Private Intersection of Certified Sets

Jan Camenisch[1⋆] and Gregory M. Zaverucha[2⋆⋆]

[1] IBM Research
Zürich Research Laboratory
CH-8803 Rüschlikon
jca@zurich.ibm.com

[2] Cheriton School of Computer Science
University of Waterloo
Waterloo ON, N2L 3G1, Canada
gzaveruc@cs.uwaterloo.ca

Abstract. This paper introduces certified sets to the private set intersection problem. A private set intersection protocol allows Alice and Bob to jointly compute the set intersection function without revealing their input sets. Since the inputs are private, malicious participants may choose their sets arbitrarily and may use this flexibility to affect the result or learn more about the input of an honest participant. With certified sets, a trusted party ensures the inputs are valid and binds them to each participant. The strength of the malicious model with certified inputs increases the applicability of private set intersection to real world problems. With respect to efficiency the new certified set intersection protocol improves existing malicious model private set intersection protocols by a constant factor.

Keywords: private set intersection, secure two-party computation, certified sets

1 Introduction

The problem of private set intersection is the following. Alice and Bob hold sets S_A and S_B, respectively. They would like to jointly compute the intersection, in such a way that reveals as little as possible about S_A to Bob and S_B to Alice. In other words, both Alice and Bob should learn $S_A \cap S_B$ but nothing more.

While this task could be completed with general secure multiparty techniques, it is far more efficient to have a dedicated protocol, especially since the number of communication rounds will be constant. A number of such protocols exist in the literature. A problem common to all previous protocols is that the inputs S_A and S_B can be chosen arbitrarily by Alice and Bob. Our protocols allow Alice and Bob to use only certified sets, that is, sets which have been approved by a trusted party. The trusted party authorizes the set once for each party, then does not participate in the protocol.

⋆ Work funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483.

⋆⋆ Work done while visiting IBM Research. Supported by an NSERC PGS scholarship.

We also consider a variant of private intersection, when A and B wish to compute the cardinality of the set intersection. This is referred to as the *private intersection cardinality* problem.

Private set intersection protocols may find applications in online recommendation services, medical databases, and many data related operations between companies, which may even be competitors. An example from the law enforcement field is given by Kissner and Song [22]; suppose a law enforcement official has a list of suspects and would like to know if any of them are customers of a particular business. To protect the privacy of the other customers, and keep the list of suspects private, the business and the law use a private set intersection protocol to learn only those names appearing on both lists.

Motivation for certified private sets. The goal of certifying the private sets of participants is to restrict their inputs to "sensible" or "appropriate" inputs. This reduces the strength of a malicious participant.

Suppose Bob is malicious in the following sense; he follows the protocol, but wishes to learn as much about S_A as possible. Bob's strategy is to populate a set S'_B with all of his best guesses for S_A and to have $|S'_B|$ be as large as Alice will allow. This maximizes the amount of information Bob learns about S_A.

In the extreme case, Bob may claim S_B contains all possible elements, which will always reveal S_A. He may also vary his set over multiple runs of the protocol, in order to learn more information over time. These attacks are even more powerful when the protocol can be executed anonymously. Note that all this behaviour is permitted in any model which allows the participants to choose their inputs arbitrarily.

The weakness of models which allow arbitrary inputs reduces the practicality of private set operations. The following examples are made possible by the use of certified private sets. A certification authority (CA) is a trusted party who certifies that each participant's set is valid. Once the sets are certified, the CA need not be online. For example, suppose companies want to perform set operations on their financial data. Each company uses a different, but trusted, accounting firm who certifies the data. The companies can then perform as many operations with as many other companies with their certified data.

Since our approach to certifying sets shares a lot with anonymous credentials, this area may also benefit from our work. Credential holders may treat values in their certificates as sets, and intersect them. For example, two pseudonymous/anonymous users may intersect their credentials to determine they live in the same city and were born in the same year. As another example, they may determine whether their ages are within y years by intersecting sets of integers $\{age - y, \ldots, age, \ldots, age + y\}$, where age is the certified value from the credential. This facilitates privacy-enhanced social networking.

Credential holders may also prove things such as at "I satisfy at least two of the following five conditions". This is effected by privately intersecting the credential with a list of the conditions. The two satisfied conditions may be revealed (by using set intersection) or kept private (by using set intersection cardinality).

This work may also find applications in revocation strategies for anonymous services. The revocation list is private and certified to maintain the privacy of revoked users. When an anonymous user authenticates, their singleton set (their certified ID) is intersected with the revocation list. If the user has been revoked, the intersection contains their ID, otherwise the service provider is assured they are not on the revocation list. Using an intersection cardinality protocol would keep the ID hidden and only reveal whether the user was on the list or not. Since the revocation list is certified, the user is assured that the server does not populate it arbitrarily, to de-anonymize users.

Certified sets are also useful in the suspect-list example of Kissner and Song. Privacy conscious businesses will only reveal information about customers when law enforcement has a warrant for such information (signed by a judge). In this case, the judge digitally signs the list of suspect names for the law enforcement agent. This convinces the business owner that information is only being revealed in accordance with a warrant. On the other hand, the business may also get their customer list certified by a credit card company, bank or tax authority to convince the law enforcement agency that the list is complete and contains valid names.

Related Problems. A number of previously studied problems in the literature are similar to private set intersection. We list them to point out how they differ from the current problem.

A *secret handshake* protocol allows Alice and Bob to confirm that they are both members of the same group (a spy agency, for example). At the end of the protocol, both Alice and Bob learn that either they are both members of the group, or nothing at all. This problem can be viewed as private set intersection (or intersection cardinality) with sets of size one.

The *socialist millionaires' problem* is very similar; two parties would like to determine whether they have the same amount of money, or no information if they have different amounts of money [4]. This problem can also be solved by private intersection of singleton sets, and benefits from certification. The bank of each party may certify the balance, ensuring that one may only run the protocol with their true amount of money.

Private information retrieval (PIR) allows Alice to query a database held by Bob, without revealing her query. Alice may make queries for any item of the database and learn arbitrary blocks of it, independent of the set she holds. In a weak model where Alice is trusted to only query for items which belong to her set, PIR could implement private set intersection. As we have argued, many applications of set intersection require security in the presence of malicious adversaries.

Oblivious transfer (OT) is a protocol which allows Alice to transfer one of two items to Bob, such that Bob can choose which item he wants, keep his choice hidden from Alice, and learn nothing about the other item. OT can be used to construct private set intersection protocols (see [15]) however these are less efficient than specialized protocols, and efficiency decreases when elements are chosen from larger domains.

Finally, *password-based key exchange* allows two users who share a low-entropy password to establish a strong shared key. This protocol should fail if the passwords are different, and therefore bears similarity to the private intersection problem with sets of size one. However, the security requirements and definitions are quite different due to the differing goals of the two protocols.

Contributions and Outline. Our new protocol boasts the following features. The protocol (described in Section 6) may output the intersection or the intersection cardinality, and in either case both parties learn the output. Certified sets, which are the inputs to the protocol, are presented in Section 5. Participants can use different certifying authorities, provided both parties trust the authorities. We prove security in the strongest model in the literature, malicious participants with certified inputs. The model is described in Section 2, and the security proof appears in Appendix A. A strategy for adding fairness in the presence of participants which may abort the protocol prematurely is given in Section 6.4. With respect to computational complexity, the number of arithmetic operations is comparable to existing non-certified methods, however the dominant operations occur in a group with a significantly faster operation. The amount of communication is comparable to previous approaches as well, but again, each communicated element is smaller due to the choice of group. Efficiency is discussed in Section 7. Section 3 will provide background related to the protocol, and Section 4 reviews related protocols from the literature. An explicit description of the zero knowledge protocols used is given in Appendix B.

2 Model and Definition

We work in a stronger version of what is generally called the malicious model, and we focus on the two-party case. The malicious model is formally defined in the book of Goldreich [17, §7.2]. In our case we will have the participants, who hold sets they wish to intersect, and the certification authorities CAs. We assume the CAs will be honest.

In the malicious model, either participant may behave arbitrarily, while privacy is maintained for the honest participant. Limitations of this model are (i) security is only guaranteed when one participant is assumed honest, (ii) we cannot prevent parties from aborting the protocol, and (iii) inputs to the protocol may be arbitrary. We will lift limitation (iii) by allowing only certified sets as inputs.

We discuss ways to mitigate unfairness due to aborts in Section 7.4 by applying optimistic fair-exchange protocols [1]. Using these techniques, if one party aborts prematurely, we are guaranteed that both parties learn the same amount of information.

The Ideal Functionality. In the ideal functionality for the private intersection of certified sets a trusted party U will perform the intersection. Essentially, certification authorities will inform U of the a participant's certified set, then two parties signal U that they wish to compute the intersection of their sets. We now describe these steps in greater detail.

Certify: Upon receiving a message $(\mathsf{Certify}, S_{P_i}, P_i, \mathrm{CA}_j)$ from CA_j, U records that CA_j has certified the set S_{P_i} for use in the protocol by participant P_i.

IdealProtocol: The message $N = (\textsf{IdealProtocol}, P_i, P_j)$ from P_i, indicates that P_i would like to run the protocol with P_j. Upon receiving N from P_i, if U has received (IdealProtocol, P_j, P_i) the IdealProtocol begins, otherwise N is stored and U waits. If U has not received $(\textsf{Certify}, S_{P_i}, P_i, *)$, CA_{P_i} is set to `null` and S_{P_i} is set to $\emptyset$ before IdealProtocol begins. U behaves analogously if it has not received (Certify, S_{P_j}, P_j, $*$).

At each step, after receiving output from U each party must respond with either "`ok`" to continue the protocol, or "`abort`" to end the protocol at this point. This is required to model limitation (ii) above, and to allow participants to abort if they do not trust the CA of the other participant. Here we describe the IdealProtocol between participants A and B.

1. (a) U sends $\mathrm{CA}_A, |S_A|$ to B.
   (b) B responds `ok` or `abort`.
2. (a) U sends $\mathrm{CA}_B, |S_B|, |S_A \cap S_B|$ to A.
   (b) A responds `ok` or `abort`.
3. (a) U sends $S_A \cap S_B$ to B.
   (b) B responds `ok` or `abort`.
4. (a) U sends $S_A \cap S_B$ to A.
   (b) A responds `ok` or `abort`.

Simulation and trusted CAs. In order to simulate the protocol against malicious adversaries, the simulator must know which CAs the honest participant trusts. Without this information the malicious party may distinguish interaction with the simulator from interaction with the honest party since the simulator would not be able to consistently reject the same CAs as the honest participant. Since the list of CAs trusted by a participant is not considered private information, we assume that honest participants make the list public.

Remark 1. A and B should agree to use authorities they both trust before approaching U, since B may learn $|S_A|$ before A can decide that CA_B is untrustworthy (if B is malicious he may learn $|S_A|$ and abort). Since the "role" of the participant in the protocol is not specified, we simply assume that the first person to send the IdealProtocol message will play the role of A in the description of IdealProtocol. If a participant has multiple sets (and/or multiple certifying authorities) these are handled by a associating a different identity to each set, for example $A||$`set1`, $A||$`set2`, etc.

The real world model. In the real world there is no trusted party U, and participants are polynomial time algorithms, initialized with public keys of the CAs as required. A malicious participant may follow any polynomial time strategy.

Remark 2. An honest participant will abort if any deviation from the protocol is detected. Adversarial behaviour can thus serve to accomplish three outcomes.

1. Learning more about the other party's set than what is allowed in the ideal model.
2. Preventing the other party's output from being correct.
3. Using uncertified set elements in a protocol run.

With the definitions of the real and ideal models in place, we can now give a precise definition of a secure certified private intersection protocol.

Definition 1. *Let A and B^* be parties holding sets S_A and S_{B^*} from a domain D, certified by CA_A, CA_{B^*} respectively. Without loss of generality, B^* may behave arbitrarily (real-world adversary). Let Π be a private set intersection protocol, and $\mathcal{D}$ be the joint distribution of the outputs of A and B^* from Π when CA_A and CA_{B^*} are honest. Π is a* secure certified private intersection protocol *if a there exists a simulator (ideal-world adversary) which is given black-box access to B^* such that $\mathcal{D}$ is computationally indistinguishable from the joint output distribution of the simulator and A in the ideal world.*

Models for computing the intersection cardinality. A slightly modified ideal model applies to private computation of the intersection cardinality with certified sets. In the description above, Steps 3 and 4 are replaced by the single step "U sends $|S_A \cap S_B|$ to B". The real world model is unchanged.

3 Background

In this section we give the building blocks and notation we will use. The notation $x \in_R X$ denotes that x is chosen uniformly at random from the set X. We use $\{0,1\}^\ell$ to represent the set of all binary strings of length ℓ, as well as the set $[0, 2^\ell - 1]$ of integers. The notation $\pm \{0,1\}^\ell$ is used for the set $[-2^\ell + 1, 2^\ell - 1]$.

3.1 Zero Knowledge Proofs

When presenting protocols we express zero knowledge (ZK) proofs using notation introduced by Camenisch and Stadler [5], which allows the aim of proof protocols to be described without giving all details. In short, the notation

$$PK \{(x, y, \ldots) : \textit{statements} \text{ involving } x, y, \ldots\}$$

means the prover is proving knowledge of $(x, y, \ldots)$ such that these values satisfy *statements*. The types of statements are knowledge of, relations between, and the length of discrete logarithms.

The realization of the proofs of knowledge described above may be done in a variety of ways, each requiring different amounts of interaction and security assumptions. For the security of our protocol, we require that all ZK proofs be efficiently simulated. A protocol for concurrent ZK which may be simulated is given by Damgård [9]. The protocol uses the public key of a third party as the auxiliary string. In the protocols we present, since the CAs public key must be known by both A and B, it may be used as the auxiliary string. Replacing the verifier in a three-move ZK proof by a hash function gives a non-interactive ZK proof of knowledge [13]. Since it is non-interactive, there are no concurrency issues, and simulation is possible in the random oracle model.

3.2 Camenisch-Lysyanskaya Signatures

The Camenisch-Lysyanskaya (CL) signature scheme [7] signs L-tuples of strings from $\{0,1\}^{\ell_m}$. Given a signature on a tuple of elements, we may efficiently prove possession of a signature on some or all elements in the tuple. Further, this proof may be completed without revealing the signature itself.

We now describe a basic version of the scheme, where the signer learns all of the messages. In Section 5 we discuss possible applications of the signer's ability to sign tuples where some of the messages are hidden. A number of length related security parameters are used in the CL-signature scheme. For details on how they are chosen, see [7].

Key generation For a security parameter ℓ_n, choose an ℓ_n-bit RSA modulus $n = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, p' and q' are prime. Choose uniformly at random $R_1, \ldots, R_L, S, Z$ from the group of quadratic residues mod n. The public key is $(n, R_1, \ldots, R_L, S, Z)$ and the secret key is (p, q).

Signing algorithm. On input $m_1, \ldots, m_L$, the signer chooses at random a prime e of length $\ell_e > \ell_m + 2$, and a random number v of length $\ell_v = \ell_n + \ell_m + \ell_\emptyset$, where $\ell_\emptyset$ is a security parameter. Compute

$$A = \left(\frac{Z}{R_1^{m_1} \cdots R_L^{m_L} S^v}\right)^{1/e} \pmod{n} .$$

The signature is (A, e, v).

Verification algorithm. (A, e, v) is a valid signature on the message $(m_1, \ldots, m_L)$ if

$$Z \equiv A^e R_1^{m_1} \cdots R_L^{m_L} \pmod{n} ,$$

$m_i \in \pm\{0,1\}^{\ell_m}$, and $2^{\ell_e - 1} < e < 2^{\ell_e}$.

Proof of possession. This proof assumes the prover wishes to keep all messages hidden. Let ℓ_H be a security parameter. Choose $r \in_R \{0,1\}^{\ell_n + \ell_\emptyset}$, and randomize the signature (A, e, v) as $(A' = AS^{-r} \pmod{n}, e, v' = v + er)$. The randomized signature is communicated to the verifier and the prover asserts:

$$\begin{aligned} PK\{(e, v', m_1, \ldots, m_L) : \quad & \frac{Z}{A'^{2^{\ell_e - 1}} R_1^{m_1} \cdots R_L^{m_L}} \equiv \pm A'^e S^{v'} \pmod{n} \\ \wedge \quad & m_i \in \{0,1\}^{\ell_m + \ell_\emptyset + \ell_H + 2} \text{ for } i = 1 \ldots L \\ \wedge \quad & e - 2^{\ell_e - 1} \in \pm\{0,1\}^{\ell'_e + \ell_\emptyset + \ell_H + 1}\} . \end{aligned}$$

The first predicate convinces the verifier that the signature is in fact valid, while the second and third prove that it is well formed with respect to the system parameters. For details of how the interval checks on e and the m_i are realized, see [7] (this proof is also given in more detail as the part of the protocol in Appendix B). Security of the CL-signature scheme relies on the strong RSA (SRSA) assumption, see [7] for details of this assumption and a security proof.

The proof of possession as stated is of limited utility, it merely asserts that the holder has a signature on some tuple of correctly formed messages. However, we will compose this proof with one to show that certain operations were completed using the signed values. This will allow A and B to prove that only signed values are used in the intersection protocol.

3.3 Homomorphic Encryption

We also review two homomorphic encryption schemes used for private set intersection. Both are *additively homomorphic*, i.e. for two encryptions $E(m_1), E(m_2)$ of messages m_1, m_2, $E(m_1) \star E(m_2) = E(m_1 + m_2)$, where $\star$ is a group operation on ciphertexts. It follows by repeated addition that $E(m_1)^c = E(cm_1)$ for an integer c.

The Paillier Cryptosystem. The Paillier cryptosystem [25] encrypts plaintexts from $\mathbb{Z}_n^*$ as ciphertexts in $\mathbb{Z}_{n^2}^*$. Security relies on the decisional composite residuosity assumption, which requires (as a minimum) that n be difficult to factor. For encryption, decryption and to operate on encrypted values requires arithmetic mod n^2.

The cryptosystem is probabilistic, IND-CPA secure, and allows efficient proofs of plaintext knowledge, as well as multiplicative relationships on plaintexts [10]. We will largely treat Paillier encryption as a black box, or generic homomorphic encryption scheme with these properties, and do not describe the details of the system here.

A homomorphic Elgamal variant. Our new protocols will use a standard variant of Elgamal encryption [11]. Setup consists of choosing a cyclic group G of prime order q, such that the discrete log problem is difficult in G. The parameter ℓ_q is the bitlength of q. Next choose a generator $g \in G$ and the secret key $x \in_R \mathbb{Z}_q^*$. The public key is $g, h = g^x$. To encrypt m, choose $r \in_R \mathbb{Z}_q^*$, and compute $E(m) = (g^r, g^m h^r)$. The additive homomorphic property is easily verified. Efficient decryption is not possible; but to recognize an encryption of zero, given x, compute $(g^r)^{-x}(g^m h^r) = g^{-rx} g^{m+rx} = g^m$, which is one precisely when $m = 0$. This test will be sufficient for our protocols, decryption will not be necessary.

As with Paillier, the scheme is probabilistic and IND-CPA secure. ZK proofs of plaintext knowledge are simply proofs of knowledge of discrete logs. It is worth noting that arithmetic in G will be significantly faster than in $\mathbb{Z}_{n^2}^*$ for comparable levels of security and the ciphertexts will be smaller.

3.4 Verifiable Shuffles

A sub-protocol we use in our intersection cardinality protocol is a verifiable shuffle decryption. A *verifiable shuffle* of ciphertexts takes a list of ciphertexts $e_1, \ldots, e_k$ as input, and outputs a second list of ciphertexts $E_1, \ldots, E_k$, which contain the *same* plaintexts in a permuted order. The public key of the e_i, E_i is the same. In a *verifiable decryption*, the decryptor proves that the decrypted values correspond to the ciphertexts without revealing the private key. A *verifiable shuffle decryption* is the combination, first the ciphertexts are shuffled, then decrypted and proof is given that the plaintexts correspond to input ciphertexts. The result of the operation is that the verifier does not learn which input ciphertexts correspond to which plaintexts, the permutation is kept secret.

One can simply combine a shuffle protocol (such the one of Groth and Ishai [18]), with a proof of correct decryption, or one may use a combined protocol (such as the one of Furukawa [16]). The combined method of Furukawa, which is specialized to Elgamal ciphertexts, requires $14k$ exponentiations and communication of approximately k group elements.

4 Existing Private Intersection and Cardinality Protocols

In this section we describe previously known protocols to solve the private set intersection problem (and variants).

The work of Freedman, Nissim and Pinkas (FNP) was the first to present the private set intersection problem, and protocols to solve it [15]. We will describe their design strategy in some detail, since it underlies most of the subsequent work on this topic (including our own). *Throughout this paper, we assume that* $|S_A| = |S_B| = k$ to simplify presentation, however all of the protocols presented also work when $|S_A| \neq |S_B|$.

Suppose pk_A is the public key of A for the Paillier cryptosystem (or a scheme providing similar features). Let R be a ring, $R[x]$ be the polynomials with coefficients from R, and D be the domain to which S_A and S_B belong. We will require that $|D|/|R|$ is negligible. First, A represents $S_A = (a_1, \ldots, a_k)$ as the roots of a degree k polynomial, $f = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} \alpha_i x^i$, then encrypts the coefficients with pk_A. These are then sent to B, who evaluates f at each $b_i \in S_B$ homomorphically. The key observation is that $f(b_i) = 0$ if and only if $b_i \in S_A \cap S_B$. B returns $w_i = E(s_i f(b_i) + b_i))$ to A, for a randomly chosen value s_i. If $b_i \in S_A \cap S_B$ then A leans b_i upon decrypting. If $b_i \notin S_A \cap S_B$ then w_i decrypts to a random value.

This simplified version of the protocol is secure in the semi-honest model. To cope with malicious parties, FNP give protocols to deal with the cases when A may be malicious, or when B may be malicious. They also sketch a strategy for combining the two to handle either A or B behaving maliciously. The protocol uses a cut-and-choose technique, which quickly becomes inefficient in both computation and communication as k grows.

Kissner and Song (KS) [22, 23] present improved protocols for more general set operations, as well as protocols for set operations in the multiparty case. We review the crux of their approach. Let s and t be randomly chosen polynomials in $R[x]$ and f, g be polynomials representing sets S_A, S_B respectively, and $\deg s, t, f, g = k$. The authors prove that $sf + tg = \gcd(f, g) \cdot u$, where u is a uniformly random element of $R[x]$. Combined with the condition that the domain D of S_A and S_B is very small compared to R, the chance that u contains an element from D as a root is low. Therefore the only elements of D which are roots of $sf + tg$ are those in $\gcd(f, g) = S_A \cap S_B$. The parties jointly compute encryptions of $sf + tg$, then decrypt to learn the intersection. The advantage of this representation of $S_A \cap S_B$ is that it composes well with other operations, and handles more than two parties easily.

Their solution for two-party private set intersection, secure in the malicious model, has computation and communication complexity $O(k^2)$. They do not present a protocol for the two party private intersection cardinality problem secure in the malicious model. We also note that their malicious model protocols require the use of Paillier encryption (or a homomorphic scheme with equivalent properties).

Hohenberger and Weiss [20] provide protocols for a private disjointness test where A is semi-honest and B may be malicious, and a private intersection cardinality protocol in the semi-honest model. Their protocols are also based on the paradigm of FNP, but use the homomorphic Elgamal variant presented in §3, and rely on the ability to recognize encryptions of zero.

Hazay and Lindell [19] give protocols for two party private set intersection using a novel approach based on oblivious pseudorandom function evaluation (instead of oblivious polynomial evaluation). The protocol is more efficient than previous solutions, however security is proven in a relaxed version of the malicious model. A further difference of this protocol with the one presented here is that the output is only learnt by one participant.

Finally, Kiayias and Mitrofanova [21], and Ye et al. [27] provide protocols for a restricted version of private set intersection, the case when a single bit is output, indicating whether the intersection is non-empty. We omit details of these papers, since solutions to this problem are much less efficient than intersection and cardinality, and differ significantly from the present work.

5 Certified Sets

Here we describe the process a CA uses to certify a set for a participant. Once certified, the set may be used in the private set intersection protocol of Section 6. A discussion of the possibility of using certified sets with existing private set intersection protocols is available in [28].

Certification will be done by the CA, who issues a CL-signature to the set holder A for the set $S_A = (a_1, \ldots, a_k)$. Given this signature (or certificate) A must be able to prove the following.

1. That encrypted coefficients correspond to the polynomial representation of a certified set.
2. That the set used in a computation is certified.
3. The size of the set.

Let S_A be represented by the polynomial $f(t) = \sum_{i=0}^{k} \alpha_i t^i$. The message space of the CL signature scheme used by the CA must have length $k+1$. We now present the method to certify the set S_A. Our presentation assumes the homomorphic Elgamal scheme is used.

First, the CA signs the coefficients and the degree of the polynomial. Signing coefficients allows requirement 1 to be easily proven. During certification, the user sends $S_A, \alpha_0, \ldots, \alpha_k$ to the CA. The CA checks whether α_i are the coefficients of $f(t) = \prod_{a \in S_A}(t - a)$ and that S_A is valid for the user. Then the CA issues two signatures, one on $(k, \alpha_1, \ldots, \alpha_k)$ and one on $(k, a_1, \ldots, a_k)$.

Proof that ciphertexts $E_i = (g^{r_i}, g^{\alpha_i} h^{r_i})$ contain encryptions of certified coefficients is:

$$\begin{aligned}\mathrm{PK}\{(\alpha_0,\ldots,\alpha_{k-1}, r_0,\ldots,r_{k-1}) : &\ \alpha_i \text{ are CL-signed} \\ &\wedge E_i = (g^{r_i}, g^{\alpha_i} h^{r_i}) \text{ for } i = 1,\ldots,k-1\}\ ,\end{aligned}$$

where proof that "α_i are CL-signed" is done as described in §3.2. Proving that elements used in a computation are certified is easy; one simply proves that they are CL-signed. The size of the set $|S_A| = k$ is the first attribute in the signature and should be revealed and checked during proof.

In the case when the CA's public key has $L > k+1$ bases, the elements corresponding to the additional bases are set to zero and ignored during the protocol.

Extensions. The authentication of set holders may also be included in the certification process, by including an identifier or pseudonym as the first value signed by the CA. During the first proof involving the signature, the holder may reveal or prove something about their identity. Preventing users from sharing their sets and signatures is not possible, but this problem has been studied in the context of anonymous credentials, see [6, 8] for some deterrents. Note that shared sets may not be combined to participate in the protocol with a larger set, the CL signature scheme prevents this.

Another possible extension is to allow users to keep some set elements hidden from the CA (since this feature is provided by the CL-signature scheme). This way some elements may remain completely private, while still preventing the user from changing their set, and limiting the size of the set.

6 New Certified Private Intersection Protocol

We now describe our new protocol for privately computing the intersection and intersection cardinality of certified sets. We begin with an overview before giving complete details.

6.1 Overview

Suppose A has $S_A = (a_1,\ldots,a_k)$ and B has $S_B = (b_1,\ldots,b_k)$. We first sketch a private intersection cardinality protocol where both S_A and S_B are certified. This protocol will be extended below to compute the actual intersection as well. Suppose $f(t) = \sum_{i=0}^{\ell} \alpha_i t^i = \prod_{j=1}^{\ell}(t - a_j)$ represents S_A. G will be the group used for Elgamal homomorphic encryption.

- The CA certifies S_A and S_B using the method from Section 5.
- A encrypts α_i using Elgamal homomorphic encryption (denoted $E(\cdot)$) under his public key, and proves that this was done correctly. In this same proof A proves holdership of a CL-signature on α_i for $i = 1,\ldots,k$, and the cardinality of S_A.

- B first verifies the proof that the encryptions of α_i were formed correctly and checks the cardinality of S_A. B computes $w_i = E(s_i f(b_i))$ where $s_i \in_R \mathbb{Z}_q^*$, for each $b_i \in S_B$ using the homomorphic properties of E. A proof is included that w_i are computed correctly, that b_i are signed and that the cardinality of S_B is correct.
- A decrypts w_i to get $g^{s_i f(b_i)}$, and counts how often $g^{s_i f(b_i)} = 1$; this total is the intersection cardinality.
- A outputs the cardinality to B, and proves it is correct using a verifiable shuffle decryption, as described in §3.4.

Extension to compute $S_A \cap S_B$. The following steps can be added to the protocol to provide the intersection, not just its cardinality.

- When B computes w_i for all $b_i \in S_B$, he stores a lookup table mapping $w_i \leftrightarrow (s_i, b_i)$.
- A decrypts; whenever $w_i = 1$, he proves this to B, who looks up the value (s_i, b_i). In this way B learns $S_A \cap S_B$. A must also prove $f(b_i) \neq 0$ (when this is the case) to convince B that the entire intersection is output.
- B reports $S_A \cap S_B$ to A as pairs (s_i, b_i), who checks it for consistency by checking $w_i \stackrel{?}{=} E(s_i f(b_i))$ and by checking $|S_A \cap S_B| = |\{i : D(w_i) = 1\}|$.

6.2 Detailed Description

We now describe the complete protocol, with non-interactive ZK proofs, the details of which are given in Appendix B. Recall that Elgamal ciphertexts have the form $E_i = (g^r, g^m h^r)$. In this section we will refer to the first element of the ciphertext as $E_{i,1}$ and the second as $E_{i,2}$.

Setup:

A has the set S_A, represented by $f(t) = \sum_{i=0}^{k} \alpha_i t^i$, certified as in §5.
B has the set S_B, also certified with the method of §5.
A generates the homomorphic Elgamal parameters G and $pk_A = (g, h)$ which are made public, and $x = \log_g h$ kept secret.

Protocol:

1. A computes $E_i = E(\alpha_i) = (g^{r_i}, g^{\alpha_i} h^{r_i})$ using pk_A, $r_i \in_R \mathbb{Z}_q$ for $i = 1, \ldots, k$.
2. A creates

$$\begin{aligned} P_1 = \mathrm{PK}\{(\alpha_0, \ldots, \alpha_k, r_0, \ldots, r_k) :& \\ & E_{i,1} = g^{r_i} \ \wedge \ E_{i,2} = g^{\alpha_i} h^{r_i} \text{ for } i = 1, \ldots, k \\ & \wedge\, k \text{ and } \alpha_i \text{ are CL-signed}\} \end{aligned}$$

3. A sends $(E_0, \ldots, E_k, P_1)$ to B.
4. B verifies P_1, and aborts if verification fails.

5. B homomorphically evaluates f at elements in S_B by computing

$$v_i = \left(\prod_{j=0}^{k} E_{j,1}{}^{(b_i)^j}, \prod_{j=0}^{k} E_{j,2}{}^{(b_i)^j} \right)$$

for each $b \in S_B$ in random order, then computes $w_i = (v_{i,1}{}^{s_i}, v_{i,2}{}^{s_i})$ for $s_i \in_R G$. Note that $w_i = E(s_i f(b_i))$. B stores a table mapping $w_i \leftrightarrow (b_i, s_i)$ (this may be omitted if only the intersection cardinality is desired).
6. B creates the proof

$$\begin{aligned} P_2 = \mathrm{PK}\{&(b_0, \ldots, b_k, s_0, \ldots, s_k) : \\ &w_i = \left(\prod_{j=0}^{k} E_{j,1}{}^{(b_i)^j s_i}, \prod_{j=0}^{k} E_{j,2}{}^{(b_i)^j s_i} \right), \quad \text{for } i = 1, \ldots, k \\ &\wedge k, b_i \text{ are CL-signed}\} \ . \end{aligned}$$

7. B sends $(w_1, \ldots, w_k, P_2)$ to A.
8. A verifies P_2, and aborts if this fails. A must also check that $s_i \neq 0$ by ensuring that $w_{i,1} \neq 1$ for $i = 1, \ldots, k$.
9. If the intersection cardinality is desired: (if not, skip to Step 10)
   (a) A initializes a counter $c = 0$, decrypts w_i to get $g^{s_i f(b)}$ and increments c if $g^{s_i f(b)} = 1$.
   (b) A outputs c, the size of the intersection. Using a verifiable shuffle decryption protocol, A proves that c of the ciphertexts w_i decrypt to 1, without revealing which ones. (See Section 3.4.)
   (c) B verifies the shuffle decryption proof.
   (d) The protocol terminates.
10. A decrypts w_i for $i = 1, \ldots, k$, and creates the following partition of $\{w_1, \ldots, w_k\}$:

$$\begin{aligned} \mathcal{C}_1 &= \{w_i : D(w_i) = 1\} \ , \\ \mathcal{C}_y &= \{w_i : D(w_i) = y \neq 1\} \ . \end{aligned}$$

11. A proves that the decryptions of w_i are (or are not) equal to zero with the following proof:

$$\begin{aligned} P_3 = \mathrm{PK}\{(x) : (w_{i,1}{}^{-x})(w_{i,2}) = 1 \ \forall \ &\{i : w_i \in \mathcal{C}_1\} \\ \wedge \quad (w_{i,1}{}^{-x})(w_{i,2}) = y_i \neq 1 \ \forall \ &\{i : w_i \in \mathcal{C}_y\} \\ \wedge \quad g^x = h\} \ . \end{aligned}$$

12. A sends C_1, C_y, P_3 to B.
13. B verifies P_3. (Note also that B must check that P_3 contains the correct number of statements, i.e. that all w_i appear in one of C_1, C_y).
14. For each i such that $w_i \in \mathcal{C}_1$, B recovers (s_i, b_i) from the lookup table, and adds it to a set X.
15. B sends X to A, which contains $S_A \cap S_B$, and A checks that
   (a) $|X| = |\mathcal{C}_1|$, and

(b) $w_i = E(s_i f(b_i))$ (recomputed using the revealed values (s_i, b_i)).

If either check fails, A learns that B has output $S_A \cap S_B$ incorrectly.

Remark 3. In both steps 11 and 9b when A proves to B that w_i is not contain an encryption of zero, it is important that the decrypted value, $g^{s_i f(b_i)}$, is not revealed since B knows s_i. A must therefore blind the ciphertexts in C_y (which are not encryptions of zero) as $w_i^{u_i}$ where $u_i \in_R \mathbb{Z}_q^*$. Since $s_i, u_i \neq 0$, $g^{s_i u_i f(b_i)} = 1$ if and only if $f(b_i) = 0$, as required.

6.3 Security and Privacy

The following theorem (proven in Appendix A) shows that the new protocol securely implements the ideal functionality described in Section 2.

Theorem 1. *The protocol of Section 6.2, when constructed with a secure ZK protocol, and an IND-CPA secure homomorphic encryption scheme, is a secure certified private intersection protocol (by Definition 1) assuming the the SRSA assumption holds.*

When the homomorphic Elgamal variant is used as the encryptions scheme, security of the protocol relies on the following assumptions: the SRSA assumption in $\mathbb{Z}_n^*$, the discrete logarithm problem in G, and any additional assumptions required for the security of the zero knowledge proofs.

6.4 Adding Fairness

Until this point we have not addressed the question "What if B aborts the protocol after learning $S_A \cap S_B$, but before A does?". The possibility for such an unfair outcome is undesirable in a situation where either A or B may be malicious. In this section we sketch the incorporation of optimistic fair exchange (OFE) protocols to the private set intersection protocol of §6.2. An OFE scheme allows A and B to swap two values "simultaneously", i.e. both are guaranteed to receive the value held by the other. A trusted third party is present, but only participates when one party does not complete the protocol (hence the term *optimistic*). Example OFE schemes are given in the work of Asokan et al. [1].

To add OFE to our set intersection protocol, we weave two instances of the protocol together, where A and B have opposite roles in each instance. The new protocol is now symmetric, i.e. A and B must perform equivalent operations and communicate equivalent values at the same steps, which are exchanged fairly. Any abort thus results in a fair outcome, where both parties finish with equal knowledge about the other's input set. Using an OFE protocol in such a generic way may yield a protocol with room for improvement; we leave such improvements to future work.

7 Efficiency

A detailed analysis of the computational and communication costs of our new protocol is given in the extended version of this paper [28]. We count the number of exponentiations required in $\mathbb{Z}_n^*$ (the CL-signature group) and G (the Elgamal group). The dominating computations, evaluating the polynomial, all occur in G. A detailed comparison of our new protocol to a certified version of the FNP protocol and/or the malicious, two-party, non-certified protocol of Kissner and Song [22, Figure 8] would be beyond the scope of this work. Communication and computation costs are asymptotically equal, each being $O(k^2)$.

The constants however, will be significantly smaller since Elgamal parameters are smaller than Paillier parameters providing equivalent security. For 80-bits of security, Paillier with a 1024-bit modulus, yields ciphertexts of 2048 bits. In the Elgamal case we may use the elliptic curve group given by NIST curve P-192 (a curve over $\mathbb{F}_p$, where p is a 192-bit prime) [14]. The Elgamal ciphertexts are thus 384 bits, 5.3 times smaller than the equivalent Paillier ciphertext.

In addition to providing faster arithmetic, operating primarily in G allows the protocol to scale to higher security levels. The size of parameters required for Paillier grow quadratically as a function of the security level, while parameters for elliptic curve systems grow linearly (see Lenstra and Verheul [24]).

Since the ZK proofs are relatively simple, they are good candidates for the batch verification techniques of Bellare, Garay and Rabin [2]. Fast exponentiation and multi-exponentiation techniques (see [3] for a survey) are also applicable, and will improve performance significantly. Implementations may use Horner's rule for polynomial evaluation (as suggested in [15]). When multiple processing units are available, a parallel polynomial evaluation scheme due to Estrin [12] will further improve performance.

References

1. N. Asokan, V. Shoup and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications* **18** (2000), 593–610.
2. M. Bellare, J. Garay and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. *EUROCRYPT 1998*, LNCS **1403** (1998), 236–250.
3. D. J. Bernstein. Pippenger's exponentiation algorithm. Manuscript. `http://cr.yp.to/papers.html#pippenger`.
4. F. Boudot, B. Schoenmakers and J. Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete and Applied Math.* **111** (2001), 23–36.
5. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report **TR 260** (1997), Institute for Theoretical Computer Science, ETH Zürich.
6. J. Camenisch, A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Proceedings of EUROCRYPT 2001*, LNCS **2045** (2001), 93–118.
7. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. *Proceedings of SCN 2002*, LNCS **2576** (2002), 268–289.
8. J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, M. Meyerovich. How to win the clone wars: efficient periodic n-times anonymous authentication. *Proceedings of CCS 2006*, ACM Press, 201–210.

9. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. *Proceedings of EUROCRYPT 2000*, LNCS **1807** (2000), 418–430.
10. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. *PKC 2001*, LNCS **1992**, 119–136.
11. T. Elgamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **31** (1986), 469–472.
12. G. Estrin. Organization of computer systems: The fixed-plus variable structure computer. *Proceedings of the Western Joint Computer Conference* (1960), 33–40.
13. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. *Proceedings of CRYPTO 1986*, LNCS **263** (1987), 186–194.
14. National Institute of Standards and Technology. Digital signature standard (DSS). *FIPS PUB* **186-2** (2000).
15. M. J. Freedman, K. Nissim and B. Pinkas. Efficient private matching and set intersection. *Proceedings of EUROCRYPT 2004*, LNCS **3027** (2004), 1–19.
16. J. Furukawa. Efficient verifiable shuffle decryption and its requirement of unlinkability. *Proceedings of PKC 2004*, LNCS **2947** (2004), 319–332.
17. O. Goldreich. *The Foundations of Cryptography – Volume 2 Basic Applications*. Cambridge University Press, New York, 2004.
18. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. *Proceedings of EUROCRYPT 2008*, LNCS **4965** (2008), 379–396.
19. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. *Proceedings of Theory of Cryptography 2008*, LNCS **4948** (2008), 155–175.
20. S. Hohenberger and S. Weiss. Honest-verifier private disjointness testing without random oracles. *Proceedings of PET 2006*, LNCS **4258** (2006), 277–294.
21. A. Kiayias and A. Mitrofanova. Testing disjointness of private datasets. *Proceedings of Financial Cryptography 2005*, LNCS **3570** (2005), 109–124.
22. L. Kissner and D. Song. Private and threshold set intersection. Technical report **CMU-CS-04-182** (2004), School of Computer Science, Carnegie Mellon University.
23. L. Kissner and D. Song. Privacy preserving set operations. *Proceedings of CRYPTO 2005*, LNCS **3621** (2006), 241–247.
24. A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. Journal of Cryptology **14** (2001), 255–293.
25. P. Paillier. Public-key cryptosystems based on composite residuosity classes. *Proceedings of EUROCRYPT 1999*, LNCS **1592** (1999), 223–239.
26. R. Pass. On deniability in the common reference and random oracle model. *Proceedings of CRYPTO 2003*, LNCS **2729** (2003), 316–337.
27. Q. Ye, H. Wang, J. Pieprzyk, and X.-M. Zhang. Efficient disjointness test for private datasets. To appear in *Proceedings of the 13th Australasian Conference on Information Security and Privacy, ACISP 2009.*
28. Extended technical report version of this paper.

A Proof of Theorem 1

We consider three cases. First, when both A and B are honest we show the protocol output is the same as in the ideal-world (the correct output). Then, in the cases when A or B is malicious, we describe a simulator which satisfies Definition 1.

Suppose Y is the intersection output by the protocol when A and B are honest (for input sets S_A and S_B). If the following two claims hold, then the protocol is correct.

Claim 1: For every $y \in Y$, $y \in S_A$ and $y \in S_B$. Since $y \in Y$, it must therefore be that the decryption of some w_i, which is $g^{s_i f(y)}$ is equal to one (in the notation of Section 6). It must be that $f(y) = 0$, since $s_i \neq 0$ and G has prime order. We are assured that $f(t) = \prod_{a_i \in S_A}(t - a_i)$ by the validity of the CL-signature, therefore the only points at which f is zero are elements of S_A, therefore $y \in S_A$. Since S proves that w_i are computed using nonzero s_i and $b_i \in S_B$, we also have that $y \in S_B$.

Claim 2: For every y such that $y \in S_A$ and $y \in S_B$, we also have $y \in Y$. Similarly, since we are assured by the CL-signature that f is created with elements of S_A and $E(s_i f(b))$ is computed for all $b \in S_B$, it is not possible that some $y \in S_A \cap S_B$ has $f(y) \neq 0$ and as a result, will always be included in Y.

We prove privacy for A, i.e., we describe a simulator SIM_{B^*}, which is given black-box access to B^* in the ideal model such that the output distributions of A and B^* in the real world are indistinguishable from the ones of A and SIM_{B^*} in the ideal world. (Here, B^* may or may not follow the protocol.) The simulator's output is computationally indistinguishable from the view of B^* in a real protocol execution. Intuitively, SIM_{B^*} sits between U (the trusted party) and B^*, and interacts with both in such a way that B^* is unable to distinguish protocol runs with SIM_{B^*} from real-world protocol runs with A.

SIM_{B^*} is the following polynomial time algorithm. First SIM_{B^*} sends (IdealProtocol, B^*, A) to U. IdealProtocol begins and SIM_{B^*} receives $\mathrm{CA}_A, |S_A|$. SIM_B then creates $E_i = E(\alpha_i) = (g^{r_i}, g^{\alpha_i} h^{r_i})$ for randomly chosen (r_i, α_i), and forges the proof P_1. SIM_{B^*} sends P_1 and E_i, to B^* (protocol Step 3), and responds `ok` to U. Now SIM_{B^*} receives $(w_1, \ldots, w_k, P_2)$ from B^* (protocol Step 7), or if B^* aborts, SIM_{B^*} returns `abort` to U and stops. Recall that SIM_{B^*} knows the CAs which A trusts, and may therefore reject CA_B (used in P_2) if A does not trust CA_B. If P_2 is invalid, SIM_{B^*} also returns `abort` to U and stops. From P_2, SIM_{B^*} extracts b_j, $j = 1, \ldots, k$ and the mapping $w_i \leftrightarrow b_j$, i.e., knowledge of which w_i corresponds to an encryption of $f(b_j)$. SIM_{B^*} now receives $S_A \cap S_{B^*}$ from U, which gives SIM_{B^*} enough information to create the sets C_y, C_1 consistent with $w_i \leftrightarrow b_j$ and $S_A \cap S_B$. C_1 and C_y along with forged proofs of decryption P_3 are sent to B^* (protocol Step 12), and SIM_{B^*} responds `ok` to U. A receives the intersection and the protocol is complete. Finally SIM_{B^*} outputs whatever B^* outputs.

Let us argue that the output distributions in the real and the ideal world are (computationally) indistinguishable. First note that due to the security of the CL signature scheme, $B*$ in the real world cannot obtain a certificate on a set different from what it can obtain in the ideal world, hence the output of A will be identical in both worlds. We next explain why the view of B^*, as output by SIM_{B^*} is computationally indistinguishable from B^*'s view in real protocol runs with A. The encryptions E_i of random values are indistinguishable from the honest encryptions because the encryption scheme is IND-CPA secure. The forged proofs are also indistinguishable, by the zero-knowledge property. In the last step, the sets C_1, C_y are created exactly as A would in the real world, since at this point SIM_{B^*} has $w_i \leftrightarrow b_j$ and $S_A \cap S_B$. This means that B^* cannot distinguish whether or not it runs with the real world A or with SIM_{B^*}, i.e., any difference in the B^* views would imply one of our assumptions is false.

We now prove privacy for B in a similar manner, by describing an efficient simulator SIM_{A^*}. SIM_{A^*} sends ($\mathsf{IdealProtocol}$, A^*, B) to U, and waits to receive CA_B, $|S_B|$, $|S_A \cap S_B|$ from U (recall that B is honest and will not abort). SIM_{A^*} receives $E_0, \ldots, E_k$ from from A^* and proof P_1 (protocol Step 3). If P_1 is invalid or if CA_A is untrusted by B or if A^* has aborted, SIM_{A^*} stops and returns `abort` to U. Otherwise SIM_{A^*} extracts $\alpha_0, \ldots, \alpha_k$ from P_1, recovers S_A and responds `ok` to U.

Now SIM_{A^*} must perform Step 5. First choose a set Z of $|S_A \cap S_B|$ indices randomly from $\{1, \ldots, k\}$. For every $i \in Z$, compute $w_i = (g^{s_i}, h^{s_i})$ where $s_i \in_R \mathbb{Z}_q^*$. For the remaining indices $j \in \{1, \ldots, k\} - Z$, compute w_j as the homomorphic encryption of random values from $\mathbb{Z}_q^*$. SIM_{A^*} forges the proof that this was done correctly, then receives $S_A \cap S_B$ from U. If A^* does not abort, SIM_{A^*} receives, then verifies P_3. If P_3 is invalid SIM_{A^*} returns `abort` to U and stops. Otherwise SIM_{A^*} responds with (s_i, b_i) for $b_i \in S_A \cap S_B$ and s_i as chosen above (protocol Step 15).

The indistinguishability in this case comes from the IND-CPA security of the encryption; A^* cannot distinguish w_i for $i \notin Z$ from encryptions created during a real run of the protocol. The check in Step 15 passes since decryption of $w_i = g^{-s_i x} h^{s_i} = 1$, only when $i \in Z$, and therefore only $|S_A \cap S_B|$ times. Note that the mapping $b_i \leftrightarrow s_i$ is unimportant, since $f(b_i) = 0$ and w_i is an encryption of zero when $i \in Z$. Since the forged proof is also indistinguishable, the views of A^* during a real protocol run and the simulation are indistinguishable. Furthermore, due to the security of the CL signatures, the outputs of B in both worlds will be the same. $\square$

B Detailed Description of ZK Proofs

In this section we explicitly state the operations required to realize the ZK proofs and verifications of our new protocol (§6.2), using non-interactive ZK based on the Fiat-Shamir heuristic. Let $H : \{0,1\}^* \to \{0,1\}^{\ell_H}$ be a cryptographic hash function.

Creating P_1 (Step 2)

1. (*Randomize Signature*) Choose $r' \in \{0,1\}^{\ell_n + \ell_\emptyset}$, compute $\tilde{A} = AS^{r'} \pmod{n}$, compute $\tilde{v} = v + er'$, and compute $e' = e - 2^{\ell_e - 1}$.
2. Compute
$$U = \tilde{A}^{r_e} S^{r_{\tilde{v}}} \left(\prod_{i=1}^{k} R_i^{r_{\alpha_i}} \right) \pmod{n}$$
where $r_e \in_R \{0,1\}^{\ell'_e + \ell_\emptyset + \ell_H}$, $r_{\tilde{v}} \in_R \{0,1\}^{\ell_v + \ell_\emptyset + \ell_H}$ and $r_{\alpha_i} \in_R \{0,1\}^{\ell_m + \ell_\emptyset + \ell_H}$
3. For $i = 1, \ldots, k$, compute $T_i = g^{r_{r_i}},\ g^{r_{\alpha_i}} h^{r_{r_i}}$, where $r_{r_i} \in_R \{0,1\}^{\ell_q + \ell_\emptyset + \ell_H}$.
4. (*Challenge.*) Compute $c = H(\tilde{A} || U || T_1 || \ldots || T_k)$.

5. Compute, in $\mathbb{Z}$: $s_{e'} = r_e - ce'$, $s_{\tilde{v}} = r_{\tilde{v}} - c\tilde{v}$, $s_{\alpha_i} = r_{\alpha_i} - c\alpha_i$ for $i = 1, \ldots, k$, and $s_{r_i} = r_{r_i} - cr_i$ for $i = 1, \ldots, k$.
6. Output $P_1 = (c, s\text{-values from Step 5})$.
7. Send $(P_1, \tilde{A})$ to the verifier.

Verifying P_1 (Step 4)

1. Compute

$$\hat{U} = \left(\frac{Z}{\tilde{A}^{2^{\ell_e - 1}}}\right)^c \left(\tilde{A}^{s_{e'}} S^{s_{\tilde{v}}}\right) \left(\prod_{i=1}^{k} R_i^{s_{\alpha_i}}\right)$$

2. Let $E_i = (x_i, y_i)$. For $i = 1, \ldots, k$, compute $\hat{T}_i = x_i^c g^{s_{r_i}},\ y_i^c g^{s_{\alpha_i}} h^{s_{r_i}}$.
3. Compute $\hat{c} = H(\tilde{A}||\hat{U}||\hat{T}_1|| \ldots ||\hat{T}_k)$. If $\hat{c} \neq c$, reject the proof.
4. Check that $s_{e'} \in \pm \{0,1\}^{\ell'_e + \ell_\emptyset + \ell_H + 2}$ and $s_{\alpha_i} \in \pm \{0,1\}^{\ell_m + \ell_\emptyset + \ell_H + 3}$.

Creating P_2 (Step 6)

1. (*Randomize Signature*) Choose $r' \in_R \{0,1\}^{\ell_n + \ell_\emptyset}$, compute $\tilde{A} = AS^{r'} \pmod{n}$, compute $\tilde{v} = v + er'$, and compute $e' = e - 2^{\ell_e - 1}$.
2. Compute

$$U = \tilde{A}^{r_e} S^{r_{\tilde{v}}} \left(\prod_{i=1}^{k} R_1^{r_{b_i}}\right) \pmod{n}$$

where $r_e \in_R \{0,1\}^{\ell'_e + \ell_\emptyset + \ell_H}$, $r_{\tilde{v}} \in_R \{0,1\}^{\ell_v + \ell_\emptyset + \ell_H}$ and $r_{b_i} \in_R \{0,1\}^{\ell_m + \ell_\emptyset + \ell_H}$
3. We now prove the w_i are formed correctly, for each $b_i \in S_B$. Compute

$$T_{b_i} = \left(\prod_{j=0}^{k} (w_{i,1})^{r_{b^j}}\right)^{r_{s_i}}, \left(\prod_{j=0}^{k} (w_{i,2})^{r_{b^j}}\right)^{r_{s_i}}$$

where $r_{b^j} \in_R \{0,1\}^{\ell_m + \ell_\emptyset + \ell_H}$, and $r_{s_i} \in_R \{0,1\}^{\ell_q + \ell_\emptyset + \ell_H}$.
4. Compute $c = H(\tilde{A}||U||T_{b_1}|| \ldots ||T_{b_k})$.
5. Compute (in $\mathbb{Z}$): $s_e = r_e - ce'$, $s_{r_{\tilde{v}}} = r_{r_{\tilde{v}}} - cr_{\tilde{v}}$,

$$\begin{aligned} s_{b^j} &= r_{b^j} - cb^j && \text{for each } b \in S_B \text{ and } i = 1, \ldots, k \text{ and} \\ s_{s_i} &= r_{s_i} - cs_i && \text{for } i = 1, \ldots, k\ . \end{aligned}$$

6. Output $P_2 = (c, \text{values from Step 5})$.
7. Send $\tilde{A}$ to the verifier.

Verifying P_2 (Step 8)

1. Compute

$$\hat{U} = \left(\frac{Z}{\tilde{A}^{2^{\ell_e - 1}}}\right)^c \left(\tilde{A}^{s_e} S^{s_{r_{\tilde{v}}}}\right) \left(\prod_{i=1}^{k} R_i^{s_{b^i}}\right)\ .$$

2. Let $w_i = (x_i, y_i)$. For each $(s_b, s_{b^2}, \ldots, s_{b^k})$ (i.e. for each $b \in S_B$), compute

$$\hat{T}_{b_i} = x_i^c \left(\prod_{j=0}^{k} (E_{i,1})^{s_{b^j}} \right)^{s_{s_i}}, \; y_i^c \left(\prod_{j=0}^{k} (E_{i,2})^{s_{b^j}} \right)^{s_{s_i}} .$$

3. Compute $\hat{c} = H(\tilde{A}||\hat{U}||\hat{T}_{b_1}||\ldots||\hat{T}_{b_k})$, reject the proof if $\hat{c} \neq c$.
4. Check $s_e \in \pm \{0,1\}^{\ell'_e+\ell_\emptyset+\ell_H+2}$, $s_{\tilde{v}}, s_{b^j} \in \pm \{0,1\}^{\ell_m+\ell_\emptyset+\ell_H+3}$.

Creating P_3 (Step 11) Let $\mathcal{C}_1 = \{w_i : D(w_i) = 1\}$, $\mathcal{C}_y = \{w_i : D(w_i) \neq 1\}$. As noted in Remark 3, to prove that w_i is contained in $\mathcal{C}_y$, we need to reveal a blinded decryption of w_i; therefore we compute the the set $\mathcal{D}_y$ as follows: for each $w_i = (c_1, c_2) \in \mathcal{C}_y$, compute $d_i = c_1^{-xu_i} c_2^{u_i}$ where $u_i \in_R \mathbb{Z}_q^*$, and add d_i to $\mathcal{D}_y$. The proof will then show that $d_i = {c_1}^{a_i} {c_2}^{u_i} = {c_1}^{-xu_i} {c_2}^{u_i}$, where a_i is an element such that $1 = h^{u_i} g^{a_i}$, i.e., $a_i = -xu_i$ as $h = g^x$. Now, the proof that ${c_1}^{-x} c_2 = 1$ cannot be done directly, therefore the prover will assert that $c_2 = c_1^x$ (which is equivalent). Hence the proof we compute is

$$\begin{aligned} P_3 = \{(x, u_i, a_i) : c_2 = {c_1}^x \quad & \forall\, (c_1, c_2) \in \mathcal{C}_1 \\ \wedge \quad 1 = h^{u_i} g^{a_i} \quad & \forall\, i,\ w_i \in \mathcal{C}_y \\ \wedge \quad d_i = c_1^{a_i} c_2^{u_i} \quad & \forall\, d_i \in \mathcal{D}_y \\ \wedge \quad h = g^x \} \; . & \end{aligned}$$

1. Compute $t_x = g^{r_x}$ for $r_x \in_R \mathbb{Z}_q^*$.
2. Initialize a set T_1. For each $(c_1, c_2) \in \mathcal{C}_1$ compute $c_1^{r_x}$ and add the result to T_1.
3. Initialize a set T_C. For each i such that $w_i \in \mathcal{C}_y$, choose r_{a_i} and r_{u_i} at random from $\mathbb{Z}_q^*$. Compute $h^{r_{u_i}} g^{r_{a_i}}$ and add it to T_C.
4. Initialize a set T_y. For each $d_i \in \mathcal{D}_y$ compute $c_1^{r_{a_i}} {c_2}^{r_{u_i}}$, and add the result to T_y.
5. Compute $c = H(\mathcal{C}_1||\mathcal{C}_y||\mathcal{D}_y||t_x||T_1||T_C||T_y)$.
6. Compute (in $\mathbb{Z}_q$): $s_x = r_x - cx$, $s_{u_i} = r_{u_i} - cu_i$, and $s_{a_i} = r_{a_i} - ca_i$ for all i such that $d_i \in \mathcal{D}_y$.
7. Output the proof $P_3 = (c, s_x, s_{u_i}, s_{a_i})$.
8. Send P_3, the indices of $\mathcal{C}_1$, $\mathcal{C}_y$, the set $\mathcal{D}_y$ to the verifier.

Verifying P_3 (Step 13) Partition the ciphertexts into sets $\mathcal{C}_1$, $\mathcal{C}_y$, based on index information from the prover. Also ensure $1 \notin \mathcal{D}_y$.

1. Initialize a set $\hat{T}_1$. For each $(c_1, c_2) \in \mathcal{C}_1$ compute ${c_2}^c {c_1}^{s_x}$, and add this value to $\hat{T}_1$.
2. Initialize a set $\hat{T}_C$. For each i such that $w_i \in \mathcal{C}_y$, compute $h^{s_{u_i}} g^{s_{a_i}}$ and add it to $\hat{T}_C$.
3. Initialize a set $\hat{T}_y$. For each $(c_1, c_2) \in \mathcal{C}_y$ and the corresponding $d_i \in \mathcal{D}_y$, compute ${d_i}^c (c_1^{s_{a_i}} c_2^{s_{u_i}})$, and add the result to $\hat{T}_y$.
4. Compute $\hat{t}_x = h^c g^{s_x}$.
5. Compute $\hat{c} = H(\mathcal{C}_1||\mathcal{C}_y||\mathcal{D}_y||\hat{t}_x||\hat{T}_1||\hat{T}_C||\hat{T}_y)$. Reject the proof if $\hat{c} \neq c$.