International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.12, December 2013 DOI:10.15693/ijaist/2013.v2i12.10-14

Analysis from the Scratch of Secure Failure Detection in Trusted Pals

Mr.S.Samson Dinakaran [1], Dr.M.Devapriya[2]

Research Scholar[1], Research Guide[2]

[1] Sri Krishna Arts & Science College, Coimbatore, Tamil Nadu, India.
[2] Government Arts College, Coimbatore, Tamil Nadu, India.

Abstract:- We present a modular redesign of TrustedPals, a smartcard-based security framework for solving secure multiparty computation (SMC) problem also known as secure computation or multi-party computation (MPC), a subfield of cryptography .The goal of methods for secure multi-party computation is to enable parties to jointly compute a function over their inputs, while at the same time keeping these inputs private. TrustedPals allows reducing SMC to the problem of fault-tolerant consensus between smartcards, where only process crashes and message omissions may take place. Hence, within the redesign aimed at incorporating failure detection, we investigate the problem of solving consensus in such an omission failure model augmented with failure detectors. The sub-problem of MPC that has received special attention by researchers because of its close relation to many cryptographic tasks is referred to as secure two-party computation (2PC) or just as Secure function evaluation (SFE). This area of research is concerned with the question: 'Can two party computation be achieved more efficiently and under weaker security assumptions than general MPC?' We make a comparative study of several protocols for SMC and try to identify a problem to be solved and implemented.
Keywords: TrustedPals, SMC, Consensus, SFE, Cryptography.

1. INTRODUCTION

The growth of the Internet has triggered tremendous opportunities for cooperative computation, where people are jointly conducting computation tasks based on the private inputs they each supplies. These computations could occur between mutually untrusted parties, or even between competitors. For example, customers might send to a remote database queries that contain private information; two competing financial organizations might jointly invest in a project that must satisfy both organizations' private and valuable constraints, and so on. Today, to conduct such computations, one entity must usually know the inputs from all the participants; however if nobody can be trusted enough to know all the inputs, privacy will become a primary concern. This problem is referred to as Secure Multi-party Computation Problem (SMC) in the literature. Research in the SMC area has been focusing on only a limited set of specific SMC problems, while privacy concerned cooperative computations call for SMC studies in a variety of computation domains.Secure multi-party computation (also known as secure computation or multi-party computation (MPC)) is a subfield of cryptography. The goal of methods for secure multi-party computation is to enable parties to jointly compute a function over their inputs, while at the same time keeping these inputs private. For example, two millionaires can compute which one is richer, but without revealing their net worth. In fact, this very example was initially suggested by Andrew C. Yao in a 1982 paper.,[1] and was later named the problem. The concept is important in the field of cryptography and is closely related to the idea of zero-knowledgeness. In general it refers to computational systems in which multiple parties wish to jointly compute some value based on individually held secret bits of information, but do not wish to reveal their secrets to one another in the process.

For example, two individuals who each possess some secret information—x and y, respectively—may wish to jointly compute some function $f(x,y)$without revealing any information about $x$and $y$other than what can be reasonably deduced by knowing the actual value of $f(x,y)$, where "reasonably deduced" is often interpreted as equivalent to computation within polynomial time. The primary motivation for studying methods of secure computation is to design systems that allow for maximum utility of information without compromising user privacy. Secure computation was formally introduced in 1982 by A. Yao [2] (incidentally, the first recipient of the Knuth Prize) as computation. The millionaire problem and its solution gave way to a generalization to multi-party protocols.[3] In an MPC, a given number of participants p_1, p_2, ..., p_N each have a private data, respectively d_1, d_2, ..., d_N. The participants want to compute the value of a public function F on N variables at the point (d_1, d_2, ..., d_N). An MPC protocol is secure if no participant can learn more from the description of the public function and the result of the global calculation than what he/she can learn from his/her own entry — under particular conditions depending on the model used.

2. DEFINITIONS OF SMCs

In this section we present the definition for secure two-party computation.

Two-party computation:

A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and

denote it f : {0, 1}_ × {0, 1}_ ! {0, 1}_ × {0, 1}_, where f = (f1, f2). That is, for every pair of inputs (x, y), the output-pair is a random variable (f1(x, y), f2(x, y)) ranging over pairs of strings. The first party (with input x) wishes to obtain f1(x, y) and the second party (with input y) wishes to obtain f2(x, y).

Adversarial behavior:

Loosely speaking, the aim of a secure two-party protocol is to protect an honest party against dishonest behavior by the other party. In this paper, we consider malicious adversaries who may arbitrarily deviate from the specified protocol. When considering malicious adversaries, there are certain undesirable actions that cannot be prevented. Specifically, a party may refuse to participate in the protocol, may substitute its local input (and use instead a different input) and may abort the protocol prematurely. One ramification of the adversary's ability to abort, is that it is impossible to achieve "fairness". That is, the adversary may obtain its output while the honest party does not. As is standard for two-party computation, in this work we consider a static corruption model, where one of the parties is adversarial and the other is honest, and this is fixed before the execution begins.

Security of protocols (informal):

The security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted third party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the above-described ideal computation. Execution in the ideal model. As we have mentioned, some malicious behavior cannot be prevented (for example, early aborting).
This behavior is therefore incorporated into the ideal model. An ideal execution proceeds as follows:
Inputs: Each party obtains an input, denoted w (w = x for P1, and w = y for P2).

Send inputs to trusted party: An honest party always sends w to the trusted party. A malicious party may, depending on w, either abort or send some w0 2 {0, 1}|w| to the trusted party.
Trusted party answers first party: In case it has obtained an input pair (x, y), the trusted party first replies to the first party with f1(x, y). Otherwise (i.e., in case it receives only one valid input), the trusted party replies to both parties with a special symbol ?

Trusted party answers second party: In case the first party is malicious it may, depending on its input and the trusted party's answer, decide to stop the trusted party by sending it ? after receiving its output. In this case the trusted party sends ? to the second party. Otherwise (i.e., if not stopped), the trusted party sends f2(x, y) to the second party.
Outputs: An honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the message obtained from the trusted party.

3. MPC PROTOCOLS ASSUMPTIONS

Like many cryptographic protocols, the security of an MPC protocol can rely on different assumptions:

- It can be computational (i.e. based on some mathematical problem, like factoring) or unconditional (usually with some probability of error which can be made arbitrarily small).
- The model in which the scheme is described might assume that participants use a synchronized network (a message sent at a "tick" always arrives at the next "tick"), that a secure and reliable broadcast channel exists, that a secure communication channel exists between every pair of participants (an adversary cannot read, modify or generate messages in the channel), etc.
- The centrally controlled adversary considered can be passive (only allowed to read the data of a certain number of participants) or active (can corrupt the execution protocol or a certain number of participants).
- An adversary can be static (chooses its victims before the start of the multi-party computation) or dynamic (can choose its victims during the course of execution of the multiparty computation). Attaining security against a dynamic adversary is often much harder than security against a static adversary.
- An adversary can be defined as a threshold structure (meaning that it can corrupt or simply read the memory of a number of participants up to some threshold), or be defined as a more complex structure (it can affect certain predefined subsets of participants, modeling different possible collusions). These structures are commonly referred to as adversary structures. The opposite set consisting of the sets of honest parties that can still execute a computational task is related to the concept of access structures.

4. TWO PARTY COMPUTATION PROTOCOLS

The sub-problem of MPC that has received special attention by researchers because of its close relation to many cryptographic tasks is referred to as secure two-party computation (2PC) or just as Secure function evaluation (SFE). This area of research is concerned with the question: 'Can two party computations be achieved more efficiently and under weaker security assumptions than general MPC?'

5. **VIRTUAL PARTY PROTOCOLS**
6.

Virtual Party Protocol is an SMC protocol which uses virtual parties and complex mathematics to hide the identity of the parties.

7. **SECURE SUM PROTOCOLS**

Secure sum protocols allow multiple cooperating parties to compute some function of their individual data without revealing the data to one another.[7]

8. **TRUSTED Vs UNTRUSTED PALS**

Trusted Network

Computers on the trusted network can transparently access such departmental services as NFS (home and project disks), NIS (distributed account and other information), printers, software packages, etc.

Access to this network is limited to machines administered by the Lab Staff, in order to secure sensitive data and maintain the availability of departmental resources. Users should not attempt to connect their personal machines to this network. A current list of machines on the trusted network can be found here.

Untrusted Network

In most cases, computers on the untrusted network are controlled and configured by their owners. Since this could potentially allow improper access to sensitive or personal data, or could cause operational disruptions to the production computer network, these machines are isolated on a separate sub-net, and are not given direct access to many core departmental computer services (see the previous section of this page). Some services can be accessed via an authentication and access package called SAMBA. To set this up please see the help guides for Windows and MacOS.

9. **CONCLUSION AND FUTURE WORK**

The **first** contribution is the size of the payload is to be chosen. It is necessary to find an acceptable tradeoff between security and performance such that a message size provides better security in expense of worse performance. So, we use an adaptive model for the tradeoff between service performance and security in service-based environments is presented. The performance and security metrics allow us to quantitatively calculate how much protection a security configuration vector can provide and how much performance will be decreased by that security configuration vector.

Minimum Requirement Validation

Basically, the tradeoff between performance and security is implemented through resource allocation. First, the SBS has to allocate certain amounts of resources for both performance and security to satisfy their minimum requirements. Then, if there are more resources, the Service based systems can allocate the available resources for better performance or better security. Hence, to check whether the tradeoff is possible, we first need to make sure that the minimum performance and security requirements can be satisfied. The service based systems required that the delay should be less than traffic. The success probability of an attacker with capability *c* should be less than minimum security requirement. Service based systems only supports a limited number of security algorithms and key lengths, we can check whether both the minimum performance and security requirements can be satisfied by enumerating all supported security algorithms and key lengths to see if the above condition can be satisfied.

Tradeoff Objective Function

When both minimum performance and security requirements are satisfied, the Service based systems can use the available resources for better performance or security. To have better security, as shown in the security metric, the Service based systems can use either a stronger algorithm with a longer key, or a larger protection percentage. Hence, to control the tradeoff between performance and security, we have to combine the performance metric and security metric together as a tradeoff objective function.

(i) Performance Biased Objective Function

The performance biased objective function is a tradeoff objective function that tries to maximize performance without violating the minimum security requirement. For the tradeoff objective function , the performance biased objective function sets the weighting factor of the security to 0. In this case, to minimize the tradeoff objective function is equivalent to minimizing the delay. When the encryption algorithm and the key length are fixed, the performance biased tradeoff should always use the minimum protection percentage.

(ii) Security Biased Tradeoff Function

The security biased tradeoff function is a tradeoff objective function that tries to maximize security without violating the minimum performance requirements. For the tradeoff objective function , the security biased tradeoff function sets the weighting factor of the performance to 0. In this case, to minimize the tradeoff objective function is equivalent to minimizing the attacker's success probability *S*. When the encryption algorithm and the key length are fixed, we can compute the upper limit for the protection percentage from the minimum performance requirements

(iii) Non-linear tradeoff objective function

If the SBS consumer's preferences on performance and security, i.e., the weighting factors do not change with the real-time performance and security conditions, we call such a

tradeoff objective function as a linear tradeoff objective function, like the performance biased tradeoff function and the security biased tradeoff function. On the contrary, if the weighting factors *a* and *b* are related to the current performance and security, we call such a tradeoff objective function as a non-linear tradeoff objective function. There may be various ways to define a non-linear tradeoff model, but all definitions should have the following properties: The weighting factor of performance increases when the performance approaches the minimum performance requirement. The minimum performance or security requirements are critical for the SBS. Hence, when the minimum performance requirements are not satisfied, the weighting factor of performance becomes infinite.

The **second** contribution is to achieve minimal storage and communication effort is necessary to overcome the consensus in the model. So, we use unbounded buffers in this model. Unbounded-buffer places no practical limit on the size of the buffer. For that we use competitive algorithm, the normalized diameter of the network, for general networks with general metrics. This minimizes the delivery costs, but is associated with a storage cost.

Competitive online algorithm

Let us now present an competitive online algorithm for general networks, and analyze it using h-balls. For ease of exposition we analyze the algorithm for unit costs. Algorithm Store is rather simple: upon the arrival of a request at a node v at time t, the request is satisfied by obtaining a movie from the closest node u that has a copy of the movie. The movie is saved in v's cache for a period of time that is proportional to the distance between u and and v. If several requests arrive at the same time, we process them one by one in an arbitrary order. In addition to that we use approximation algorithm for Movie Allocation in the grid network. To simplify the exposition, we first describe the algorithm for the special case of unit costs.

The **third** contribution is using aggregation method for reducing the complexity. Because in the Trusted pal framework, the connection between the untrusted and trusted system is achieved by associating each process in the untrusted system (i.e., each host) with exactly one process in the trusted system (i.e., a security module) and vice versa. So, it increases the complexity and cost. So, in order to overcome this problem, we use aggregation methods in the untrusted process to combine the untrusted hosts and to give one security model for a group of untrusted hosts.

Supervised rank aggregation Method

The whole sequence is divided into three parts: training, labeling and testing or validation. Each attribute of an example has the capacity to provide some unique information about the data when considered individually. The training examples are ranked based on the attribute values. So, for each attribute we will get a ranked list of all examples. Considering only the top k ranked examples and with an assumption that when we rank the examples according to their attribute values, the positive examples should be ranked on the top, we compute the performance of each attribute. This performance is measured in terms of either precision or false positive rate or a combination of both. Based on the individual performances, a weight is assigned to each attribute. For validation, we use examples obtained from the validation graph characterized by same attributes and try to rank all examples based on their attribute values. So for n different attributes we shall have n different rankings of the test examples. These ranked lists are then merged using a supervised rank aggregation method and the weights of the attributes obtained during learning process.

10. REFERENCES

1. ^ Andrew Chi-Chih Yao: Protocols for Secure Computations (Extended Abstract) FOCS 1982: 160-164
2. ^ Andrew C. Yao, Protocols for secure computations (extended abstract)
3. ^ O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In Proceedings of the nineteenth annual ACM conference on Theory of computing, pages 218-229. ACM Press, 1987.
4. ^ Claudio Orlandi: Is multiparty computation any good in practice?, ICASSP 2011
5. ^ Peter Bogetoft and Dan Lund Christensen and Ivan Damgård and Martin Geisler and Thomas Jakobsen and Mikkel Krøigaard and Janus Dam Nielsen and Jesper Buus Nielsen and Kurt Nielsen and Jakob Pagter and Michael Schwartzbach and Tomas Toft: Multiparty Computation Goes Live, Cryptology ePrint Archive: Report 2008/068
6. ^ Pathak Rohit, Joshi Satyadhar, Advances in Information Security and Assurance, Springer Berlin / Heidelberg, ISSN 0302-9743 (Print) 1611-3349 (Online), ISBN 978-3-642-02616-4, DOI 10.1007/978-3-642-02617-1
7. ^ Rashid Sheikh, Brijesh Kumar and Durgesh Kumar Mishra, Privacy Preserving k-secure sum protocols, International Journal of Computer Science and Information Security, ISSN 1947-5500 (Online),Vol.6, No.2, Nov. 2009

External links

- Solution to the Millionaire's Problem A description of Yao's algorithm
- Helger Lipmaa's links about multiparty computation
- Nick Szabo, "The God Protocols"
- Secure distributed CSP (DisCSP) solvers — a web-application with an applet-interpreter to design and run your own full-fledged secure multiparty computation (based on the SMC declarative language). Uses secure arithmetic circuit evaluation and mix-nets.

- VMCrypt A Java library for scalable secure computation. By Lior Malka.
- The Fairplay Project — Includes a software package for secure two-party computation, where the function is defined using a high-level function description language, and evaluated using Yao's protocol for secure evaluation of boolean circuits.
- The SIMAP project; Secure Information Management and Processing (SIMAP) is a project sponsored by the Danish National Research Agency aimed implementing Secure Multiparty Computation.
- Secure Multiparty Computation Language - project for development of a 'domain specific programming language for secure multiparty computation' and associated cryptographic runtime.
- VIFF: Virtual Ideal Functionality Framework — Framework for asynchronous multi-party computations (code available under the LGPL). Offers arithmetic with secret shared values including secure comparison.
- Sharemind: a framework for privacy-preserving data mining — A distributed virtual machine with the capability to run privacy-preserving operations. Has a privacy-preserving programming language for data mining tools. Includes developer tools.
- Virtual Parties in SMC A protocol for Virtual Parties in SMC (Secure Multi Party computation)
- MPC Java-based implementation A Java-based implementation of the MPC protocol based on Michael.B, Shafi.G and Avi.W's theorem ("Completeness theorems for non-cryptographic fault-tolerant distributed computation") with Welch-Berlekamp error correcting code algorithm to BCH codes. Supports multiple players and identification of "cheaters" with Byzantine protocol. By Erez Alon, Doron Friedland & Yael Smith.
- SEPIA A java library for SMC using secret sharing. Basic operations are optimized for large numbers of parallel invocations (code available under the LGPL).
- Essential bibliography Secure Multiparty Computation