**CERIAS Tech Report 2007-06**

# PRIVACY-PRESERVING CREDIT CHECKING

by Atallah, M., K. Frikken, and C. Zhang

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

# Privacy-Preserving Credit Checking *

Keith Frikken
Computer Sciences
Department and CERIAS
Purdue University
kbf@cs.purdue.edu

Mikhail Atallah
Computer Sciences
Department and CERIAS
Purdue University
mja@cs.purdue.edu

Chen Zhang
School of Management
Purdue University
zhang153@mgmt.purdue.edu

## ABSTRACT

Typically, when a borrower (Bob) wishes to establish a tradeline (e.g., a mortgage, an automobile loan, or a credit card) with a lender (Linda), Bob is subjected to a credit check by Linda. The credit check is done by having Linda obtain financial information about Bob in the form of a credit report. Credit reports are maintained by Credit Report Agencies, and contain a large amount of private information about individuals. Furthermore, Linda's criteria for loan qualification are also private information. We propose a "privacy-preserving" credit check scheme that allows Bob to have his credit checked without divulging private information to Linda while protecting Linda's interests. We give protocols for achieving the above while: i) protecting Bob's private information, ii) making sure that Bob cannot lie about his credit (thus Linda is assured that the information is accurate), iii) that Linda's qualification criteria are protected, and iv) that the CRA does not learn from the protocols anything other than "Bob requested a loan from Linda". What distinguishes this work from the traditional two-party privacy-preserving framework is (i) the need for secure and privacy-preserving *third-party verification* of the accuracy of the inputs used, and (ii) the fact that the function being computed is private to the lender and should not be revealed to either the borrower or to the above-mentioned third-party verifier. Although we choose to present the techniques of this paper for the credit checking application domain, they have much broader applicability and in fact work for any situation where there is a repository of public and private information about individuals, that is subsequently used for making decisions that impact the individuals (a credit rating agency is but one example of such a repository).

*Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.


*EC'05,* June 5–8, 2005, Vancouver, British Columbia, Canada.


## Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce—*security*

## General Terms

Design, Security

## Keywords

Privacy, secure multi-party computation, e-commerce, secure protocol

## 1. INTRODUCTION

Typically, when a borrower (Bob) wishes to establish a tradeline (e.g., a mortgage, an automobile loan, or a credit card) with a lender (Linda), Bob is subjected to a credit check by Linda. The reason that Linda does a credit check is so that she will have confidence that Bob is trustworthy enough to pay back the loan. In order for Linda to do a credit check on Bob, she contacts a Credit Report Agency (CRA) and obtains a credit report about Bob. It is the purpose of the CRAs to track financial information about an individual, and currently there are three main CRAs: Equifax [1], Experian [2], and TransUnion [3]. When Linda obtains Bob's credit report, she determines if he qualifies for the loan by determining if his credit report satisfies certain criteria that she defines. There is a substantial amount of information in a credit report including: how many tradelines a person has, how much debt a person owes (and to whom), the number of late payments that a person has made in the past, and many other facts about the borrower's financial history [1, 2, 3].

The problem with revealing a borrower's credit report to the lender is that it unnecessarily reveals too much information, often too shady entities who may perform mischief. While many lenders are typically credible organizations, this is not always the case, and even for credible lenders some employees of the lender may be corrupt. Another problem besides the obvious leakage of private information is that this can be an aid for identity thieves as it could serve as a filtering process for them to see who has good credit, and more dangerously, to see who checks their own credit (as this information is contained in the credit report and someone who periodically looks at his own credit report is not as likely to be victimized as someone who never looks at it). Furthermore, Lenders do not need to know all of the information in the credit report, but only need to know if certain conditions are satisfied. One naive way to provide privacy

would be for Linda to reveal her criteria to the CRA who would then report back to Linda if Bob satisfies her criteria. This is not acceptable for a couple of reasons: i) the CRA becomes a bottleneck of the loan-processing system and ii) Linda's criteria for loan qualification are often private. Another possibility is for the CRA and Linda to engage in a two-party private protocol, however the CRA is still a bottleneck in this architecture.

We propose the usage of privacy-preserving protocols to solve the apparent contradictory goals of being able to determine if a borrower satisfies certain criteria while not revealing private information about the borrower's credit report. After engaging in the protocol, all that Bob should learn is whether or not he qualifies for a loan. Furthermore, all that Linda should learn is whether or not Bob qualifies for the loan. However, there is the additional requirement that the borrower should not be able to lie about his credit report, and thus the information needs to be verified by a CRA. It is also desirable that the information flow of the proposed protocol should mimic the way loan-processing takes place today. All that the CRA should learn is that Bob applied for some line of credit with Linda; of course if Bob qualifies and then accepts the loan from Linda, then she would report the information about the loan to the CRA. With such a set of protocols it would be possible for borrowers to have the option of privacy-preserving credit checking.

In this paper we introduce privacy-preserving protocols for achieving the above-defined problem. The protocols are *efficient* in that they require communication and computation (modular exponentiations, which are typically the slowest operation) proportional to the size of the credit report and the policy of the lender, and while the computational overhead for the CRA is much larger than in the current non-private setting, much of this work can be precomputed off-line. Furthermore, our protocols are *simple* in that they mimic the way loan-processing takes place today with the only difference being that a borrower must register with the CRA(s) if he wishes to use such protocols. Finally, the protocols are *private* in that they satisfy the privacy requirements outlined above and are *correct* in that the lender has certainty that the borrower's information is accurate. What distinguishes this work from the traditional two-party privacy-preserving framework is (i) the need for secure and privacy-preserving *third-party verification* of the accuracy of the inputs used, and (ii) the fact that the function being computed is private to the lender and should not be revealed to either the borrower or to the above-mentioned third-party verifier. Somewhat surprisingly, our solution relies on the beneficiary of the loan application (the borrower) to honestly carry out the computation of the outcome: What keeps him honest is the fact that the encoding of that outcome (as well as of all the other intermediary results) is known only to the lender; the borrower can only hurt himself by not following the protocol.

Although we choose to present the techniques of this paper for the credit checking application domain, they have much broader applicability and in fact work for any situation where there is a repository of public and private information about individuals, that is subsequently used for making decisions that impact the individuals (a credit rating agency is but one example of such a repository). Other repositories contain information collected and maintained for different purposes than credit ratings, e.g., information about peoples' shopping habits and preferences, level of education, income, number and age of their children, history of contributions and donations to charities, etc. The information in these repositories is typically collected without the active and knowing participation of the individuals concerned, and the repositories often contain wrong information that the individual never gets a chance to correct. Moreover, these repositories are often used for purposes the individual would not approve of, such as junk mail, spam, and more nefarious purposes such as identity theft. A privacy-respecting repository of the kind we envision may, because its reputation and business model are based on offering privacy to its customers, cause individuals to actually contribute additional (and accurate) information to the repository, thereby increasing its value over the competition's. Such a repository may have information (backed with proof or certification) that John Doe is fluent in French, has a degree in Mechanical Engineering, has extensive experience in the analysis of seismic data, all in addition to the usual financial and personal information stored in today's CRAs. The lender-like entities who use such a repository would inquire about a specific individual only with the individual's permission (e.g., when he applies for a loan, a security clearance, a permission to buy restricted merchandise) would benefit not only from the more accurate information in the repository, but also from the fact that no one (including the repository) will know the precise nature of their credit-like inquiries about an individual; this is especially important when screening a candidate for a security-sensitive position, for access to restricted online material, for eligibility to a special discount, or any other situation where there is fear that a malevolent individual could "game" the screening system if such an individual knew the precise screening criteria and algorithm.

The rest of this paper is organized as follows. In Section 2 we formally define the problem and outline the privacy requirements of our protocols. In Section 3 we survey the related work in this area. In Section 4 we define building blocks that are needed by our protocols. In Section 5 we define a preliminary protocol for privacy-preserving credit checks, and this protocol is extended in several ways in Section 6. Finally, our results are summarized and future directions are discussed in Section 7.

# 2. PROBLEM DEFINITION

## 2.1 Problem Definition

The credit report essentially is a set of of booleans (e.g., "has the borrower ever filed for bankruptcy in the past?") and a set of integers (e.g., the amount of debt owed). A credit report contains other types of information (such as to whom the current debt is owed), but this information is not necessary to make a decision on loan qualification (the type of tradeline might be useful, but this can be encoded as an integer). The credit report can be represented as a set of boolean *attributes*, where the value is true if a person satisfies that attribute. For boolean information, the encoding is trivial; for example "has person $X$ ever been bankrupt?" would be an attribute. To encode integer values, an attribute would be defined for each bit in the binary representation of the value; an example in this case would be "is the 5th bit of person $X$'s debt true?". The lender has many *criteria* that it looks at for loan qualification; the criteria can be computed from one or more attributes. Ex-

ample criteria include: "does person $X$ have any liens on their home?", "is person $X$'s debt below some threshold?", and "is the number of tradelines that person $X$ has had in the past above some threshold?". If an attribute or a criterion is true for a person, we say that the person *satisfies* the attribute or criterion. We denote the attributes for a credit report by $a_1, \ldots, a_m$ and the criteria by $c_1, \ldots, c_n$, where each criterion is a function of a subset of the attributes. We define a boolean function *sat* where $sat(a_i)$ and $sat(c_i)$ represent whether or not the person satisfies attributes $a_i$ and criterion $c_i$ respectively (i.e., $sat(c_i) = 1$ if $c_i$ is satisfied and is 0 otherwise). Finally, the lender has a *policy* for determining if a borrower qualifies for a specific loan. This policy is some function of $sat(c_1), \ldots, sat(c_n)$.

We make several assumptions in our initial protocol for this problem (see Section 5); many of these assumptions are relaxed in Section 6. The assumptions include:

1. *Bounded Credit Report Size:* We assume that each integer value in the credit report can be bounded by some value, and that the total number of entries in the credit report can also be bounded for all people. To protect the size of the credit report the CRA must insert dummy entries into each credit report to make all credit reports not distinguishable by size.
2. *Accurate CRA Assumption:* It is assumed that the CRA is trusted by the lenders to provide accurate information, however the CRA is not a "trusted-third party" in that it should not learn information about a lender's policy. We discuss techniques for handling a malicious CRA that colludes with borrowers to probe a lender's policy in Section 6.5.
3. *Single CRA Assumption:* Initially, we assume that there is a single CRA which has all information required by the lender to make a decision about a borrower, however this is not realistic. There may be multiple CRAs with the same information and it is possible that there could be a discrepancy between the CRAs' information. We discuss extensions of our protocols to multiple CRAs in Section 6.5.
4. *Criteria Assumption:* We assume that the criteria come in one of two forms: i) a single attribute criteria or ii) a comparison against a threshold criteria. This captures most (if not all) common things done to a credit report by the lender. It is not difficult to extend our protocols to other type of criteria, and we discuss some mechanisms for this in 6.3.
5. *Known Criteria Assumption:* We assume that the lender does not mind revealing the general form of his or her criteria. For example, the lender is not worried about revealing that it has a criterion that compares a person's debt against a threshold, but the lender does not want to reveal the threshold. In many cases a global set of criteria may be used by multiple lenders without revealing specific information. This is generalized in Sections 6.3.
6. *Policy Assumption:* Initially, we assume that the policy is of the form: a borrower qualifies for a loan if he satisfies at least $t$ criteria (where $t$ is a private threshold defined by the lender). While this is not a realistic form of policy in many cases, it is a preliminary step. We explore generalized policies in Sections 6.1, 6.2, and 6.3.
7. *Passive Behavior Assumption:* Initially, we assume that the parties are passive (i.e., honest-but-curious) in that they will engage in the steps of the protocols, but will try to learn additional information. We generalize our protocols to a malicious adversary model (for borrowers and lenders) in Section 6.4. We discuss techniques for handling a malicious CRA that colludes with borrowers to probe a lender's policy in Section 6.5.

## 2.2 Privacy and Correctness Requirements

We now define the privacy and correctness requirements of our protocols.

**Borrower:** All that the borrower should learn is whether or not he qualified for the loan, and nothing else (except of course what he can deduce from this outcome and from his knowledge of his own credit report, which is unavoidable). Furthermore, the borrower should not be able to lie about his credit report (i.e., his values must be the values stored at the CRA).

**Lender:** The lender should learn whether or not the borrower qualified for the loan, and nothing else (except of course what he can deduce from this outcome and from his knowledge of his own policy, which is unavoidable).

**CRA:** The CRA should learn only that a specific borrower is applying for a loan with a specific lender. He should learn neither the lender's policy, nor whether the borrower qualified for the loan. Of course, in practice, if the loan is approved and does happen, the CRA will be informed by the lender of this fact so he can update the borrower's record – but if the loan is approved yet does not happen for some reason (e.g., the borrower and lender could not agree on an interest rate) then the CRA never learns whether the loan was approved or not.

# 3. RELATED WORK

Secure Multi-party Computation (SMC) was introduced in [19], which contained a scheme for secure comparison; suppose Alice (with input $a$) and Bob (with input $b$) desire to determine whether or not $a < b$ and without revealing any information other than this result (this is referred to as "Yao's Millionaire Problem"). More generally, SMC allows Alice and Bob with respective private inputs $a$ and $b$ to compute a function $f(a, b)$ by engaging in a secure protocol for some public function $f$. Furthermore, the protocol is private in that it reveals no additional information. By this what is meant is Alice (Bob) learns nothing other than what can be deduced from $a$ ($b$) and $f(a, b)$. Elegant general schemes are given in [8, 7, 4, 6] for computing any function $f$ privately. However, these general solutions are considered impractical for many problems, and it was suggested in [9] that more efficient domain-specific solutions can be developed. Also, a constant round malicious-adversary protocol has been introduced recently [11]. Furthermore, there has been recent work on developing an implementation of the general SMC results [13]. The framework for the general results of SMC is similar to the work in this paper, but our framework is different for a couple reasons: i) the inputs of the borrower need to be verified by a third party, and ii) the lender's policy function needs to be hidden. Our results are also similar to some work on private auctions [16] where there was a third party (an auction issuer) involved in the computation, but in this case the third party was an active member of the protocol.

There are two areas that are related to privacy-preserving credit checking, and we outline these problems here:

1. *Selective Private Function Evaluation (SPFE):* SPFE was introduced in [5] whose goal is for one party to compute a private function over a subset of another party's database without revealing the function. This is similar in that it computes a hidden function, but the problem proposed in this paper is not a special case of SPFE, because: i) the work in [5] focuses on simple functions like summation of a subset of the database and ii) it does not compute the functions in a third-party verifiable manner.
2. *Automated Trust Negotiation:* The goal of this work is to allow people to authenticate themselves for access control purposes, and while much of the work assumes that the policies can be discussed publicly, there has been some work on keeping policies private [12, 10]. However, that type of privacy was different than what the credit checking problem requires: The privacy in [12, 10] protects the policies against entities that do not satisfy the policy, but if an entity satisfies the policy then it learns the policy (or at least a substantial amount of information about the policy).

# 4. BUILDING BLOCKS

## 4.1 Primitives and Notations

We use the following primitives and notations in this paper:

1. *Encryption/Decryption:* To represent encryption and decryption we use the functions $Enc$ and $Dec$, where $Enc(m,k)$ ($Dec(m,k)$) represents the encryption (decryption) of message $m$ with key $k$. Note that this notation can be used for symmetric key systems as well as asymmetric key systems.
2. *Oblivious Transfer:* There are many equivalent definitions of Oblivious Transfer (OT). In this paper we use the definition of chosen 1-out-of-$k$ OT to be that Alice has a set of items $x_1, \ldots, x_k$ and Bob has an index $i \in \{1, \ldots, k\}$. The OT protocol allows Bob to obtain $x_i$ without revealing any information about $i$ to Alice and without revealing any information about other $x_j$ ($j \neq i$) values to Bob. For a high level survey of OT the reader is referred to [18] and for more recent developments see [14, 15].

## 4.2 Scrambled Circuit Evaluation

To make the paper self-contained, we review scrambled circuit evaluation in Appendix A. The reader unfamiliar with the results in this area is encouraged to read the appendix as it will clarify our protocols substantially. We utilize an efficient mechanism for secure two party circuit simulation in constant rounds (as in [20]). This can be extended to more than two parties (as in [17]). For two parties, the protocol requires a 1-out-of-2 OT per input wire, and it requires $O(1)$ evaluations of a pseudorandom function (such as AES) per gate in the circuit.

## 4.3 Oblivious Gates/Circuits

In the construction of a scrambled circuit, the gates are constructed for some publicly defined function $f$; typically a verifiable computation requires that the generator of the circuit prove to the evaluator that the circuit is well-formed. However, there are cases where not having the function be publicly defined can be useful, especially when the function is private. In this case, the construction defined in the previous section can easily be modified to use *oblivious gates* where the evaluator does not know the function that each gate computes. This has many useful applications including:

1. It is easy to construct an oblivious comparison circuit (i.e., one that can compute $=$, $\neq$, $>$, $<$, $\geq$, and $\leq$ without revealing which comparison is done) with size proportional to the number of bits in the values.
2. A binary tree of oblivious gates (with inputs $a_1, \ldots, a_n$) can be used to compute many useful functions (without revealing which function is being used) including:
   (a) $\bigwedge_{i=1}^{n} a_i$, $\bigvee_{i=1}^{n} a_i$, $\bigoplus_{i=1}^{n} a_i$, etc.
   (b) For any subset of the values $S$, $\bigwedge_{i \in S} a_i$,$\bigvee_{i \in S} a_i$, $\bigoplus_{i \in S} a_i$, etc.
   (c) Other functions like: for a subset $S_1$ of the first half of the values and another subset $S_2$ of the second half of the values, the function $\bigvee_{i \in S_1} a_i \wedge \bigvee_{i \in S_2} a_i$.
3. By using oblivious binary trees on the results of other binary trees of oblivious gates, a wide variety of policies can be computed including any monotonic circuit and many other useful structures.

# 5. A PRELIMINARY PROTOCOL

In this section we introduce a protocol for achieving privacy-preserving credit checking. This is not the final protocol, but rather it should be viewed as a "warmup" for the better protocols that come later on. There are many extensions and improvements to this protocol, but to simplify this exposition we postpone their discussion until Section 6. Furthermore, we use a modular approach to describe the protocol, but the round efficiency can be greatly improved by combining many of the phases (which we briefly discuss in Section 5.2). To simplify this protocol we make several assumptions that are outlined in Section 2.1.

## 5.1 Protocol Description

**Phase 0: Setup:** For each borrower, the CRA has a credit report represented by attributes $a_1, \ldots, a_m$. When Bob registers with the CRA, the CRA and Bob establish a shared encryption key (call it $k$).

**Phase 1: Loan Request:** When Bob attempts to establish a loan with Linda, she contacts the CRA to obtain information about Bob's credit report. When the CRA receives a request for a credit report, the CRA generates two keys (for a symmetric encryption scheme) for each attribute (note that these values can be pre-computed). Let the pair for attribute $a_i$ be denoted by $(e_i^0, e_i^1)$. The CRA sends to the Lender the following two things: i) All of the key values $e_1^0, \ldots, e_m^0, e_1^1, \ldots, e_m^1$, and ii) the key values encrypted with the key $k$ (i.e., $Enc((e_1^{sat(a_1)}, \ldots, e_m^{sat(a_m)}), k)$). The Lender stores the first part of this message, but forwards the second part to the Borrower who decrypts the value and stores the result. In what follows, the goal is to build a circuit that determines if the Borrower qualifies for the loan. Thus we need to address how to obtain the circuit's input, how to build a circuit, and how to obtain the results from the circuit.

**Phase 2: Attribute Input:** In this phase Linda chooses two keys (for a symmetric encryption scheme) for each at-

tribute, and Bob learns one of the keys if he satisfies the corresponding attribute but learns the other key if he does not satisfy the attribute. Let the keys for attribute $a_i$ be denoted $s_i^0$ and $s_i^1$, where the borrower will learn $s_i^{sat(a_i)}$. For each attribute $a_i$, Linda computes an ordered pair $(Enc(s_i^0, e_i^0), Enc(s_i^1, e_i^1))$. She sends all of these pairs to Bob. For each attribute $a_i$, Bob computes $Dec(Enc(s_i^{sat(a_i)}, e_i^{sat(a_i)}), e_i^{sat(a_i)})$. After this step Bob has only the keys that correspond to his attributes. Note that it is possible for Linda and Bob to skip the previous step by using the values $(e_i^0, e_i^1)$ as Linda's output and $e_i^{sat(a_i)}$ as Bob's output for the phase. However, this does not allow Linda to precompute circuits for loan qualification.

**Phase 3: Determining Satisfiability of Criterion:** For each criterion $C$, we need to be able to compute $sat(C)$. A criterion $C$ has a certain set of attributes, and for each attribute there are two possible keys (one corresponding to Bob satisfying the attribute and another to Bob not satisfying it), and his input to the circuit will be his corresponding keys for the attributes. In order to be able to use a circuit on the output of this satisfiability computation the output of this part should be one of two keys (which Linda chooses), where one corresponds to a borrower that satisfies the criterion and the other corresponds to a borrower that does not satisfy the criterion. If the criterion is a single attribute, then this is trivially done by using the attribute's input key as the output key. If the criterion is a comparison, then a standard circuit for comparison can be used. Thus either type of criterion can be implemented in communication proportional to the number of attributes in the criterion.

**Phase 4: Result Combination:** After the previous phase, for each criterion Linda has two keys and Bob has one of these keys. During this phase Linda and Bob evaluate a circuit that determines if he qualifies for the loan. Recall our assumption that Bob qualifies if he satisfies at least $t$ criteria, in other words Bob qualifies if $\sum_{i=1}^{n} sat(c_i) > t$ for some $t \leq n$. This can easily be done with the addition circuit, followed by a comparison circuit with value $t$. Note that the size of this circuit is of size proportional to number of criteria.

**Phase 5: Obtaining Result:** After the previous phase, Bob has one of two keys $k_0$ or $k_1$ (both of which were generated by Linda). Key $k_1$ corresponds to Bob qualifying for the loan, and key $k_0$ corresponds to Bob not qualifying for the loan. The question now becomes how does Bob prove he qualified for the loan to Linda. Bob proves this to Linda by sending her the key that he receives (recall that we assume for now the honest-but-curious model). When Linda receives $k_1$ from Bob, she is convinced that he qualified for the loan, and if instead she receives $k_0$ then she denies him the loan.

## 5.2 Summary of the Protocol

We now state the above protocol in more concise terms, leaving out the details but describing the information flow:

1. To use the privacy-preserving credit check, Bob registers with the CRA and sets up an account for this service, the CRA and Bob establish a private key (call it $k$).
2. When Bob requests a loan from Linda, she contacts the CRA with a request to run a protocol that determines for her whether Bob's credit report satisfies her criteria (without revealing the report itself to her).
3. The CRA sends Linda a set of encryption keys; there are two keys for each attribute (one corresponding to each possible value of the attribute). The CRA also sends a single decryption key for each attribute (the one that corresponds to Bob's actual attribute) encrypted with the value $k$.
4. Linda builds a scrambled circuit (see Appendix A) that determines if Bob qualifies for a loan. To obtain the input for the circuit, she encrypts the encodings (again see Appendix A) with the encryption functions that she received from the CRA. She sends Bob the following items: the circuit, a set of messages from which he can obtain the input wire encodings of the circuit, and the message from the CRA with the decryption keys.
5. Bob decrypts the decryption keys and obtains the inputs to the circuit using these keys. He then evaluates the circuit and upon finding the output, sends the result to Linda.
6. Linda tells Bob whether or not he qualifies for the loan.

The communication complexity of the above is proportional to the number of attributes, and requires one round of messages between Linda and the CRA and another between Bob and Linda. The CRA must generate the key pairs for each attribute, but this is not expensive because it is just random number generation. The CRA must also encrypt the decryption keys, but this can be done using a fast symmetric key encryption system (such as AES). Linda needs to do two symmetric encryption operations for each attribute and needs to generate a circuit for the credit check, but the latter can be pre-computed. Bob needs to perform a single symmetric key operation for each attribute, and then needs to evaluate a circuit (where the circuit has size proportional to the number of attributes), but since this is in the passive model these operations can be a symmetric key system. The correctness of the protocols trivially follows from the correctness of secure circuit evaluation.

## 5.3 Security of Protocols

We now discuss the security of the above protocol.

**Borrower:** During Phase 2, the borrower has only one key per attribute and thus he can obtain at most one input for the circuit. Thus the borrower can find at most one value for the output of the circuit (assuming he is computationally bounded).

**Lender:** This design goals for the lender are trivially satisfied in the passive model as the lender learns only a single value after the computation of the circuit, and because the message from the CRA is encrypted with a key unknown to Linda.

**CRA:** The CRA sees only a request from Linda about Bob.

# 6. EXTENSIONS

## 6.1 Weighted Threshold Based Policies

In the protocol in Section 5, Bob qualifies for a loan if he satisfies at least $t$ out of $n$ criteria (where $t$ is a private parameter defined by Linda). This can be generalized to add more flexibility to the system. For example, some criteria may be more important than other criteria. In the case where the criteria are globally defined (by and for all lenders) some of the criteria may not apply to Linda. In this section we propose a weighted threshold based policy. Each crite-

rion $c_i$ is assigned a weight $w_i$, where $w_i \in \{0, \ldots, N-1\}$. Bob qualifies if $\sum_{i=1}^{n}(sat(c_i)w_i) > t$ for some threshold $t$ (again $t$ is privately defined by Linda). The techniques in Section 5 can be used, however between phases 3 and 4 another step must then be added to replace $sat(c_i)$ with $w_i sat(c_i)$. Essentially each criterion value must be expanded to a $(\log N)$-bit scrambled value. This can easily be done with a 2-ary gate having $(\log N)$ outputs, and size $O(\log N)$. The circuit is also changed to add the larger values. Note that the circuit size is now $O(m + n \log N)$ where $m$ is the number of attributes and $n$ is the number of criteria.

## 6.2 Combinatorial Circuit Based Policies

The previous notion of a weighted policy allows the definition of many policy types, but it does not allow policies of the following forms:

1. "(At least 3 out of these specific 4 criteria) and (At least 2 out of these specific 5 criteria)".
2. "Criterion A or (Criterion B and Criterion C), but not (Criterion A and Criterion B).

Thus if the policy has a complex structure, then there may not be a way to represent it with a weighted threshold system, and even if there is such a representation, it may not be intuitive. In this section, we propose a more generic structure that is more powerful and more intuitive. For this representation the lender defines several "super-criteria", which are expressible as threshold policies, weighted threshold policies, or other combinatorial based circuit policies. The lender defines a set of these super-criteria and then combines the results using a binary tree of oblivious gates. Clearly, anything that can be expressed with threshold policies or weighted threshold policies can be expressed with this mechanism, and the examples given above can easily be expressed with this representation. This reveals the general structure of the policy but none of its specifics – actual values of thresholds, cutoffs, etc. (as discussed in Section 4.3). However, to hide the structure, padding the circuit with "dummy" entries can further hide the structure of the circuit.

## 6.3 General Policies or Criteria

While the combinatorial circuit-based policies described in the previous section allow the lender to represent a large class of policies, it is still limited. In addition, complicated criteria composed of multiple attributes cannot be represented. To handle such requirements one could extend the techniques to represent an arbitrary policy (by using an $n$ input gate that represents the policy). However, such an extension would require exponential communication/computation.

## 6.4 Malicious Adversaries

In our previous protocols we assumed that the behavior of the parties is passive or honest-but-curious (i.e., that the parties will follow the protocols, but will try to glean additional information). We postpone discussion of malicious CRAs until Section 6.5, and thus in this Section we assume that the CRA is honest (as it has very little incentive to behave otherwise). However, in this section we look at what happens when Linda or Bob deviates from the protocol.

- As a borrower, there is little Bob can do to deviate from the protocol. The only type of attack we worry about is Bob falsely qualifying for a loan, and since his only messages are his loan request and the message which states whether or not he obtained the loan (which he cannot forge with non-negligible probability), a malicious Bob has no extra advantage.
- Linda violates Bob's privacy if she learns additional information about Bob's credit report. She can do some things to try to probe Bob' credit, and there are essentially three ways in which she can do this: i) abort the protocol after receiving the results, ii) having the output gate output more than two things, iii) creating a malformed circuit. We now examine each of these in more detail (note that this solution does not require the gates to be some form of encryption that can be verified with Zero-Knowledge proofs):
  1. Linda could create a circuit that reveals something other than whether or not Bob qualifies, and then she could abort the protocol after receiving Bob's response. This "abort" could take the form of a contrived network failure, and Bob's only choice would be to run the protocols again with Linda. To avoid this we propose a modification of Phase 5 of the protocol. Instead of revealing a random string, the circuit reveals one of two random keys. As part of the circuit Linda also sends a message, which is an encryption with the "qualify" key of a signed message stating that Bob qualifies for the loan. Now, Linda cannot do the above abort attack, because Bob can resend the signed message.
  2. Linda could make the output of the gate have more than two outputs, which would reveal additional information to her about Bob's credit report. However, this is already handled by the solution offered for the previous problem, assuming that it is hard to generate two different keys that encrypt to the same value for two different signatures.
  3. Linda could send a circuit that is malformed (i.e., some inputs will lead to an output, but other sets of inputs lead to Bob not being able to compute the output by not being able to decrypt any message at a gate). If Bob reports a problem, then a malicious Linda has more information since there are now two fail states. To avoid this Bob, sends the same message if there is an error in the circuit as if he gets to the output gate but does not qualify.

## 6.5 Multiple CRAs

The assumption that there is a single CRA is not realistic because in practice there are multiple CRAs. If the information is used in independent ways, then this can be handled by a trivial extension to our protocols. Thus we consider the case where the lender uses multiple CRAs and wants to make sure that the borrower satisfies the criteria at all of the CRAs (which may have conflicting information). When there are multiple sources for the same information, a lender would like to add "defense in depth", and we propose a conflict resolution strategy that resolves conflicts with the safest possible approach. Essentially the circuit computes the criterion for each of the CRAs, and then combines them into a single value. If the criterion is positive (i.e., it helps Bob get the loan) then the lender would want to make sure that

all of the CRAs information states that Bob satisfies that criteria, which can easily be done with a set of AND gates. If the criterion is negative (i.e., it hurts Bob's chances of getting the loan) then the safest approach that Linda could take is that Bob has the property in question if one or more of the CRAs claim he has that property, and this can easily be computed with a set of OR gates. Clearly, either of these cases can be done with a binary tree of oblivious gates.

In the single CRA case, a single malicious CRA that colludes with one or more borrowers to probe a lender's policy is unavoidable. However, when multiple CRAs are used and the policy is to use the safest possible approach (as outlined above), then unless all of the CRAs are corrupt, the colluding parties are limited in what they learn about the lender's policy. The probing of the malicious CRAs is limited to credit reports that are worse than the borrower's credit report with the non-colluding CRAs.

# 7. CONCLUSIONS AND FUTURE WORK

Typically, when a borrower (Bob) wishes to establish a tradeline (e.g., a mortgage, an automobile loan, or a credit card) with a lender (Linda), Bob is subjected to a credit check by Linda. The credit check is done by having Linda obtain financial information about Bob in the form of a credit report. Credit reports are maintained by Credit Report Agencies (CRAs) and contain large amounts of private information about individuals. Furthermore, Linda's criteria for loan qualification are confidential. Clearly, the current mechanisms for credit checking violates the privacy of borrowers. In this paper we introduce privacy-preserving protocols that allow the lender to determine if a borrower's credit satisfies certain criteria without revealing the credit report to the lender, without revealing the specific details of the criteria to the borrower, and in a manner that is verifiable by the CRA yet does not reveal to the CRA either the criteria or the outcome (i.e., whether the criteria are satisfied or not). The protocols are *efficient* in that they require communication and computation (modular exponentiations, which are typically the slowest operation) proportional to the size of the credit report and the policy of the lender, and while the computational overhead for the CRA is much larger than the non-private setting, but much of this work can be pre-computed off-line. Furthermore, our protocols are *simple* in that they have a communication architecture that mimics the current way that credit checks are done, the only exception being that a borrower must register with the CRA(s) if he wishes to use such protocols. Finally, the protocols are *private* in that they satisfy the privacy requirements outlined above, and are *correct* in that the lender has certainty that the borrower's information is accurate. These protocols largely benefit the borrowers, and thus in order for the CRAs and the lenders to adopt such systems the borrower would have to pay extra for such a service. However, since most members of the population are not that concerned about privacy, there would be a relatively small number of users of the privacy service, and the cost would be higher than if it had been spread over a huge user population. We conjecture that it would require legal mandate for such protocols to be put into place, unless identity theft gets much worse, or some scandal surfaces about blatant misuse of credit records. The techniques of this paper for the credit checking application domain, they have much broader applicability and in fact work for any situation where there is a repository of public and private information about individuals, that is subsequently used for making decisions that impact the individuals. As future work we propose the additional problem of how to incorporate multiple entities' data (employment records, insurance information, etc) with this methodology to provide even more privacy for the borrower. Also, before this is a functional system there are many non-trivial interface questions (for the description of credit reports and the description of policies) that must be resolved.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their useful comments and suggestions.

# 8. REFERENCES

[1] Equifax. `http://www.equifax.com`
[2] Experian. `http://www.experian.com`
[3] TransUnion. `http://www.transunion.com`
[4] Michael Ben-Or and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM Press, 1988.
[5] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright. Selective private function evaluation with applications to private statistics, 2001.
[6] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM Press, 1988.
[7] 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.
[8] Oded Goldreich. Secure multi-party computation. Working Draft, 2000.
[9] Shafi Goldwasser. Multi party computations: past and present. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6. ACM Press, 1997.
[10] Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and Hilarie Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society*, October 2003.
[11] J. Katz and R. Ostrovsky. Round optimal secure two-party computation. In *CRYPTO 04*, 2004.
[12] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*. ACM Press, July 2003.
[13] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - a secure two-party computation system. In *Proceedings of Usenix Security*, 2004.
[14] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254. ACM Press, 1999.
[15] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages

448–457. Society for Industrial and Applied Mathematics, 2001.

[16] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139. ACM Press, 1999.

[17] P. Rogaway. *The Round Complexity of Secure Protocols*. Ph.d. thesis, MIT, 1991. Available at `http://www.cs.ucdavis.edu/ rogaway/papers`.

[18] Bruce Schneier. *Applied Cryptography – Protocols, algorithms, and souce code in C*. John Wiley & Sons, Inc., 1996.

[19] A.C Yao. Protocols for secure computation. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[20] A.C Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

# APPENDIX

# A. SECURE CIRCUIT EVALUATION

In this protocol, one party is a *generator* of a scrambled circuit and the other party is an *evaluator*. The generator creates a scrambled circuit where each wire of the circuit has two encodings (one for each possible value of the wire), and the gates contain information that allow an evaluator to obtain the encoding of the output wire given the encoding for the gate's input wires. What makes this a private circuit evaluation is that the evaluator learns the encoding corresponding to his input for each input wire), and thus learns only one encoding per wire.

We now describe in more detail a protocol for Scrambled Circuit Evaluation.

- Circuit Generation: For each wire in the circuit $w_1, \ldots, w_n$, the generator creates random encodings for the wires (in this case the encodings are a key for a trapdoor function or are a random seed that can be used by a pseudo-random number generator to generate such a key). We denote the encodings of 0 and 1 for wire $w_i$ respectively by $w_i[0]$ and $w_i[1]$. The question now becomes how to develop gates that allow an evaluation of the circuit. To create a 2-ary gate for a function $f$ with input wires $w_i$ and $w_j$ and with output wire $w_k$, the gate consists of four messages (where $m$ is a publicly defined marker, used to recognize when an item has been successfully decrypted):
  1. $Enc(Enc(m||w_k[f(0,0)], w_j[0]), w_i[0])$
  2. $Enc(Enc(m||w_k[f(0,1)], w_j[1]), w_i[0])$
  3. $Enc(Enc(m||w_k[f(1,0)], w_j[0]), w_i[1])$
  4. $Enc(Enc(m||w_k[f(1,1)], w_j[1]), w_i[1])$

  Note that the scrambled gate is these messages in a randomly permuted order. Clearly the scrambled circuit with fan-in 2, can be represented in size proportional to the size of the circuit. It is a natural extension of this to create an $n$-ary gate with $m$ outputs that has size proportional to $2^n m$, and while this should be avoided for large $n$ due to the exponential blowup in gate size, there are situations where this is useful.
- Learning Input Wires: In order to evaluate a circuit the evaluator must know the values of the input wires. For input wires corresponding to the generator's inputs, the generator simply sends the evaluator the encoding of each of his inputs. For input wires corresponding to the evaluator's inputs, the two parties engage in a 1-out-of-2 OT where the two "messages" are the generator's encodings of 1 and 0, and the evaluator gets the encoding corresponding to his input for that wire.
- Evaluating the Circuit: To evaluate a gate the evaluator decrypts each message in the gate with the keys that it has for the input wires. Only one of these decrypted messages will contain the marker $m$ (the others will look random), and thus the evaluator will learn exactly one encoding for the output wire (he will know it is the correct value for that wire, but of course he cannot tell whether it corresponds to a 0 or a 1).
- Learning the Result: If the goal is to have the evaluator simply learn the result, then it is enough for the generator to tell the evaluator both encodings of the output wire.

With passive adversaries it is enough for the trapdoor function used for gate construction to be a strong encryption function (such as AES). Thus any circuit with constant fan-in can be simulated privately in time proportional to the size of the circuit. We now briefly review the circuit size for various functions that are needed for credit checking:

1. *Comparing two $k$-bit values:* Requires only $O(k)$ gates in a 2-ary circuit.
2. *Adding two $k$-bit values:* Requires only $O(k)$ gates in a 2-ary circuit.
3. *Adding $k$ single bit values:* Requires only $O(k)$ gates in a 2-ary circuit.