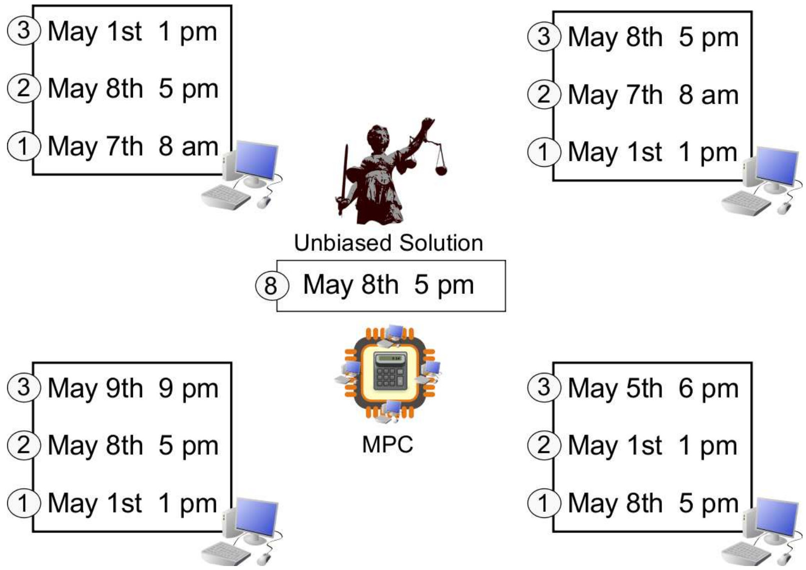

Fig. 3. Scheduling application example for SR

- *Reliability:* Our framework is resilient against a range of possible errors during SMC computations.
- *Modularity and Simplicity:* The framework is built in a way such that it can easily be extended to support additional modules such as other cryptosystems or different implementations of set operations. In addition, an application developer can integrate any function that can be expressed in the grammar in Equation 1 in a straightforward manner.

In the rest of this section, we describe the design and implementation of our framework. We start by a description of the system model we use to implement SMC-MuSe, continue with a description of the core features of SMC-MuSe and finally show how SMC-MuSe supports in protocol and application development.

3.1 System Model

The general goal of privacy-preserving multi-party protocols is to support n parties in jointly computing a function of their private inputs without revealing their inputs to each other. In the following we discuss four different choices for a system model supporting this goal and thereby motivate our own choice.

Model (1) of Figure 4 illustrates the *centralized approach*, which allows for an easy solution. It assumes that there is a third party that is trusted by all parties. Specifically, each party sends its inputs to the trusted third party (over a secure channel), the third party computes the output from the received plaintext inputs and sends back the result and only the result to each one of the participating parties. Unfortunately, in practice the existence of such a trusted third party is a very strong and quite unrealistic assumption.

A second approach is the *distributed approach* which is widely adopted in theoretical SMC. In theory, secure multi-party computation protocols solve the problem of a trusted third party and allow n parties to jointly compute a function of their private inputs without revealing anything but what can be deduced from

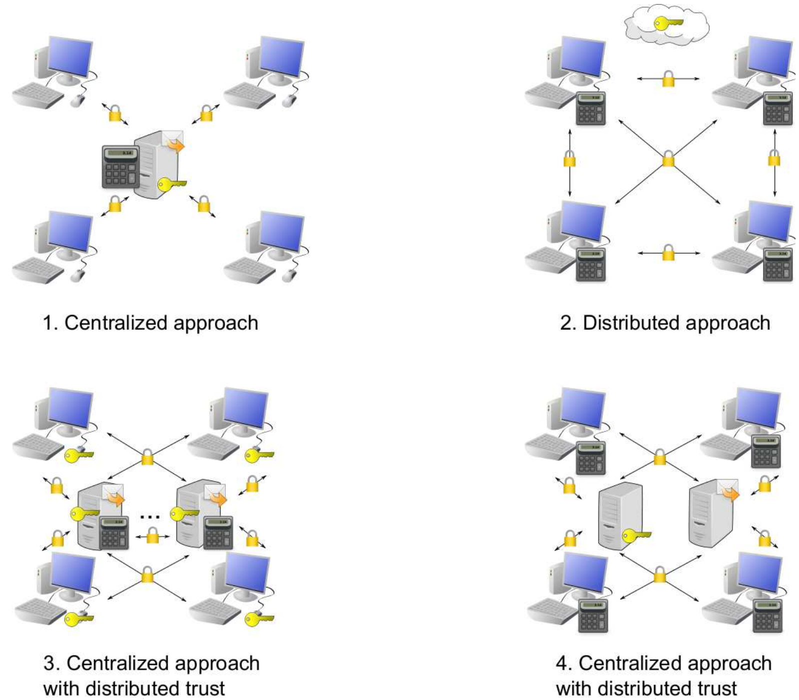

Fig. 4. System models (1) - (4)

the intended outcome to each other or to anyone else, not even to a trusted third party. To reach these privacy goals, SMC protocols typically assume that (1) all participating parties can directly communicate with each other, (2) there is a secure channel between any two of the participating parties, (3) any additional keying material that is required by the parties to jointly compute the function in a privacy-preserving manner is already available at the parties. In particular, these assumptions also hold for privacy-preserving multiset operations, where assumption (3) implies that each party holds a key share for an asymmetric semantically secure homomorphic threshold cryptosystem. Model (2) of Figure 4 illustrates the system model and assumptions of theoretical SMC in general.

Unfortunately, all three of the above assumptions on which SMC protocols are built are hard to establish in practice. Direct communication between Internet users is often prohibited due to policy restrictions, the presence of NATs or firewall configurations. Establishing a secure channel between any two users either requires an out of band channel to check the authenticity of credentials or the involvement of a third party like a certification authority or some form of a key server. Finally - in case of privacy-preserving multiset operations - assumption (3) requires the existence and implementation of a distributed key generation protocol that allows the n parties to jointly generate the key shares for the threshold cryptosystem.

Model (3) in Figure 4 illustrates a system model which is widely used in existing frameworks for SMC. Here, the parties are divided according to their role within the SMC computation. There are input parties, computing nodes, and result parties. The input parties provide the encrypted input to the computing nodes. The computing nodes perform the computation of the chosen function and forward their results to the result parties. A party/node can have any subset of those roles. Typically, the clients are input and result parties. The computation nodes are trusted servers within the network. The clients trust that the nodes correctly carry out the computations and forward the messages. All parties/nodes agree on keying material beforehand. Further details on this system model are discussed in Section 5.

Model (4) of Figure 4 illustrates our system model. We solved the challenges of theoretical SMC by introducing two independent non-colluding third parties called the Server component and the Keyserver in the following. The client components are responsible for all SMC computations, the initiation of protocol runs, and the exchange of messages with each other via the Server component.

The Server component is responsible for correctly forwarding messages with encrypted content between the clients. It is important to note that in contrast to the trusted third party in Model (1), the Server component in Model (4) does not learn anything about the plaintext inputs of the participating parties.

The Keyserver is responsible for generating SSL-certificates and generating key shares for the public/private key for the homomorphic threshold cryptosystem. Note that the Keyserver—although in possession of the private key pairs for the homomorphic threshold cryptosystem—does not have access to the messages exchanged between the clients via the Server. Unlike the trusted third party in Model (1), the Keyserver in Model (4) does not learn anything about the inputs or the output of the SMC computations. Note that SMC implementations operating in Model (3) typically assume pre-established keying material such that as opposed to Model (4) the key management problem of the theoretical Model (2) is not addressed in Model (3).

On the one hand, this system model allows for an easy automated setup phase and solves the theoretical assumptions for secure channels and key generation in a convenient way. On the other hand, one relies on two independent partially trusted components. We argue that without any trust as suggested in theoretical SMC (Model (2) in Figure 4), it is hardly possible to implement SMC in a way that it is easily accessible for non-technical users. This claim is further supported if one reviews existing related work (see Section 5).

3.2 Design Features

Based on the system model, we now continue the description of the design features of the SMC-MuSe framework. In the following we categorize design and implementation decisions based on the properties they enable—indicated in parenthesis.

Asynchronous Communication: Our choice of a system model implies that the Server component is involved in all communication. Thus, for a huge user base the expected traffic on the server side is assumed to be very high. Therefore, we have adapted the JBoss Netty project [Net] which provides an efficient asynchronous event-driven network application framework such that it can handle encrypted traffic, certificate verification, and serialization (*Scalability*).

Message Processing: We chose the MessagePack project (MP) [MPa] to serialize messages. MP provides small and fast serialization for many programming languages. Fast message processing on Keyserver/Server component- and

Worker Name	Computation Purpose / Task
ENCRYPT	Encrypt an input polynomial.
KEYGEN	Generate key pairs for an MPROS run.
ADD	Add two encrypted polynomials.
MULT	Multiply an encrypted polynomial and an unencrypted polynomial.
INTERSECT	Combination of ADD/MULT operations on encrypted polynomials.
UNION	Combination of MULT operations on encrypted polynomials.
REDUCTION	Combination of ADD/MULT operations on encrypted polynomials.
PARTIALDECRYPT	Partial decrypt an encrypted polynomial.
RECOVERY	Decryption of an encrypted polynomial.
	by combining an array of partially decrypted polynomials.
RESULTCHECK	Compute the roots of the decrypted polynomial.

Table 1. List of implemented workers

client-side is possible since messages are handled in an asynchronous event-loop. We use message queues to report results, new messages and state changes.

SMC Computation: SMC-MuSe provides interfaces to compute the intersection, union and reduction operation on multisets. These SMC operations provide the core security and privacy functionality of the framework. These expensive computations are outsourced to so-called worker components such that computation and message handling are separated (*Modularity*). A list of available workers is given in Table 1.

The multiset operations currently implemented in SMC-MuSe operate on input multisets encrypted by a semantically secure additively homomorphic threshold cryptosystem. The output of any combination of operations on multisets is jointly decrypted by the participating parties. This is why apart from the workers for the set operations there are workers for the encryption of inputs, the partial threshold decryption of an output and the combination of partial descriptions of an output. Note that all workers (except *KEYGEN* which is obviously computed on the key server) are executed on the clients. On each client the same set of workers is implemented and used for computations.

State-based SMC-Protocol Implementation: Any SMC-protocol that can be implemented in SMC-MuSe is a composition of multiset operations expressible in Equation 1. SMC-MuSe enables the implementation of SMC-protocols in a modular state-based fashion which yields three main advantages. First, it allows the Server component to store the state of the computation and encrypted intermediate results. In case of an error during computation or when a client disconnects, the Server component can use the stored information to later on resume an SMC-protocol run. In addition, storage failures (e. g., database errors) can be handled by means of frequent backups. Overall, this results in highly resilient SMC-protocol runs (*Reliability*). Second, the state-based implementation allows for easy adaptation of protocol runs (*Modularity*). This enables the modification of the computed function in an easy fashion. Third, the modular design yields source code that can be comprehended easily (*Simplicity*).

Secure and Efficient Storage of SMC-protocols: We chose MongoDB [MON] to store serialized SMC-protocol runs on the Server component and a file-based storage mechanism to keep intermediate results on the clients. Note that all stored intermediate results are encrypted. MongoDB is a high-performance NOSQL database available for different platforms (*Scalability*). The database is locally accessible by the Server component after authentication with separate login credentials.

Automated Secure Channel Establishment: SMC-MuSe provides interfaces to automatically generate an SSL keystore containing a private/public key pair for a client. The public key is submitted to the Keyserver for signing. The Keyserver generates a public key certificate and returns this to the client over a keyserver-side authenticated SSL connection. The client can then use the SSL certificate to establish mutually authenticated SSL connections with the Server as well as the Keyserver.

Automated Key Generation and Distribution: SMC-MuSe provides interfaces to two semantically secure homomorphic threshold cryptosystems. Currently, SMC-MuSe supports threshold versions of the ElGamal and Paillier cryptosystem [Elg85,Pai99]. The Keyserver component automatically generates key shares for any keysize/clients combination. Keying material is distributed to connected clients via secure channels. In addition, the Keyserver keeps a pool of freshly generated keying material for different combinations of keysize/clients to avoid delays while starting an SMC-protocol run. Note that fresh key shares are distributed to the clients whenever a new SMC-protocol run is initiated.

Summary: SMC-MuSe provides well-documented interfaces to all of the frameworks features such that application developers can make use of them when implementing a privacy-preserving application based on private multiset operations. In particular, they do not need to worry about implementing a threshold cryptosystem, implementing private multiset operations, setting up secure channels, or generating and distributing keys to the clients.

3.3 MPROS Protocols

In this section, we describe the design and implementation of the MPROS protocols on the example of MR-sb focusing on the communication and computation models.

Figure 5 illustrates our state-based implementation of the MR-sb protocol as an SMC-protocol. All computations are done by the clients utilizing the workers for SMC computation, compare Table 1. All communication is asynchronous. The Server component forwards messages between parties during an MPROS run. A message contains an opcode, information about the receiver, and encrypted data. The Keyserver component securely provides the keying material to all parties using mutually authenticated SSL connections.

First, the MPROS protocol starts with an initialization. Each client chooses his inputs and orders them according to his preferences (1.). Second, the Keyserver generates key shares for all clients and distributes them (2.). Third, all clients compute the encryption of the polynomial representing their input (3.). Next, all clients jointly compute the intersection operation on the encrypted polynomials (4.). $c+1$ clients compute the reduction operation on the encryption of the intersection polynomial (5.). All clients jointly decrypt the encryption of the reduction polynomial (6.). Finally, each client calculates the result of the computation by calculating the roots of the polynomial (7.).

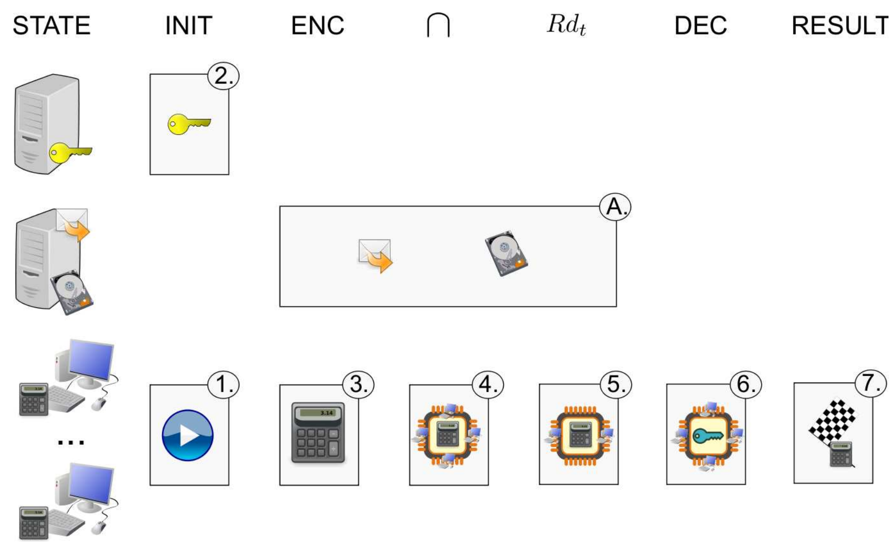

Fig. 5. States of the MR-sb protocol

During an MPROS protocol run, the Server component forwards incoming encrypted messages and stores encrypted intermediate results of the states ENC, $\cap$, and Rd_t to resume MPROS runs in case of client or Keyserver/Server component errors (A.). Due to the state-based implementation, clients can conveniently disconnect and reconnect within an SMC-protocol run.

3.4 GUI Components

To illustrate the potential of SMC-MuSe and in particular the MPROS implementation in SMC-MuSe, we created a proof-of-concept privacy-preserving Doodle-like application. Like Doodle, this new scheduling application allows several parties to agree upon a common meeting time. However, unlike in Doodle, in our application each party can assign preferences to each one of its free time slots which are taken into account when the best meeting time according to the sum of ranks or the minimum of ranks is determined. Figure 3 illustrates how the scheduling application works for four parties scheduling a meeting with SR-sb or SR-rb. Note that all parties keep their inputs private from each other and do not even reveal them to any server.

The graphical user interface (GUI) of the application is designed to target even non-tech-savvy user. A short video illustrating the components of SMC-MuSe and example executions of the scheduling application can be found here[2]. Appendix A includes screenshots of the GUI for desktop clients and provides further links to videos illustrating the implemented features.

The GUI components allow users to register an account with the Server component, maintain a friends list, add and delete friends from this list, securely chat with his friends, and —most importantly— execute MPROS protocol runs for scheduling a meeting with any subset of his friends.

Specifically, when a user first registers for SMC-MuSe, the client application automatically generates an SSL keystore. After the Keyserver signing process,

[2] http://www.youtu.be/ArsD5bQjxOk (short version)

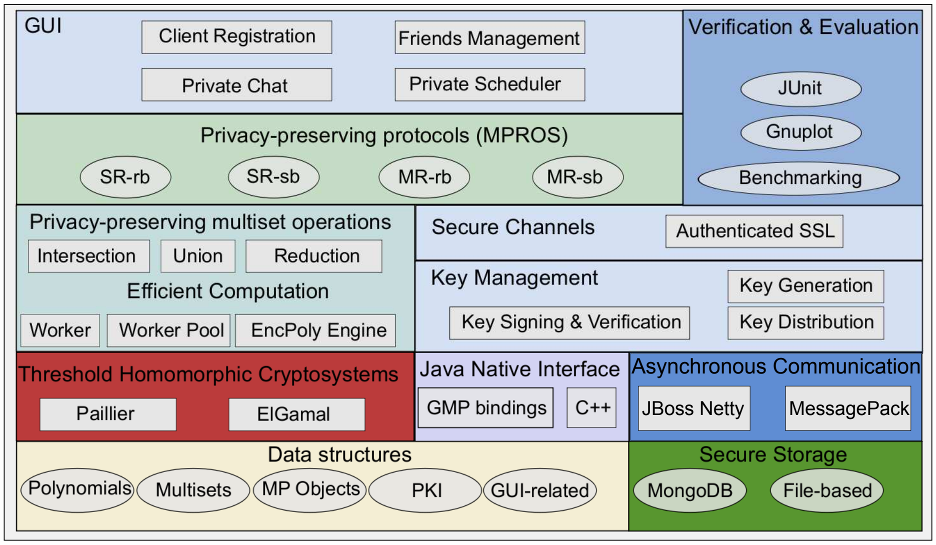

Fig. 6. Overview of SMC-MuSe

the client establishes mutually authenticated SSL connections with the Server as well as the Keyserver. User information associated with each friend added by the user includes an email address, a full name, and a public key for the Paillier cryptosystem. The latter is used for end-to-end encryption of chat messages exchanged between a user and a friend. The reconciliation part of the client application enables a user to create, initiate, and execute scheduling instances based on MPROS runs with any subset of friends. Note that further applications e. g. Sealed-Bid Auctions or Borda Count Voting [MNMW11], could easily be integrated reusing most of the existing GUI components.

3.5 Implementation

SMC-MuSe is written in Java. The overall library consists of two homomorphic cryptosystems, privacy-preserving multiset operations, multi-party reconciliation protocols, different components for communication and computation (see *Framework Features* above), and also a first GUI-supported application (Privacy-preserving Doodle). We use the gmp library for C++ to efficiently compute expensive mathematical operations such as modular exponentiation with JNI [JNI].

We have implemented threshold versions of the ElGamal and Paillier cryptosystem [Elg85,FP01,Pai99] as the additively homomorphic cryptosystems. Table 2 provides an overview of used libraries. The Worker components were implemented as a Worker Thread Pool which can handle up to 10 jobs in parallel. The GUI elements were also written in Java using the Java Swing framework [SWI]. We have created a Java Web Start application [WEB] which supports Ubuntu 10+, Debian Squeeze, Windows 7, and Max OS X 10.6.8+ in 32-bit and 64-bit modes. We have also developed an Android Client application which supports Android 2.2+ devices. Overall, the newly-developed framework SMC-MuSe consists of approximated 16,000 source lines of code (SLOC). An overview of all implemented components is given in Figure 6.

Library Name	Version
Java Platform	JDK 1.7, version 1.7.0_07
Java Runtime	Java HotSpot(TM) VM, version 1.7.0_07
JBoss Netty	Version 3.2.4.Final
MessagePack	Msgpack version 0.5.1-devel
MongoDB	mongoDB version 2.0.2

Table 2. Overview of used libraries

4 Evaluation

We evaluate our SMC-MuSe framework by analyzing the performance of the four MPROS protocols with respect to computation and communication overhead as there are already theoretical results regarding the protocol complexities [NMW11]. We measure the worst case complexity which means that the unbiased solution is the least preferred common input among all parties. We provide a description of our test setup in Section 4.1 and present a selection of test results in Section 4.2.

4.1 Test Environment and Parameters

We chose a test environment of desktop computers rather than a cluster-based test setup as the applications currently implemented in SMC-MuSe typically run on a user's desktop. The setup consists of 12 identical systems each with a 2.93 GHz i7 CPU 870 and 16 GB RAM running a 64-bit Linux with kernel version 2.6.32. One machine is dedicated as the Keyserver and another one as the Server component. The remaining ten machines are set up clients.

As a cross-platform effort we also successfully tested our framework with a Windows client as well as a Mac OS X client. The Chat Client application detects which OS is running and automatically loads the respective native libraries included within the jar-file.

Each MPROS protocol run is identified by a unique session id. The adjustable parameters for each run are the number of parties n, the number of colluding attackers c, the number of inputs k, the keysize l, and the protocol type t. In general, we tested our newly-developed and implemented framework SMC-MuSe with up to 10 parties and a maximum number of 10 inputs. We varied the keysize between 512 and 2048 bit. The input domain is equal the possible key space, i. e. $D = \{0, 1\}^l$. The inputs of each party were randomly generated integers chosen from the input domain D with one common input among all parties. In order to enforce the worst case behavior, the common rule was fixed as the rule with least preference for each party.

4.2 Test Results

For all MPROS protocols, the computation effort is dominated by the number of homomorphic operations (NOH) as shown by the theoretical performance results [NMW11]. We therefore start the presentation of our test results counting the NOH per party. We then present our results measuring the overall runtime of

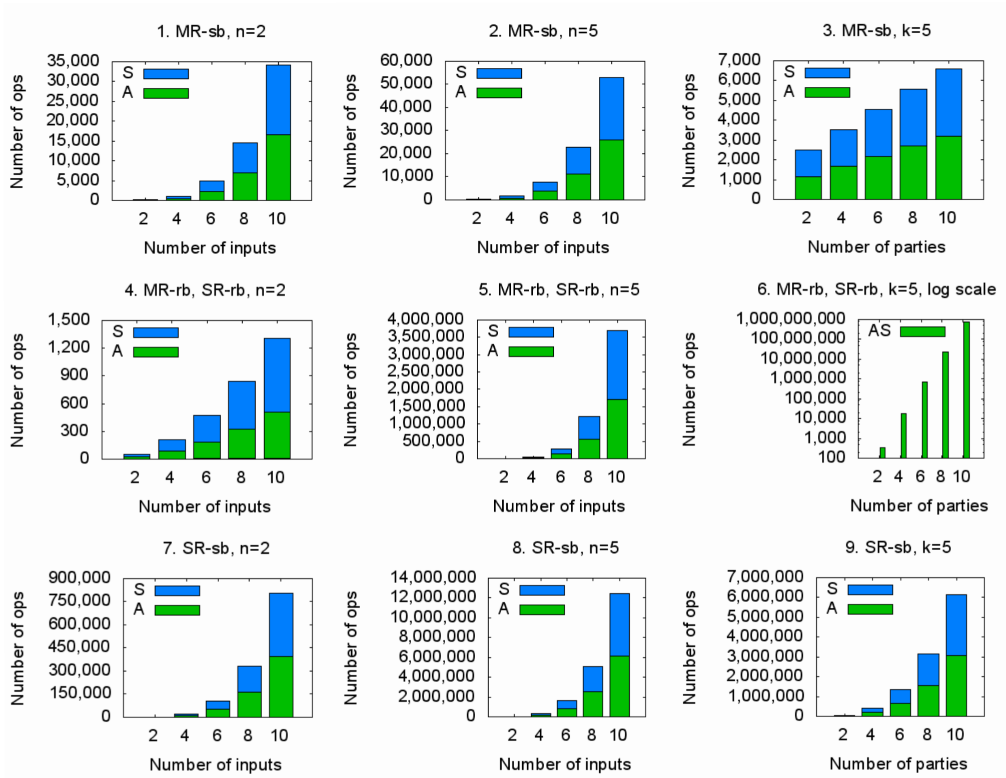

Fig. 7. Number of basic homomorphic (A), homomorphic scalar (S) operations or both in sum (AS) for *one* party with $c = n - 1$ in the worst case. (1-3) show results for MR-sb, (4-6) for MR-rb, SR-rb, and (7-9) for SR-sb in cases $n = 2, n = 5, k = 5$.

the four MPROS protocols for a reasonable keysize of $l = 1024$ bit varying the number of parties or the number of inputs. Finally, we include performance measurements for key sizes of $l = 1024, 2048$ bit varying the number of parties and the number of inputs at the same time.

Number of Homomorphic Operations Figure 7 provides an analysis of the number of homomorphic operations for all four protocols MR-sb, MR-rb, SR-rb, and SR-sb with respect to the number of parties n and the number of inputs k. Note that the worst case runtime and NOH are the same for the round-based protocols MR-rb, SR-rb for any combination of k, n (compare Section 2). We analyzed the NOH value per party during an MPROS protocol run in the cases $n = 2$, $n = 5$, $k = 5$ and varied k respectively n from 2 to 10 with $c = n - 1$. For the Paillier cryptosystem, the basic homomorphic operation is a modular multiplication and the homomorphic scalar operation is a modular exponentiation.

In the case $n = 2$ (see (1), (4), (7) in Figure 7), the round-based protocols MR-rb, SR-rb have the lowest NOH as the number of rounds is small (k^2) and the degree of the input polynomials is only one. With respect to the set-based protocols, MR-sb has a lower NOH than SR-sb in all three cases $n = 2, n = 5, k = 5$. This is due to the simpler function to compute for MR-sb (see Section 2). In the

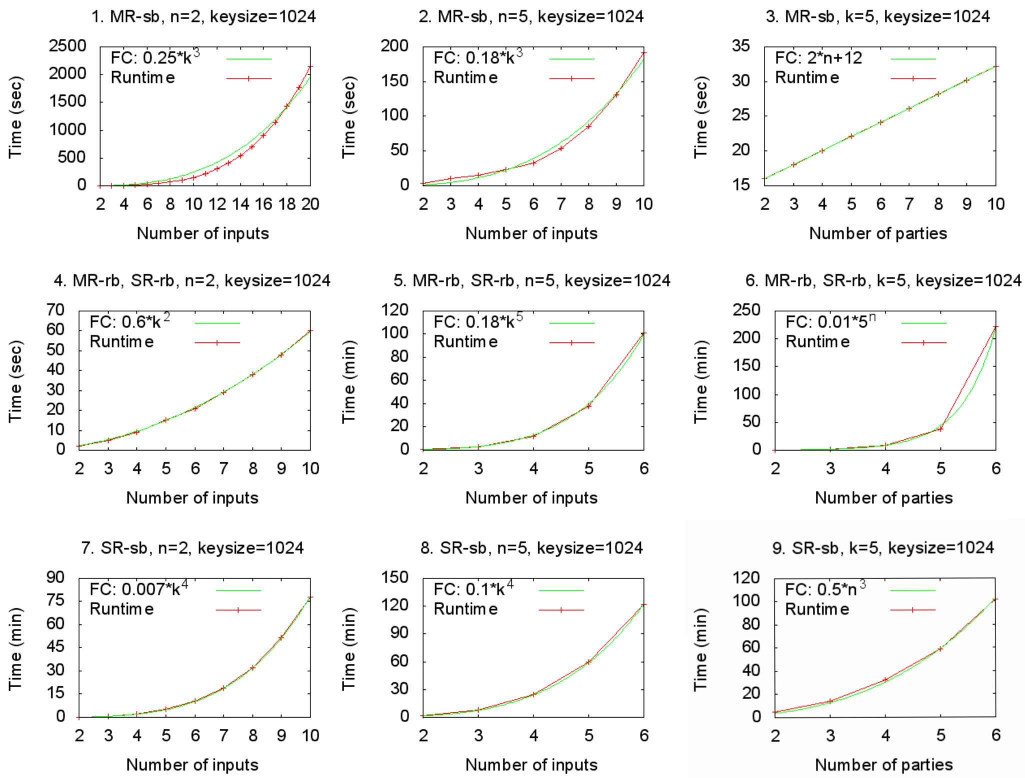

Fig. 8. Worst case runtime results with $c = n - 1$, $l = 1024$ for all MPROS protocols. (1-3) show results for MR-sb, (4-6) for MR-rb, SR-rb, and (7-9) for SR-sb in cases $n = 2, n = 5, k = 5$.

case $n = 5$ (see (2), (5), (8) in Figure 7), the MR-sb has the lowest and SR-sb the highest NOH. However, the round-based protocols MR-rb and SR-rb also show a high NOH due to the larger number of rounds (k^5). The exponential factor of the number of parties n in the round-based protocols becomes more clear in the case $k = 5$ (see (3), (6), (9) in Figure 7). Here, MR-rb and SR-rb have the largest NOH of approximately one billion for $n = 10$.

We expect that the set-based protocols MR-sb, SR-sb have a lower runtime than the round-based protocols MR-rb, SR-rb for a fixed value of n. As the NOH is relatively large for MR-rb, SR-rb , and SR-sb we limit our tests to n, k combinations up to $n = 6, k = 6$ for those protocols. In the special case $n = 2$, we extend our tests with k up to 20 for MR-sb and up to 10 for MR-rb, SR-rb, and SR-sb.

MPROS Performance We provide a detailed performance analysis of MPROS protocol runs for the keysize $l = 1024$ and $c = n - 1$ in Figure 8. We compare our runtime results with the theoretical analysis given in Table 3 [NMW11]. An MPROS protocol run consists of input generation, input encryption, computation of the function, threshold decryption, and computing of the final result. The overall runtime is averaged over three independent MPROS protocol runs for each parameter set. The plots show the overall runtime (*red with points*) and the

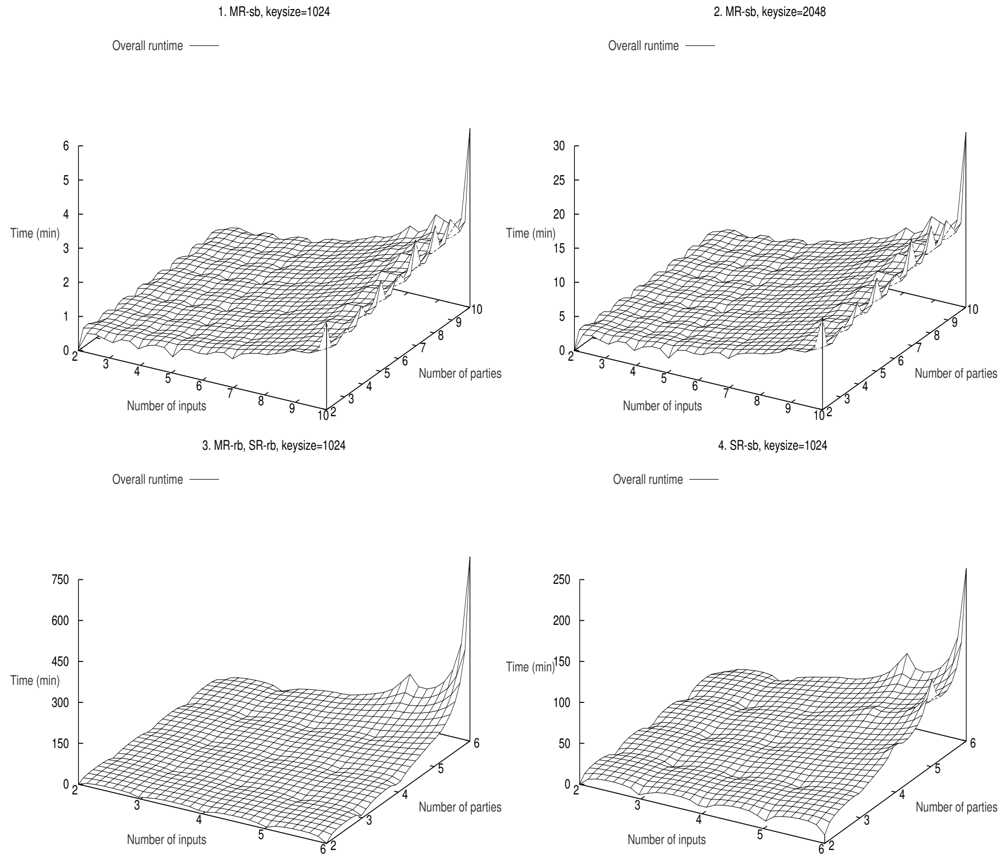

Fig. 9. Overall worst case runtime results with $c = n - 1$, $l = 1024, 2048$ for all MPROS protocols.

fitting curve (FC) best-fitting (*green*), i.e. the FC with the lowest asymptotic standard error.

For $n = 2$, we see that, as expected, the round-based protocols MR-rb, SR-rb have the lowest overall runtime. The FC matches the theoretical results of $O(k^2)$ with two parties. The set-based protocol MR-sb is also comparatively fast. Note that we tested MR-sb with up to $k = 20$. The best FC is one with expected runtime of $O(k^3)$ (green). The slowest protocol in the case of $n = 2$ is SR-sb. Here, the best-fitting FC shows a runtime of $O(k^4)$.

For $n = 5$, MR-sb is the fastest protocol again with cubic runtime with respect to the number of rules. The SR-sb protocol is still the slowest one with a best-fitting curve of $O(k^4)$. The round-based protocols MR-rb, SR-rb are much slower than in the case $n = 2$. The FC of $O(k^5)$ is best-fitting and matches the theoretical results.

In the case $k = 5$, MR-sb is again the fastest protocol with linear behavior with respect to the number of parties n as expected from theory see Table 3 [NMW10]. For $n = 6$ the round-based protocols MR-rb, SR-rb are even slower than SR-sb. In the case of MR-sb, the advantage of the set-based approach in practice already holds for $n = 3$.

There are two reasons for the advantage of the set-based protocols MR-sb, SR-sb over the round-based protocols MR-rb, SR-rb . The first one lies in the

Protocol	Communication (T)	Computation (T)	Practical results
MR-rb ,SR-rb	$O(1^l \cdot c \cdot n \cdot k^n)$	$O(1^l \cdot c \cdot n \cdot k^n)$	$O(1^l \cdot c \cdot n \cdot k^n)$
MR-sb	$O(1^l \cdot c \cdot n \cdot k^3)$	$O(1^l \cdot (c \cdot (k^6 + n \cdot k^4)))$	$O(1^l \cdot c \cdot n \cdot k^3)$
SR-sb	$O(1^l \cdot c \cdot n^3 \cdot k^3)$	$O(1^l \cdot n^4 \cdot c \cdot k^6)$	$O(1^l \cdot c \cdot n^3 \cdot k^4)$

Table 3. Summary of MPROS protocol complexities

increasing NOH for larger n, k combinations (k^n rounds in worst case). The second reason is that the communication overhead for each round is relatively high.

Summary Figure 9 shows the overall results for all MPROS protocols for the tested n, k combinations with $l = 1024$ respectively $l = 1024, 2048$ for MR-sb. The MR-sb protocol shows the best results and as such seems practical for many applications—even for a larger number of parties and inputs. For the SR-sb protocol, the results indicate that a simpler function to compute for SR-sb would be desirable. In the current implementation, the round-based protocols MR-rb, SR-rb are rather slow for a larger number of parties and rules. As shown in (1,2) the runtime is relatively stable with respect to the chosen keysize with an approximated quadratic behavior.

Table 3 summarizes the theoretical and practical performance results for all four MPROS protocols. Note that in theory (T), separate complexities are given for communication as well as computation. The complexities for the practical results are based on the overall runtime of an MPROS protocol run which includes communication and computation.

Limitations Our implementations perform well within our test range with a runtime of a few seconds to a few minutes. In particular, these runtimes are well suited for SMC-MuSe 's distributed Doodle-like scheduling application as the outcome is not expected in real-time anyway. However, it is clear that with a large number of inputs k, e. g. $k = 100$, all four implemented protocols are expected to have a high runtime. Specifically, none of the protocols seems suitable for real-time applications or applications in which the result of the reconciliation is instantly needed. With respect to the number of parties n, only the MR-sb protocol can cope with a large user base, e. g., $n = 100$ or $n = 1000$.

Further optimizations as pre-computation can be applied to our MPROS protocol implementation. Especially, the round-based protocols MR-rb, SR-rb would benefit of such optimizations as, e. g., this would allow for the compensation of communication delays.

5 Related Work

In the following we discuss related work in four areas: privacy-preserving set operations and reconciliation on ordered sets, discussions of the system model, privacy-preserving scheduling applications, and frameworks for SMC.

5.1 Privacy-Preserving Set Operations

The basic building blocks of SMC-MuSe are privacy-preserving set operations. The first two-party protocols for privacy-preserving set operations were introduced by Freedman et al. [FNP04]. Kissner et al. [KS05] extended this work to multisets and multiple parties. Other multi-party protocols for privacy-preserving set intersection include [CJS10,LW07,NAA+09]. A multi-party protocol for set union was proposed by [Fri07]. None of these protocols supports multisets though.

Recently, Blanton et al. [BA11] proposed multi-party protocols for privacy-preserving set operations on multisets. As opposed to the work of Kissner et al. [KS05] their work is based on secret sharing rather then oblivious polynomial evaluation. We based our SMC-MuSe implementation on the operations introduced in [KS05] as at the start of the implementation, only this work supported the use of multisets.

Meyer et al. [MWI07,MWI10] showed that any privacy-preserving two-party protocol for set intersection can be used to jointly solve certain optimization problems on ordered input sets. An optimization of these protocols was implemented in C++ and evaluated in [MTWM11]. Neugebauer et al. [NMW10] on the one hand showed that these two-party protocols can be generalized to multiple parties in a straight forward fashion using any multi-party set intersection protocol (MR-rb, SR-rb). On the other hand, they showed that with the help of multisets, more efficient MPROS protocols can be built (MR-sb, SR-sb). To the best of our knowledge, our implementation is the first MPROS implementation available.

Recently, Jónsson et al. propose a multi-party weighted set intersection protocol in [JKU11]. As in the SR-sb and SR-rb protocols the inputs are associated with weights and the protocol computes the sum of the weights. However, in [JKU11] the output are all values with a sum of weights above a threshold t and not only the value(s) with the highest sum of weights.

5.2 Privacy-Preserving Scheduling

A scheduling application solves the problem of finding common (best-matching) timeslots among multiple parties with certain availabilities, potentially taking individual preferences into account. In the context of SMC, this problem was first used as motivating example in [MWI10] in the two-party setting and later on implemented as a two-party iPhone app Appoint based on direct Bluetooth connections [MTWM11]. In [BJH+11], the authors also propose a scheduling application for mobile devices. As opposed to Appoint and our own multi-party scheduling application, [BJH+11] uses Model (3) as system model, i.e., it involves trusted servers for computations. The output in [BJH+11] is the set of all common inputs and does not take user preferences into account. Another privacy-preserving scheduling application for mobile devices was proposed in [DCDA11]. Here, each party assigns costs to the timeslots (similar to preferences). The output contains all timeslots where the sum of the individual costs are below a threshold (as opposed to the highest sum in Appoint, and SR-rb and SR-sb based scheduling). Also in [DCDA11], a server assists in the computation (e. g. for aggregation), i. e., the protocols use Model (3) as system model.

5.3 System Model Discussions

The challenges of implementing SMC in practice were recently discussed by Halevi et al. in [HLP11]. As we do in this report, they argue that the theoretical model for SMC (as illustrated in Model (2) of Figure 4) is not suitable for today's Internet users. In particular they question the assumption that all clients are directly connected and interact simultaneously during the computation of the desired output.

To address this problem, they study a new client/server model for secure multi-party computation in which each client interacts with the server once and the server computes the output. The server learns the output but does not learn anything else about each client's input. In contrast to [HLP11] we focus on establishing a practical system model in which all computations are still done by the clients and the server does not even learn the output of the computation. The server just enables asynchronous communication and offers a resume feature to deal with disconnected clients and network errors.

5.4 SMC Frameworks

In the following we focus on discussing frameworks which try to make SMC available to the general public and compare those to SMC-MuSe.

VIFF is a framework for SMC which is based on Shamir's secret sharing (SSS) and additively homomorphic encryption (AHE) [DGKN09,VIF]. The framework is integrated into the Python language and supports arbitrary function computation specified over Z_p or $GF(2^8)$. VIFF provides security against semi-honest adversaries and allows up to $c < \frac{n}{2}$ colluding attackers. VIFF is the only framework which uses the theoretical system model (Model (2), Figure 4). Clients need to be setup manually to install required dependencies and configure the clients. All required keying material for the secure channels as well as the secret sharing scheme needs to be manually setup. SSL channels between the clients are established with the help of the pre-configured keys. VIFF supports asynchronous communication utilizing the Twisted framework.

The most well-known practical application of VIFF is the implementation of a double auction in Denmark where the price for sugar beets was reconciled between merchants and farmers [BCD$^+$09]. However, the authors chose a system model with trusted computing nodes similar to Model (3) in Figure 4. The double auction was executed on three servers trusted in computation after the danish farmers and merchants submitted their encrypted bids to a database. The result was published to the general public.

SEPIA is a framework for SMC which is based on Shamir's secret sharing [BD08,BSMD10]. SEPIA is written in Java and enables to compute functions over Z_p. The authors implemented and tested their framework for a 62 bit prime p. This allows arithmetic operations on shares to be executed directly by CPU instructions resulting in very fast computation. However, it also limits the possible input domain and the range of computable functions since modular reductions of intermediate results have to be avoided [BSMD10].

SEPIA uses Model (3) of Figure 4 as their system model using SSL between the computing nodes and the input and computing nodes. Key generation and certification is left to an existing PKI. Keying parameters required for secret

Property	VIFF	SEPIA	Sharemind	FMP	SMC-MuSe
Cryptographic Primitive	SSS/AHE	SSS	ASS	BCE	AHE
Computable Functions	$Z_p, GF(2^8)$	Z_p	$Z_{2^{32}}$	Arbitrary	Equation 1
System Model (Figure 4)	Model (2)	Model (3)	Model (3)	Model (3)	Model (4)
Setup phase	Manual	Manual	Manual	Semi-automatic	Automatic
Already Applied in Practice	yes	yes	no	no	yes
Colluding attackers	$c < \frac{n}{2}$	$c < \frac{n}{2}$	$c < \frac{n}{2}$	$c < \frac{n}{2}$	$c \leq n-1$
Communication model	async.	sync.	sync.	sync.	async.

Table 4. Comparison of the SMC frameworks

sharing are manually configured. SEPIA is secure in the semi-honest model with up to $c < \frac{n}{2}$ colluding privacy peers. Note that in SEPIA, the computing nodes also learn the result of the computation. SEPIA uses a synchronous approach that all computing nodes have to be online during computation. The practical potential of SEPIA was illustrated in the context of distributed network monitoring [BD08].

Sharemind is a framework similar to SEPIA but it uses only three computation nodes. Sharemind is based on additive secret sharing (ASS) and is written in C++ [BLW08]. Sharemind targets high-performance computing and is especially suitable for functions which allow for a large amount of parallel computations.

FairplayMP is a framework for SMC which is based on boolean circuit evaluation (BCE) and is written in Java [BDNP08]. It allows for the evaluation of arbitrary functions specified in the framework-specific function definition language SFDL. Boolean circuits for functions specified in SFDL are automatically generated in FairplayMP. The authors tested the performance of their framework in the context of auctions with 8-bit integers as input and output.

FairplayMP uses Model (3) of Figure 4 as their system model. Secure channels are established via SSL. Currently, only a manual setup with configuration files seems to be supported. The same holds for the exchange of keying material. The framework is secure against semi-honest adversaries with $c < \frac{n}{2}$. Each input player adds his blinded input to the input wires. The circuit is jointly evaluated by the computation nodes, which communicate synchronously. Note that as opposed to SEPIA the computation nodes do not learn the result of the computation due to an initial blinding of inputs by the input parties.

Summary: Table 4 summarizes our comparison of SMC frameworks. Note that with respect to performance, a fair comparison between the frameworks currently seems close to impossible due to the very significant differences with respect to the input domain, the system model, the communication model, and the range of computable functions. In addition, the test environment used to evaluate the performance of the frameworks currently varies from desktop computers to cluster-based setups.

6 Conclusion

In this report, we presented the SMC-MuSe framework, which makes secure multi-party computation on multisets as a middleware available to application developers. We proposed a new practical system model, which overcomes the discrepancies between the system assumptions made in theory and the communication model most commonly used on the Internet today.

Ultimately, SMC-MuSe[3] brings privacy-preserving applications to regular Internet users. Our scheduling application is a nice example for such an application and we showed that its performance is well-suited to support interesting end-user applications. Currently, we plan to publish the source code of SMC-MuSe under a GPL license.

As a further part of our future work we plan to evaluate the usability of our privacy-preserving scheduling application with the help of a comprehensive user study. Furthermore we will investigate potential performance optimizations including the analysis of the benefits of recently published more efficient privacy-preserving multiset operations [BA11].

Acknowledgments

This work has been supported by the DFG project ME 3704/1-1. We also thank Christian Bien for his help in the development of the framework SMC-MuSe.

[3] SMC-MuSe in action: https://www.youtube.com/user/misterxyz42

References

BA11. Marina Blanton and Everaldo Aguiar. Private and oblivious set and multiset operations. Cryptology ePrint Archive, Report 2011/464, 2011. http://eprint.iacr.org/.

BCD+09. Peter Bogetoft, Dan Lund Christensen, Ivan Damgard, Martin Geisler, Thomas Jakobsen, Mikkel Kroigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. Financial Cryptography and Data Security, pages 325–343, Berlin, Heidelberg, 2009. Springer-Verlag.

BD08. Martin Burkhart and Xenofontas Dimitropoulos. Privacy-preserving distributed network troubleshooting - bridging the gap between theory and practice. *ACM Trans. Inf. Syst. Secur.*, 14(4):31:1–31:30, December 2008.

BDNP08. Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: A System for Secure Multi-Party Computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 257–266, New York, NY, USA, 2008. ACM.

BJH+11. Igor Bilogrevic, Murtuza Jadliwala, Jean-Pierre Hubaux, Imad Aad, and Valtteri Niemi. Privacy-preserving activity scheduling on mobile devices. In *Proceedings of the first ACM conference on Data and application security and privacy*, CODASPY '11, pages 261–272, New York, NY, USA, 2011. ACM.

BLW08. Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, pages 192–206, Berlin, Heidelberg, 2008. Springer-Verlag.

BSMD10. Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX SECURITY SYMPOSIUM*. USENIX, 2010.

CJS10. Jung Hee Cheon, Stanislaw Jarecki, and Jae Hong Seo. Multi-party privacy-preserving set intersection with quasi-linear complexity. Cryptology ePrint Archive, Report 2010/512, 2010. http://eprint.iacr.org/.

DCDA11. Emiliano De Cristofaro, Anthony Durussel, and Imad Aad. Reclaiming privacy for smartphone applications. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '11, pages 84–92, Washington, DC, USA, 2011. IEEE Computer Society.

DGKN09. Ivan Damgard, Martin Geisler, Mikkel Kroigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography'09*, pages 160–179, 2009.

DOO. Doodle easy scheduling. http://www.doodle.com/.

Elg85. T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469 – 472, jul 1985.

EY. Ernst and Young: Privacy Trends 2012. http://www.ey.com/Publication/vwLUAssets/Privacy_trends_2012/$FILE/Privacy-trends-2012_AU1064.pdf.

FNP04. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Proceedings of EUROCRYPT'04*, 2004.

FP01. Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*, pages 351–368, 2001.

Fri07. Keith Frikken. Privacy-preserving set union. In *Proceedings of the 5th international conference on Applied Cryptography and Network Security*, ACNS '07, pages 237–252, Berlin, Heidelberg, 2007. Springer-Verlag.

HLP11. Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure Computation on the Web: Computing without Simultaneous Interaction. In *CRYPTO*, volume 6841, page 128, 2011.

JKU11. Kristjan Valur Jonsson, Gunnar Kreitz, and Misbah Uddin. Secure multi-party sorting and applications. In *ACNS '11: Proceedings of the 9th international conference on Applied Cryptography and Network Security*, pages 122–122, 2011.

JNI. JNI: Java Native Interface for integration of code written in other languages. http://java.sun.com/docs/books/jni/.

KS05. L. Kissner and D. X. Song. Privacy-Preserving Set Operations (Last modified June 2006). In *CRYPTO*, pages 241–257, 2005.

LW07. R. Li and C. Wu. An unconditionally secure protocol for multi-party set intersection. In *ACNS '07: Proc. of the 5th intern. conference on Applied Cryptography and Network Security*, pages 226–236, Berlin, Heidelberg, 2007. Springer-Verlag.

MNMW11. D. Mayer, G. Neugebauer, U. Meyer, and S. Wetzel. Enabling fair and privacy-preserving applications using reconciliation protocols on ordered sets. In *34rd IEEE Sarnoff Symposium*, Princeton, 2011. IEEE, IEEE.

MON. MongoDB: Scalable, high-performance, open source NoSQL database. http://www.mongodb.org/.

MPa. MessagePack: Extremely efficient object serialization library for cross-language communication. http://msgpack.org/.

MTWM11. D. A. Mayer, D. Teubert, S. Wetzel, and U. Meyer. Implementation and Performance Evaluation of Privacy-Preserving Fair Reconciliation Protocols on Ordered Sets. In *First ACM Conference on Data and Application Security and Privacy (CODASPY'11)*, 2011.

MWI07. U. Meyer, S. Wetzel, and S. Ioannidis. Distributed privacy-preserving policy reconciliation. In *ICC*, pages 1342–1349, 2007.

MWI10. U. Meyer, S. Wetzel, and S. Ioannidis. New advances on privacy-preserving policy reconciliation. In *iacr eprint 2010/64*, 2010. http://eprint.iacr.org/2010/064.

NAA$^+$09. G. S. Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhary, and C. P. Rangan. Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security. In *Cryptology and Network Security*, pages 21–40, Berlin, Heidelberg, 2009. Springer-Verlag.

Net. JBoss Netty: Asynchronous event-driven network application framework. http://www.jboss.org/netty.

NMW10. G. Neugebauer, U. Meyer, and S. Wetzel. Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets. In *13th Information Security Conference (ISC) 2010*, LNCS, 2010.

NMW11. G. Neugebauer, U. Meyer, and S. Wetzel. Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets (Extended Version). 2011. http://eprint.iacr.org/2011/200.

Pai99. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology*, pages 223–238. Springer-Verlag, 1999.

SUN. Sunera Risk Management: Social Media Data Privacy, Oktober 2011. http://www.sunera.com.

SWI. Java Swing: Framework for graphical development. http://docs.oracle.com.

VIF. VIFF Framework. http://viff.dk/.

WEB. Java Web Start Technology: Enables standalone Java software applications deployed with a single click over the network. http://www.oracle.com.

A SMC-MuSe In Action

Figure 10, 11, and 12 show different parts of the application we implemented on to of the SMC-MuSe framework. In Figure 10, you can see the friends list chat (on the left), an example of a chat window of the integrated chat client (on the right) and the Java Web Start screen (at the top).

In Figure 11, you see the scheduling interface. On the left, you can see the creation window in which the initiator of the meeting can specify meeting name, choose a security level (high, medium, low) and choose a forced start date, i.e. a date on which the reconciliation starts even if not all of the desired participants have entered their availabilities and preferences yet. On the next tab, the initiator can select the duration of the meeting, and select the dates and times he proposes.

The scheduler view in Figure 12 shows the scheduling to which this user has currently been invited on the left. In the middle the user can select his availabilities from the shown proposed meeting times. On the right his current selection is shown and can be reordered with the up/down arrows.

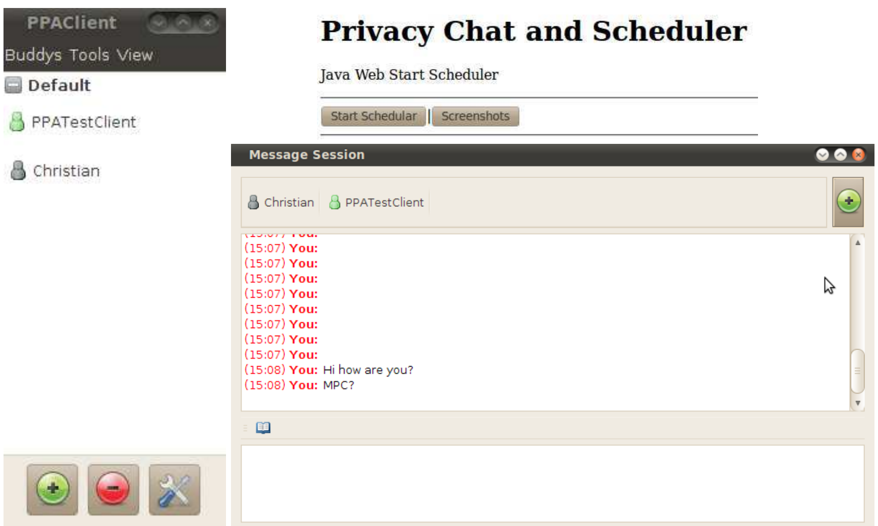

Fig. 10. SMC-MuSe's GUI

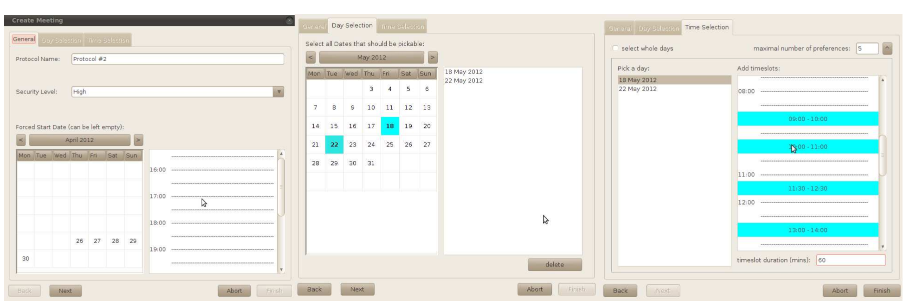

Fig. 11. SMC-MuSe's reconciliation interface

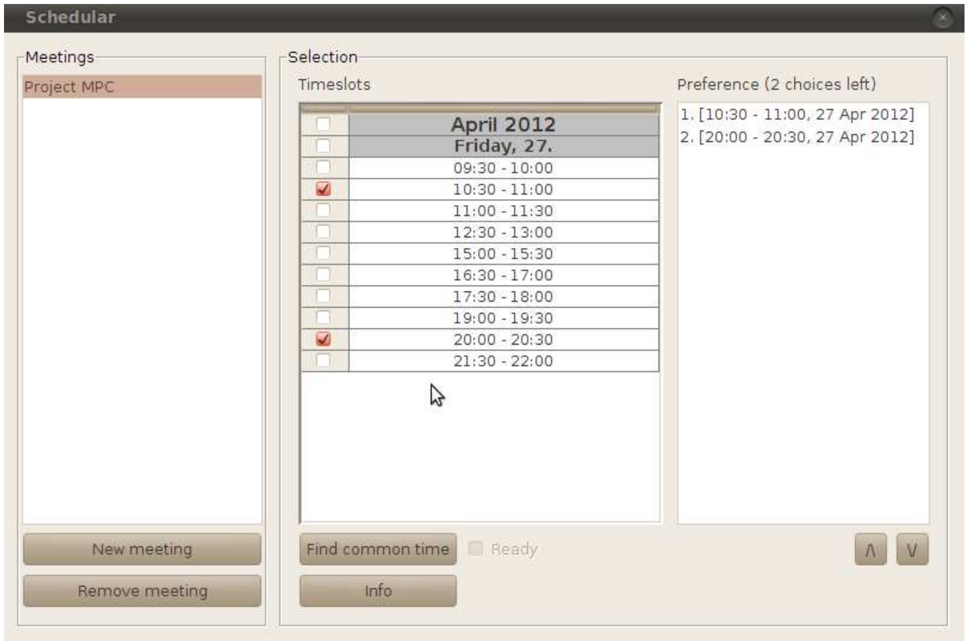

Fig. 12. SMC-MuSe's scheduler interface

We also created a variety of videos illustrating the scheduling application and other aspects of SMC-MuSe listed below. Note that all videos were created with the debug option enabled.

1. `http://www.youtube.com/watch?v=ArsD5bQjxOk`: Illustration of an MPROS run (MR-sb)
2. `http://www.youtube.com/watch?v=Owo8-Hg2Lnw`: MPROS run MR-rb
3. `http://www.youtube.com/watch?v=pUwicE0S3E4`: MPROS run SR-rb
4. `http://www.youtube.com/watch?v=H6aEvN5zjNY`: Client registration (end-users) and terminal-based start-up (application developers)
5. `http://www.youtube.com/watch?v=dtsDSgrNoQc`: Encrypted and authenticated chat, group chat, and friend management.
6. `http://www.youtube.com/watch?v=5JE0VwNpo0Y`: SR-sb and presentation of the resume feature of SMC-MuSe. A client disconnects and reconnects during an active MPROS run.
7. `http://www.youtube.com/watch?v=6Z1xpcEfeFo`: Second part of the resume feature. The MPROS run successfully finishes although a client disconnects during the computation. This works due to the asynchronous communication design and the state-based implementation of the MPROS protocols, see Section 3.

Aachener Informatik-Berichte

This list contains all technical reports published during the past three years. A complete list of reports dating back to 1987 is available from:

http://aib.informatik.rwth-aachen.de/

To obtain copies please consult the above URL or send your request to:

Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen,
Email: biblio@informatik.rwth-aachen.de

2009-01 * Fachgruppe Informatik: Jahresbericht 2009
2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications
2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices
2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs
2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete
2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I
2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs
2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem
2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm
2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs
2009-12 Martin Neuhäußer, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes
2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games
2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV'09)
2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäußer: Compositional Abstraction for Stochastic Systems
2009-16 George B. Mertzios, Derek G. Corneil: Vertex Splitting and the Recognition of Trapezoid Graphs
2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata
2009-18 Paul Hänsch, Michaela Slaats, Wolfgang Thomas: Parametrized Regular Infinite Games and Higher-Order Pushdown Strategies
2010-01 * Fachgruppe Informatik: Jahresbericht 2010
2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time

2010-03	Holger Krahn: MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering
2010-04	René Wörzberger: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme
2010-05	Daniel Retkowitz: Softwareunterstützung für adaptive eHome-Systeme
2010-06	Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Computing maximum reachability probabilities in Markovian timed automata
2010-07	George B. Mertzios: A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms
2010-08	Carsten Otto, Marc Brockschmidt, Christian von Essen, Jürgen Giesl: Automated Termination Analysis of Java Bytecode by Term Rewriting
2010-09	George B. Mertzios, Shmuel Zaks: The Structure of the Intersection of Tolerance and Cocomparability Graphs
2010-10	Peter Schneider-Kamp, Jürgen Giesl, Thomas Ströder, Alexander Serebrenik, René Thiemann: Automated Termination Analysis for Logic Programs with Cut
2010-11	Martin Zimmermann: Parametric LTL Games
2010-12	Thomas Ströder, Peter Schneider-Kamp, Jürgen Giesl: Dependency Triples for Improving Termination Analysis of Logic Programs with Cut
2010-13	Ashraf Armoush: Design Patterns for Safety-Critical Embedded Systems
2010-14	Michael Codish, Carsten Fuhs, Jürgen Giesl, Peter Schneider-Kamp: Lazy Abstraction for Size-Change Termination
2010-15	Marc Brockschmidt, Carsten Otto, Christian von Essen, Jürgen Giesl: Termination Graphs for Java Bytecode
2010-16	Christian Berger: Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles
2010-17	Hans Grönniger: Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten
2010-18	Ibrahim Armaç: Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit
2010-19	Felix Reidl: Experimental Evaluation of an Independent Set Algorithm
2010-20	Wladimir Fridman, Christof Löding, Martin Zimmermann: Degrees of Lookahead in Context-free Infinite Games
2011-01 *	Fachgruppe Informatik: Jahresbericht 2011
2011-02	Marc Brockschmidt, Carsten Otto, Jürgen Giesl: Modular Termination Proofs of Recursive Java Bytecode Programs by Term Rewriting
2011-03	Lars Noschinski, Fabian Emmes, Jürgen Giesl: A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems
2011-04	Christina Jansen, Jonathan Heinen, Joost-Pieter Katoen, Thomas Noll: A Local Greibach Normal Form for Hyperedge Replacement Grammars
2011-06	Johannes Lotz, Klaus Leppkes, and Uwe Naumann: dco/c++ - Derivative Code by Overloading in C++
2011-07	Shahar Maoz, Jan Oliver Ringert, Bernhard Rumpe: An Operational Semantics for Activity Diagrams using SMV
2011-08	Thomas Ströder, Fabian Emmes, Peter Schneider-Kamp, Jürgen Giesl, Carsten Fuhs: A Linear Operational Semantics for Termination and Complexity Analysis of ISO Prolog

2011-09 Markus Beckers, Johannes Lotz, Viktor Mosenkis, Uwe Naumann (Editors): Fifth SIAM Workshop on Combinatorial Scientific Computing

2011-10 Markus Beckers, Viktor Mosenkis, Michael Maier, Uwe Naumann: Adjoint Subgradient Calculation for McCormick Relaxations

2011-11 Nils Jansen, Erika Ábrahám, Jens Katelaan, Ralf Wimmer, Joost-Pieter Katoen, Bernd Becker: Hierarchical Counterexamples for Discrete-Time Markov Chains

2011-12 Ingo Felscher, Wolfgang Thomas: On Compositional Failure Detection in Structured Transition Systems

2011-13 Michael Förster, Uwe Naumann, Jean Utke: Toward Adjoint OpenMP

2011-14 Daniel Neider, Roman Rabinovich, Martin Zimmermann: Solving Muller Games via Safety Games

2011-16 Niloofar Safiran, Uwe Naumann: Toward Adjoint OpenFOAM

2011-17 Carsten Fuhs: SAT Encodings: From Constraint-Based Termination Analysis to Circuit Synthesis

2011-18 Kamal Barakat: Introducing Timers to pi-Calculus

2011-19 Marc Brockschmidt, Thomas Ströder, Carsten Otto, Jürgen Giesl: Automated Detection of Non-Termination and NullPointerExceptions for Java Bytecode

2011-24 Callum Corbett, Uwe Naumann, Alexander Mitsos: Demonstration of a Branch-and-Bound Algorithm for Global Optimization using McCormick Relaxations

2011-25 Callum Corbett, Michael Maier, Markus Beckers, Uwe Naumann, Amin Ghobeity, Alexander Mitsos: Compiler-Generated Subgradient Code for McCormick Relaxations

2011-26 Hongfei Fu: The Complexity of Deciding a Behavioural Pseudometric on Probabilistic Automata

2012-01 * Fachgruppe Informatik: Annual Report 2012

2012-02 Thomas Heer: Controlling Development Processes

2012-03 Arne Haber, Jan Oliver Ringert, Bernhard Rumpe: MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems

2012-04 Marcus Gelderie: Strategy Machines and their Complexity

2012-05 Thomas Ströder, Fabian Emmes, Jürgen Giesl, Peter Schneider-Kamp, and Carsten Fuhs: Automated Complexity Analysis for Prolog by Term Rewriting

2012-06 Marc Brockschmidt, Richard Musiol, Carsten Otto, Jürgen Giesl: Automated Termination Proofs for Java Programs with Cyclic Data

2012-07 André Egners, Björn Marschollek, and Ulrike Meyer: Hackers in Your Pocket: A Survey of Smartphone Security Across Platforms

2012-08 Hongfei Fu: Computing Game Metrics on Markov Decision Processes

2012-09 Dennis Guck, Tingting Han, Joost-Pieter Katoen, and Martin R. Neuhäußer: Quantitative Timed Analysis of Interactive Markov Chains

2012-10 Uwe Naumann and Johannes Lotz: Algorithmic Differentiation of Numerical Methods: Tangent-Linear and Adjoint Direct Solvers for Systems of Linear Equations

2012-12 Jürgen Giesl, Thomas Ströder, Peter Schneider-Kamp, Fabian Emmes, and Carsten Fuhs: Symbolic Evaluation Graphs and Term Rewriting — A General Methodology for Analyzing Logic Programs

2012-15 Uwe Naumann, Johannes Lotz, Klaus Leppkes, and Markus Towara: Algorithmic Differentiation of Numerical Methods: Tangent-Linear and Adjoint Solvers for Systems of Nonlinear Equations

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.