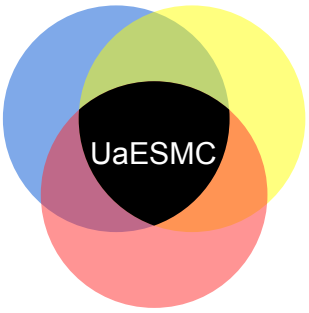

Project N°: **FP7-284731**

Project Acronym: **UaESMC**

Project Title: **Usable and Efficient Secure Multiparty Computation**

Instrument: **Specific Targeted Research Project**

Scheme: **Information & Communication Technologies**

Future and Emerging Technologies (FET-Open)

Deliverable D5.3
SMC application framework

Due date of deliverable: 30th April 2015

Actual submission date: 31st July 2015

Start date of the project: **1st February 2012** Duration: **42 months**

Organisation name of lead contractor for this deliverable: **CYB**

Specific Targeted Research Project supported by the 7th Framework Programme of the EC		
Dissemination level		
PU	Public	✓
PP	Restricted to other programme participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

Executive Summary:
SMC application framework

This document summarizes deliverable D5.3 of project FP7-284731 (UaESMC), a Specific Targeted Research Project supported by the 7th Framework Programme of the EC within the FET-Open (Future and Emerging Technologies) scheme. Full information on this project, including the contents of this deliverable, is available online at `http://www.usable-security.eu`.

The report contains the description of choices that an SMC implementor faces in terms of the computational task to be implemented, and the deployment environment. The report discusses the options the implementor has when designing the SMC protocol, and argues, which ones should be chosen in each particular situation. We find that there exists a sufficiently rich set of SMC protocols for primitive operations, such that large applications for many different kinds of tasks can be composed.

The richness of privacy-preserving applications already enabled by existing SMC protocols is demonstrated by a book "Applications of Secure Multiparty Computation", published by IOS Press in August 2015. The book has been edited by the performers of UaESMC. Throughout this report, we refer to the chapters of this book, showing how the choices made by the SMC implementor work out in practice. The electronic version of the book is available under the Creative Commons Attribution-NonCommercial 3.0 (CC BY-NC) license and downloadable from `http://ebooks.iospress.nl/volume/applications-of-secure-multiparty-computation`.

List of Authors

Peeter Laud (CYB)

Contents

Chapter 1

Introduction

One of the main outcomes of the UaESMC project, the UaESMC framework presents a coherent view of applicability of various secure computing techniques to various computational problems. The framework is intended to help an entity, the "SMC implementer", responsible for designing, implementing and deploying a privacy-preserving computation, that collects inputs from two or more entities and either publishes the outputs or makes them available to certain parties. The framework allows the SMC implementer to make the best choices regarding the underlying representations for private data, the used SMC protocols, the incentives to participate and follow the protocol, etc. In this manner, the framework facilitates a wider take-up of SMC techniques. It reflects the experience of UaESMC partners in constructing and executing SMC protocols for different tasks, obtained from activities in UaESMC and in other, concurrently-running projects.

To show the versatility and applicability of SMC, we have published a book [33] describing protocols and deployment models for solving many different problems in privacy-preserving manner. Many of the techniques described there have been developed in the UaESMC project. The examples provided by the book are intended to be taken up by developers and used in their privacy-preserving applications. There are already several examples where this has happened: the privacy-preserving statistics suite [33, Chap. 4] has been used to run a study on the future earnings of graduates and drop-outs of ICT curricula[1]; the methods for oblivious parallel data access [33, Chap. 6] have been used in privacy-preserving spam filtering in a project funded from DARPA's PROCEED (PROgramming Computations on EncryptEd Data) programme; the SMC deployment model [33, Chap. 1] has been adopted in the FP7-funded integrated project PRACTICE to discuss the application scenarios [20]. In fact, we consider the published book to be the main outcome of Task 5.3. The chapters of the SMC applications book [33] have been attached to this deliverable as an appendix.

[1] `http://cyber.ee/en/research/research-projects/prist/`

Chapter 2

The framework

When faced with an implementation of a privacy-preserving computation, the SMC implementor has to consider a number of aspects of the computation itself, the interfaces for collecting inputs and producing outputs, and the incentives of the participants. These consist of

- The number of involved parties, including the data providers and the parties due to learn the results of the computation (see also the characterization of the roles of different parties in an SMC deployment, given in [3]). Meaningful values are "selected few" (e.g. a short list of certain government agencies) and "many" (e.g. members of a social network; or voters).
- Computational capabilities of the involved parties. They may be running their own computational infrastructure and may be capable of deploying an SMC node. Or they may be able to manage a cloud-deployed SMC node. Or they may have little computational ability, limited to uploading their data to the computation and/or receiving the results. Possibly, there are different classes of involved parties with different capabilities.
- The details of the computational task itself. We discuss them below.
- The motivation of the parties to follow the protocol, including being truthful in inputting data, and in performing the steps of the protocol as they should.
- Additional infrastructure that the implementor may use, or must use. This may include the public key infrastructure of the country, in which the parties reside, and which is used to identify the parties or bind their messages to them. It may also include the bitcoin infrastructure, which may be used to create monetary incentives or penalties for the parties to follow the protocol.

We consider the primary applications for privacy-preserving computations to consist of the analysis of a data-set, followed by making a decision that optimizes the objectives of the participants. Either part may be (almost) missing. The analysis may be a statistical analysis, or a search for anomalies, or learning new patterns through data-mining. A typical decision-making process is the running of an auction, or agreeing on the details of a contract that is most beneficial to all participants, or establishing a common business process.

We will now consider the different aspects and see how they affect the choice of protocols. If different parts of the computational task have different characteristics, then it also may make sense to use different protocol sets for them, and convert as necessary, as long as the parts are sufficiently separated, such that not too many conversions are necessary [22, 12].

2.1 Number and capabilities of parties

Considering the current state of the art, our suggestion is to always try to limit the number of computing parties. These are the parties that participate in the computation more substantially than providing the

inputs (possibly by secret-sharing or encrypting them) and/or receiving the outputs. While SMC with many parties has been considered [53, 6], the existing protocols are probably too inefficient for practical applications [43].

Using a small number of parties, the choice is between protocol sets based on garbled circuits [52, 51], and protocol sets based on secret sharing [44, 15, 11, 5]. The choice between them mainly depends on the details of the computational task. For certain tasks (discussed below), protocols based on additively homomorphic encryption [8] may make sense, but there are probably not many such tasks. Fully homomorphic encryption [16] has not sufficiently matured to be generally considered in deployed protocols, but, again it is possible that it may make sense for applications with small multiplicative depth [48].

In any case, if the number of parties providing inputs is large, then they secret-share their inputs among the computing parties. This applies for both garbling circuit and secret sharing based protocols. The possible deployment schemes are discussed in [33, Chap. 1].

Depending on the computational abilities of parties, the computations of the SMC protocol may be executed by the parties themselves, or be outsourced to third parties (cloud providers). The SMC implementor has to make sure that at least the computing parties are able to manage a computation. Less computational ability is needed to give inputs to the SMC protocol, examples are provided in [33, Chap. 12].

2.2 Privacy-preserving computations

One general remark is in order before going to the details of computations. In order to make possible the arguments about the security and privacy of the implemented application, one should always use composable protocols [33, Chap. 2]. Let us now consider different details of the computational task.

2.2.1 Data types

Data manipulated by the computation may be numerical or categorical. The operations on numeric data include arithmetic operations. Both kinds of data may be subject to equality and inequality comparisons. Numeric data includes integers and real numbers, as well as elements of certain finite rings and fields. Categorical data includes strings, graphs, and other discrete structures. Both types of data may be stored in databases.

Arithmetic with integers (modeled as elements of some sufficiently large ring or field) is naturally supported by secret sharing based protocol sets. For real numbers, there exist protocols for fixed-point [7] and floating-point [24, 30] representations, covering both the arithmetic and relational operations, as well as elementary and statistical functions. One should note that the protocols for floating-point numbers are in general much more expensive, especially for addition, because they have to find a suitable value for the exponent of the result.

Secret sharing based SMC protocol sets usually support some operations on private data without any communication between the computing parties. Typically, this operation is addition in the ring or field underlying the secret sharing mechanism. Different secret sharing schemes may also support certain more complex operations with an overhead equal to much simpler operations. For Shamir's secret sharing [15], the scalar product of two vectors of private values only requires as much communication between the parties as a single multiplication. For three-party additive secret sharing as employed in Sharemind [5], the computation of all pairwise products of n private values only require $O(n)$ communication, not $O(n^2)$.

Protocols based on garbled circuits naturally allow the privacy-preserving execution of boolean circuits. The arithmetic operations have to be implemented on top of them. Significant effort has been spent on devising smaller circuits for arithmetic operations [28] or minimizing the size of circuits expressing the entire computations [29, 14]. Experience from integrated circuit design can also be used to minimize the size of the circuit [45]. In general, we believe that if the application is heavy on numeric computations, then it should use secret-sharing based SMC protocols, unless Sec. 2.2.3 gives strong reasons to choose otherwise.

On the other hand, we tend to manipulate categorical data using boolean circuits. Here the disadvantages of garbled circuits are smaller, even though purely communication-wise, secret sharing based protocol sets

should still be more efficient. Note that the comparisons of numeric values are also done using boolean circuits. Hence the SMC implementor has to choose, where to represent the private values used by the computation as numbers, and where to represent them as bit-vectors.

Examples of numerically heavy SMC protocols are given in [33, Chap. 4], while protocols working on categorical data are demonstrated in [33, Chap. 7].

2.2.2 Persistent storage

The SMC application may need database support for private values. The shares of secret-shared values can be stored by the computing parties using any suitable database technology, and recalled in the later stage of computations. In a similar manner, it is possible to store the keys and ciphertexts corresponding to the bits that we want to store in a stateful computation using garbled circuit based SMC protocols. The computing parties have to treat this database with care, not allowing it to be leaked.

Alternatively, the private values that must be stored between several executions of the private computation may be encrypted inside the SMC protocols [23, 9, 35]. In this manner, only the shares of the encryption key have to be carefully stored. The encrypted database is not sensitive by itself. On the other hand, the costs of the execution of encryption and decryption operations on top of SMC protocols are not trivial.

2.2.3 Complexity of computation, control flow and data access patterns

Some of the inputs, intermediate results and outputs of privacy-preserving computations are secret, while others are public. Public values may either be known before computation, or become known during the computation.

The operations that a computation performs may depend on data it is working on. If the data access patterns or the control flow of a privacy-preserving computation depends on private values, then the overheads of SMC protocols are higher and the deployment of specific protocols to reduce these overheads may be necessary. Having a choice between two algorithms, where the control flow or data access patterns of only one of them depend on private data, while the complexities of the algorithms are otherwise similar, it usually makes sense to choose the other algorithm. An example of such choice is presented in [33, Chap. 4.7]. In general, the following issues affect the choices of SMC protocols:

- parallelizability of the computation;
- complexity of data access patterns;
- privacy level of some intermediate values.

Parallelizability has perhaps the largest effect on our choice of protocols. Using SMC protocols based on secret sharing, each non-free operation (typically, additions are free) on private values requires one or more round-trips between the parties of the computation. Hence the round complexity of secret sharing based SMC protocols is at least proportional to the depth of the circuit expressing that computation. The round complexity of garbled circuit based protocols may be constant. If the computation we want to perform is inherently non-parallelizable, or parallelizable only with large overheads, then we may want to use garbled circuits.

If the data access patterns of a computation depend on private data, then we must use techniques of oblivious RAM (ORAM) to read the data without leaking the private addresses. Simplest techniques [34] have overheads proportional to the size of the memory from which we're reading. There exist ORAM protocols with polylogarithmic overheads, they have also been implemented on top of SMC protocols based on garbled circuits, enabling privacy-preserving memory accesses according to private addresses [10, 50]. If the memory access patterns are more restricted, e.g. the privacy of patterns is caused through the use of certain data structures, then even smaller overheads are possible [47, 17].

For SMC protocols based on secret sharing, there also exist techniques for oblivious parallel array accesses [32]. These techniques are efficient only if many accesses are performed in parallel. As secret sharing based protocols should be parallelized anyway, this is not a major restriction. An example of employing the oblivious parallel array access protocols is given in [33, Chap. 6].

The protocols for oblivious RAM have also been implemented on top of secret sharing based SMC protocols [26]. We believe that such combination of techniques has little utility in large SMC applications — the parallelism requirement of these SMC protocols is not answered by the used [46] ORAM methods.

For certain algorithms built on top of SMC protocols, the ability to declassify values in the middle of computation, and use them in the remaining part, significantly speeds up the execution. A well-known example is sorting, where methods with data-dependent memory access patterns are more efficient than oblivious methods based on sorting networks. Non-oblivious methods may also be secure if preceeded by an oblivious shuffle [21, 4] (see also [33, Chap. 4.3]). Declassification also supports more general transformation-based protocols for certain computational problems [33, Chap. 11]. SMC protocols based on secret sharing support declassification. Garbled circuits do not naturally support the declassification of values in the middle of computation. It can be emulated by splitting the computation to several parts and increasing the round complexity. The additional round complexity may be tolerable, if there is only a small number of declassification rounds (as is the case for sorting).

Certain complexities of the computational task do not affect much the choice of SMC protocol sets for privacy-preserving implementation. There are no good methods to deal with branchings based on private values. In privacy-preserving computation, both branches have to be evaluated and their effects combined [42]. This applies both for secret sharing based and garbled circuit based SMC protocols. To reduce computational overhead, one should try to identify any similarities in both branches, and execute them only once [39].

Garbled Random Access Machine [18, 25, 49] is a recent approach that does *asymptotically* reduce the cost of branching. In this setting, both the code of the machine and its memory are stored in oblivious RAM. Using MPC, the *next instruction* operation is implemented, including the client-side operations for oblivious RAM. The overheads of this approach make it currently unlikely to be competitive with more straightforward privacy-preserving implementations of algorithms.

2.2.4 Benefit of misbehaving

The computing parties may deviate from the protocol, if they wish to learn something about the private inputs, or want to change the outcome of the computation. The SMC implementor has to realistically judge if this may be the case. Possibly the computing parties can be trusted to faithfully follow the SMC protocols, because they have nothing to gain from learning the data or changing the outcome. Protocols secure only against passive adversaries may be up to two orders of magnitude faster than protocol secure also against active adversaries. The overhead of active security has two sources. First, the protocols have more steps, performing more checks. Second, for protocols based on secret sharing, the underlying fields have to be larger, because the security guarantees often depend on the size of the field. In particular, if the protocol works with boolean values, they still have to be shared over fields of size e.g. 40 bits. Security of SMC protocols against passive or active adversaries is discussed in [33, Chap. 1].

We believe that in most cases, full security against malicious parties is not necessary: a fear of getting caught with deviating should be enough to make the party follow the protocol. Verifiable SMC protocols have the potential to be much more efficient than protocols secure against fully malicious parties. We discuss verifiability in [33, Chap. 9] and show that for secret sharing based SMC protocols, they can offer a much richer set of primitive protocols, which translates to more efficient SMC applications.

The three-party protocol set of Sharemind based on additive secret sharing is secure only against passive adversaries, but it is *private* also against active adversaries, meaning that an active adversary can change the outcome of the private computation, but cannot anything it could not have learned if it had followed the protocol [40]. Using such protocol set, we do not have to verify all computations performed by computing parties. Instead, we can add verification routines into the SMC application and only verify, that

the parties followed the protocol while executing the verification routines. There are many functionalities that are easier to verify than to compute; linear programming and sorting are among the most well-known examples. Such verification is discussed in more detail in [33, Chap. 10].

The methods for achieving better-than-passively secure SMC protocols are still an active research area and further breakthroughs may be expected. In [36], an actively secure secret sharing based SMC protocol is used to construct garbled circuits, resulting in a constant-round actively secure protocol with better performance than the usual cut-and-choose based method for achieving actively secure garbled circuits. In [37], the garbling party is split into two, resulting in a three-party garbled circuit based SMC protocol, secure against one malicious party, and with almost the same efficiency than a passively secure garbled circuit.

The input parties may also misbehave by inputting data into the computation which does not match with the state of the real world and their knowledge. In this case the application itself has to be modified, possibly with the penalties for misreporting built in [33, Chap. 8]. The additional infrastructure available for the SMC parties may be used for defining the incentives.

2.2.5 Additional infrastructure

The parties may be identified by public keys, which are distributed using a well-recognized public key infrastructure. The SMC implementor may choose to incorporate this infrastructure in the SMC application, such that together with the outcome of the computation, the signatures on that outcome are also obtained. Also, if a cryptocurrency infrastructure (e.g. Bitcoin [38]) is available, then the agreed monetary transfers between participants could be a part of the outcome. Such signatures should probably be computed using SMC protocols based on secret sharing, because the boolean circuits for computing them are very large. We have discussed these options in [2]. If they are employed, then at least verifiable protocols should be used.

2.3 Additional considerations

2.3.1 Benefit of obtaining the result

Certain SMC protocol sets ensure that if the adversarial parties learn the result of the computation then all parties do. Such protocols are called *fair*. These protocols are typically expensive, providing security also against malicious adversaries. With the help of cryptocurrencies (e.g. Bitcoin) we can compile any SMC protocol into a fair one [1, 27]. The methods in this area are very recent and we expect further improvements in their efficiency, affecting both the construction of SMC protocols and the functionality provided by the Bitcoin ledger.

SMC techniques protect the intermediate values of computation from eavesdropping. They give no privacy guarantees for the outputs of the computation. If the outputs can be sensitive, e.g. if the outcomes of the statistical analysis could potentially give away something about unique individuals, then the computation itself has to be designed with appropriate safeguards. For statistical analysis type of tasks, differential privacy [13] is a common mechanism for ensuring the absence of leaks through outputs. The use of this mechanism brings its own overheads, but as discussed in [33, Chap. 5] and also demonstrated in [41], these overheads are not prohibitive at all and will likely cause only a minor increase in the running time of the SMC protocols.

2.3.2 Cost of information leakage

For certain tasks, there exist significantly faster SMC protocols, if a bit of the private information may be leaked. When working with private databases, we may have to join two tables according to a private index. A protocol for this is presented in [41], making use of techniques similar to [33, Chap. 6]. However, if we are willing to leak the equality patterns of the indices, then we can join the tables by first encrypting all indices using a pseudo-random function (typically AES), making them public, and doing the join in public [35]. This is asymptotically faster than a fully private join.

As a rule, SMC protocols do not hide the size of the tasks they are working on. The sizes may be hidden by padding, but this increases the effective size of the tasks and makes the protocols slower. In [33, Chap. 7] and in [31, 19] we see several examples of selectively opening the sizes of certain components of the task, in order to reduce the amount of padding and make the execution faster.

We do not have a general framework for deciding which parameters of the task may be leaked, and how the leaks can be quantified. Currently, the SMC implementor must weigh the possible choices. The development of a quantitative approach to measuring the leakages is a subject of future work, currently considered by agencies funding exploratory research[1].

[1] `http://www.darpa.mil/program/brandeis`

Bibliography

[1] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure multiparty computations on bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 443–458, May 2014.

[2] Dan Bogdanov, Yiannis Giannakopoulos, Roberto Guanciale, Dilian Gurov, Liina Kamm, Peeter Laud, Alisa Pankova, Pille Pruulmann-Vengerfeldt, Pille Pullonen, Riivo Talviste, Yiannis Tselekounis, and Jan Willemson. Scientific Progress Analysis and Recommendations, January 2014. UaESMC Deliverable 5.2.2.

[3] Dan Bogdanov, Liina Kamm, Sven Laur, and Pille Pruulmann-Vengerfeldt. Secure multi-party data analysis: end user validation and practical experiments. Cryptology ePrint Archive, Report 2013/826, 2013. `http://eprint.iacr.org/`.

[4] Dan Bogdanov, Sven Laur, and Riivo Talviste. A practical analysis of oblivious sorting algorithms for secure multi-party computation. In Karin Bernsmed and Simone Fischer-Hübner, editors, *Secure IT Systems - 19th Nordic Conference, NordSec 2014, Tromsø, Norway, October 15-17, 2014, Proceedings*, volume 8788 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2014.

[5] Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. High-performance secure multi-party computation for data mining applications. *Int. J. Inf. Sec.*, 11(6):403–418, 2012.

[6] Elette Boyle, Kai-Min Chung, and Rafael Pass. Large-Scale Secure Computation. Cryptology ePrint Archive, Report 2014/404, 2014. `http://eprint.iacr.org/`.

[7] Octavian Catrina and Sebastiaan de Hoogh. Secure multiparty linear programming using fixed-point arithmetic. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *ESORICS*, volume 6345 of *Lecture Notes in Computer Science*, pages 134–150. Springer, 2010.

[8] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, 2001.

[9] Ivan Damgård, Marcel Keller, Enrique Larraia, Christian Miles, and Nigel P. Smart. Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In Ivan Visconti and Roberto De Prisco, editors, *Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings*, volume 7485 of *Lecture Notes in Computer Science*, pages 241–263. Springer, 2012.

[10] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious ram without random oracles. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2011.

[11] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[12] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*. The Internet Society, 2015.

[13] C. Dwork. Differential privacy: A survey of results. *Theory and Applications of Models of Computation*, pages 1–19, 2008.

[14] Martin Franz, Andreas Holzer, Stefan Katzenbeisser, Christian Schallhart, and Helmut Veith. CBMC-GC: an ANSI C compiler for secure two-party computations. In Albert Cohen, editor, *Compiler Construction - 23rd International Conference, CC 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, volume 8409 of *Lecture Notes in Computer Science*, pages 244–249. Springer, 2014.

[15] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified vss and fact-track multiparty computations with applications to threshold cryptography. In *PODC*, pages 101–111, 1998.

[16] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.

[17] Craig Gentry, Kenny A. Goldman, Shai Halevi, Charanjit S. Jutla, Mariana Raykova, and Daniel Wichs. Optimizing oram and using it efficiently for secure computation. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies*, volume 7981 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[18] Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled RAM revisited. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 405–422. Springer, 2014.

[19] Dilian Gurov, Peeter Laud, and Roberto Guanciale. Privacy Preserving Business Process Matching. In *Proceedings of the 13th Annual Conference on Privacy, Security and Trust*. IEEE, 2015.

[20] Georg Hafner, Mario Münzer, Ferdinand Brasser, Janus Dam Nielsen, Peter Sebastian Norholdt, Dan Bogdanov, Riivo Talviste, Liina Kamm, Marko Jõemets, Meilof Veeningen, Niels de Vreede, Antonio Zilli, and Kurt Nielsen. PRACTICE Deliverable D12.1: Application Scenarios and their Requirements, April 2014. Available from `http://www.practice-project.eu`.

[21] Koki Hamada, Ryo Kikuchi, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Practically Efficient Multi-party Sorting Protocols from Comparison Sort Algorithms. In *Proc. of ICISC'12*, volume 7839 of *LNCS*, pages 202–216. Springer, 2013.

[22] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: tool for automating secure two-party computations. In *CCS '10: Proceedings of the 17th ACM conference on Computer and communications security*, pages 451–462, New York, NY, USA, 2010. ACM.

[23] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association, 2011.

[24] Liina Kamm and Jan Willemson. Secure floating point arithmetic and private satellite collision analysis. *International Journal of Information Security*, pages 1–18, 2014.

[25] Marcel Keller. The oblivious machine - or: How to put the c into mpc. Cryptology ePrint Archive, Report 2015/467, 2015. `http://eprint.iacr.org/`.

[26] Marcel Keller and Peter Scholl. Efficient, Oblivious Data Structures for MPC. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 506–525. Springer, 2014.

[27] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. Cryptology ePrint Archive, Report 2015/574, 2015. `http://eprint.iacr.org/`.

[28] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, volume 5888 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2009.

[29] Benjamin Kreuter, Abhi Shelat, Benjamin Mood, and Kevin R. B. Butler. PCF: A portable circuit format for scalable two-party secure computation. In Samuel T. King, editor, *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 321–336. USENIX Association, 2013.

[30] Toomas Krips and Jan Willemson. Hybrid model of fixed and floating point numbers in secure multiparty computations. In Sherman S. M. Chow, Jan Camenisch, Lucas Chi Kwong Hui, and Siu-Ming Yiu, editors, *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, volume 8783 of *Lecture Notes in Computer Science*, pages 179–197. Springer, 2014.

[31] Peeter Laud. A private lookup protocol with low online complexity for secure multiparty computation. In Siu Ming Yiu and Elaine Shi, editors, *ICICS 2014*, LNCS. Springer, 2014. To appear.

[32] Peeter Laud. Parallel Oblivious Array Access for Secure Multiparty Computation and Privacy-Preserving Minimum Spanning Trees. *Proceedings of Privacy Enhancing Technologies*, 2015(2):188–205, 2015.

[33] Peeter Laud and Liina Kamm, editors. *Applications of Secure Multiparty Computation*, volume 13 of *Cryptology and Information Security*. IOS Press, 2015. `http://www.iospress.nl/book/applications-of-secure-multiparty-computation/`.

[34] John Launchbury, Iavor S. Diatchki, Thomas DuBuisson, and Andy Adams-Moran. Efficient lookup-table protocol in secure multiparty computation. In Peter Thiemann and Robby Bruce Findler, editors, *ICFP*, pages 189–200. ACM, 2012.

[35] Sven Laur, Riivo Talviste, and Jan Willemson. From Oblivious AES to Efficient and Secure Database Join in the Multiparty Setting. In *Applied Cryptography and Network Security*, volume 7954 of *LNCS*, pages 84–101. Springer, 2013.

[36] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient Constant Round Multi-Party Computation Combining BMR and SPDZ. In *Proceedings of CRYPTO 2015*. Springer, 2015.

[37] Payman Mohassel, Mike Rosulek, and Ye Zhang. Fast and Secure Three-party Computation: The Garbled Circuit Approach. In Christopher Kruegel and Ninghui Li, editors, *Proceedings of 22nd ACM Conference on Computer and Communications Security (CCS)*. ACM Press, 2015.

[38] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[39] Alisa Pankova. Optimizing Secure Multiparty Computation Programs with Private Conditions. Programming Language Seminar report, University of Tartu, January 2015.

[40] Martin Pettai and Peeter Laud. Automatic Proofs of Privacy of Secure Multi-Party Computation Protocols Against Active Adversaries. In *Proceedings of 28th IEEE Computer Security Foundations Symposium*, 2015.

[41] Martin Pettai and Peeter Laud. Combining Differential Privacy and Secure Multiparty Computation. Cryptology ePrint Archive, Report 2015/598, 2015. `http://eprint.iacr.org/`.

[42] Aseem Rastogi, Matthew A. Hammer, and Michael Hicks. Wysteria: A programming language for generic, mixed-mode multiparty computations. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 655–670. IEEE Computer Society, 2014.

[43] Jared Saia and Mahdi Zamani. Recent results in scalable multi-party computation. In Giuseppe F. Italiano, Tiziana Margaria-Steffen, Jaroslav Pokorný, Jean-Jacques Quisquater, and Roger Wattenhofer, editors, *SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, January 24-29, 2015. Proceedings*, volume 8939 of *Lecture Notes in Computer Science*, pages 24–44. Springer, 2015.

[44] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[45] Ebrahim M. Songhori, Siam U. Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 411–428. IEEE Computer Society, 2015.

[46] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security*, pages 299–310. ACM, 2013.

[47] Tomas Toft. Secure data structures based on multi-party computation. In Cyril Gavoille and Pierre Fraigniaud, editors, *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 291–292. ACM, 2011. Full version in Cryptology ePrint archive, `http://eprint.iacr.org/2011/081`.

[48] Juan Ramón Troncoso-Pastoriza, Daniel González-Jiménez, and Fernando Pérez-González. Fully private noninteractive face verification. *IEEE Transactions on Information Forensics and Security*, 8(7):1101–1114, 2013.

[49] Xiao Shaun Wang, S. Dov Gordon, Allen McIntosh, and Jonathan Katz. Secure computation of mips machine code. Cryptology ePrint Archive, Report 2015/547, 2015. `http://eprint.iacr.org/`.

[50] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: Oblivious RAM for Secure Computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 191–202. ACM, 2014.

[51] Andrew C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[52] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE, 1982.

[53] Mahdi Zamani, Mahnush Movahedi, and Jared Saia. Millions of Millionaires: Multiparty Computation in Large Networks. Cryptology ePrint Archive, Report 2014/149, 2014. `http://eprint.iacr.org/`.