Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection

Stanisław Jarecki and Xiaomin Liu

University of California, Irvine

Abstract. An Oblivious Pseudorandom Function (OPRF) [15] is a two-party protocol between sender S and receiver R for securely computing a pseudorandom function $f_k(\cdot)$ on key k contributed by S and input x contributed by R, in such a way that receiver R learns only the value $f_k(x)$ while sender S learns nothing from the interaction. In other words, an OPRF protocol for PRF $f_k(\cdot)$ is a secure computation for functionality $\mathcal{F}_{\mathsf{OPRF}} : (k, x) \rightarrow (\perp, f_k(x))$.
We propose an OPRF protocol on committed inputs which requires only $O(1)$ modular exponentiations, and has a constant number of communication rounds (two in ROM). Our protocol is secure in the CRS model under the Composite Decisional Residuosity (CDR) assumption, while the PRF itself is secure on a polynomially-sized domain under the Decisional q-Diffie-Hellman Inversion assumption on a group of composite order, where q is the size of the PRF domain, and it has a useful feature that f_k is an injection for every k.
A practical OPRF protocol for an injective PRF, even limited to a polynomially-sized domain, is a versatile tool with many uses in secure protocol design. We show that our OPRF implies a new practical fully-simulatable adaptive (and committed) OT protocol secure without ROM. In another example, this oblivious PRF construction implies the first secure computation protocol of set intersection on committed data with computational cost of $O(N)$ exponentiations where N is the maximum size of both data sets.

1 Introduction

PRF and Oblivious PRF. A pseudorandom function (PRF) [17] is an efficiently computable keyed function $f_k(\cdot)$ whose values are indistinguishable, for a randomly chosen key k, from random elements in the function range. The *oblivious* PRF, or OPRF [15], is a protocol that allows the sender S, on input the key k, to let the receiver R compute the value $f_k(x)$ of a PRF $f_k(\cdot)$ on any input x of R's choice without releasing any other information to R, and do so *obliviously* in the sense that sender S learns nothing from the protocol, similarly as in oblivious transfer [28, 14] or oblivious polynomial evaluation [24]. In other words, an OPRF protocol corresponding to a PRF function $f_k(\cdot)$ is a secure computation protocol for functionality $\mathcal{F}_{\mathsf{OPRF}} : (k, x) \rightarrow (\perp, f_k(x))$. To enforce consistency between several protocol instances it is helpful to extend the above fuctionality $\mathcal{F}_{\mathsf{OPRF}}$ to include verification whether k and x contributed by S and R correspond to some previously committed values. We call this extended functionality a *committed*

OPRF, and in parallel to public-key primitives we refer to the commitment to a PRF key k as a corresponding *public key* pk.

Examples of Oblivious PRF Applications: Keyword Search, Adaptive OT, SFE for Set Intersection. An oblivious PRF has numerous exciting applications. It was introduced as a primitive by Freedman et al. [15] with an application to privacy-preserving *Keyword Search*. This protocol problem is otherwise known as symmetrically-private Keyword PIR [12] or "Keyword OT", and it can be defined as a secure computation for the following functionality: Sender S contributes a set of (keyword,data) pairs $\{x_i, y_i\}_{i=1..N}$, where all x_i's are unique, receiver R contributes a keyword x, and the functionality gives data item y_i to R if $x_i = x$, or a symbol $\perp$ if all x_i's differ from x. The reduction of Keyword OT to Oblivious PRF is very simple provided that f_k is an injection for every k and that the keyword domain is polynomial-sized: The sender S picks two PRF keys k_1, k_2, publishes a set of pairs $\{(f_{k_1}(x_i), f_{k_2}(x_i) \oplus y_i)\}_{i=1..N}$ together with commitments pk_1, pk_2 and a proof of knowledge of the corresponding k_1, k_2. (The restriction that the keyword domain is polysized is needed for simulatability because it enables extraction of sender's inputs given keys k_1, k_2.) Then R computes $z_1 = f_{k_1}(x)$ and $z_2 = f_{k_2}(x)$ via two instances of the oblivious PRF protocol, uses z_1 to check if there exists an x_i in S's set s.t. $x_i = x$, and if so then it uses z_2 to recover the corresponding y_i.

Essentially the same protocol is also a solution to the *Adaptive OT* problem, introduced by Naor and Pinkas [22]. A (fully simulatable) adaptive OT is a secure computation protocol for a reactive functionality where S contributes a sequence of N values $Y = \{y_i\}_{i=1..N}$, and then S and R can engage in any number of 1-out-of-N OT protocol instances on these values and any index i contributed by R. An adaptive OT can be implemented using oblivious (committed) PRF if S publishes a sequence $\{f_k(i) \oplus y_i\}_{i=1..N}$, a commitment pk and a ZK proof of knowledge of the corresponding k, and for each OT instance the two parties run an OPRF protocol on S's key k and R's adaptively chosen index i, which lets R compute $f_k(i)$ and thus retrieve y_i. (This protocol is related to the ROM-based adaptive OT scheme of Camenisch, Neven, and Shelat [8], as we explain in subsection below.)

Another application of oblivious PRF's was recently shown by Hazay and Lindell [19], who use an oblivious PRF to construct a very simple protocol for a set intersection function, where S and R contribute their respective sets of N data items, X and Y, and the protocol lets R compute the intersection $X \cap Y$ without revealing anything else. As in any secure function evaluation (SFE), if the protocol ensures that both parties enter previously committed inputs then both parties can compute the output (without guaranteeing fairness) if the SFE protocol is run in both directions. In Hazay-Lindell protocol, S picks a PRF key k, sends to R a commitment pk to k and a set of outputs of the PRF function $f_k(\cdot)$ on all its inputs, $X' = \{f_k(x)\}_{x \in X}$. Note that these PRF values are indistinguishable from n random values in f's range, and hence in particular X' cannot be efficiently correlated with X. Players S and R engage then in n OPRF instances on respective inputs k and y for all $y \in Y$, allowing R to compute the set $Y' = \{f_k(y)\}_{y \in Y}$. If f_k is an injection then R can conclude that $y \in X \cap Y$ iff $y \in X' \cap Y'$. Moreover, if we use committed OPRF and S proves that values $f_k(x)$

correspond to committed inputs x, the result is an SFE protocol for set intersection on committed data sets.

Previous Work on Oblivious PRF's. The first oblivious PRF construction was given by Naor and Reingold [25], based on the PRF construction given in the same paper, $f_k(x) = g^{\prod_{x_i=1} r_i}$, where key k is the sequence of T random elements $r_1, ..., r_T$ in $\mathbb{Z}_q$, q is the order of the group $\langle g \rangle$ in which the Decisional Diffie-Hellman assumption holds, and T is the bitlength of elements x in the PRF domain. This first oblivious PRF protocol required $O(t)$ rounds. Freedman et al. [15], based on the previous work of Naor and Pinkas [21, 22], improved this to a constant-round protocol, secure under DDH. This protocol implements a "weak OPRF" notion (which is just as good in all the applications listed above), where the receiver is allowed to learn additional information about the PRF key as long as that information does not change the pseudorandomness of the PRF function on any new inputs. This protocol uses T parallel OT instances, one for each r_i value, which using best known OT techniques, e.g. [23, 1], translates into $O(T)$ exponentiations. The OT-batching techniques of Ishai et al. [20] do not seem to reduce this overhead in the case of a single OPRF instance if the PRF domain size is smaller or equal to the security parameter. The Hazay-Lindell set intersection protocol implemented with this OPRF involves NT parallel OT instances, but it is an open problem to efficiently use the OT-batching techniques of [20] in this context because the case of malicious adversaries seems to require batching *committed* OT instances and proving that the committed OT inputs correspond to the PRF key.

The ROM-based Adaptive OT construction from any unique blind signature scheme given by Camenisch et al. [8] relies on a weak OPRF protocol for f_k defined as $f_k(m) = H(\mathsf{SIG}_k(m))$, which is a PRF in ROM if the signature scheme is CMA-secure, where the weak OPRF functionality on inputs (k, m) releases the signature $\mathsf{SIG}_k(m)$ in addition to the proper PRF value $H(\mathsf{SIG}_k(m))$. However, for the resulting protocol to be secure computation of this functionality we need a secure computation protocol for the blind signature functionality, $\mathcal{F}_{\mathsf{BlSg}}(k, m) \rightarrow (\perp, \mathsf{Sig}_k(m))$. While there exist efficient unique blind signature schemes, e.g. by Chaum [10] or Boldyreva [3], these schemes rely on non-standard "one more" type of security assumptions, and it is an open problem to extend such blind signature protocols to secure computation of $\mathcal{F}_{\mathsf{BlSg}}$, and to ensure that the computation proceeds on committed receiver's input m.

An Adaptive OT construction of Green and Hohenberger from blind IBE schemes [18] can also be seen as variant of an OPRF protocol. Assuming the ROM model one could define $f_k(ID) = H(sk_{ID})$ where k is the KDC's master private key and sk_{ID} is the IBE decryption key corresponding to identity ID. If this is a blind IBE scheme then we could use the blinded private-key retrieval protocol as an implementation of a weak OPRF which releases sk_{ID} in addition to the hash value $H(sk_{ID})$. However, efficient IBE's seem to require bilinear maps, which are not needed for the OPRF protocol we present, and moreover it remains an open problem to upgrade such construction to secure computation of the ideal OPRF functionality, and to further extend it to computation on committed inputs.

Our Result: Efficient (Committed) Oblivious PRF. We propose an OPRF construction which requires only $O(1)$ modular exponentiations, and has a constant number of communication rounds (two in ROM). The secure computation protocol for func-

tionality $\mathcal{F}_{\mathsf{OPRF}} : (k, x) \rightarrow (\perp, f_k(x))$ for our PRF function family $f_k(\cdot)$ builds on the Camenisch-Shoup [9] version of Paillier encryption [26], and it is secure in the CRS model under the Composite Decisional Residuosity (CDR) assumption. The PRF $f_k(\cdot)$ itself is a variant of the PRF construction of Dodis-Yampolskiy [13] based on the Boneh-Boyen signature [4], but moved to a group of a composite order, the safe RSA modulus on which the Camenisch-Shoup encryption operates. As far as we know, this OPRF construction is the first constant-round efficient OPRF which is actually a secure computation protocol for the ideal OPRF functionality $\mathcal{F}_{\mathsf{OPRF}}$.

This PRF has a useful feature that f_k is an injection for every key k, and it secure on a domain of bitstrings of length $|q|$ under the Decisional q-Diffie-Hellman Inversion (q-DHI) assumption on a group whose order is a safe RSA modulus. Consequently, by positive hardness results of [9, 13] (see also related upper bounds on q-DHI hardness given by Jung Hee Cheon [11]), the domain of this PRF is polynomially-sized, but this restriction does not stop any of the OPRF applications we listed above. The one application it constrains is the secure computation of set intersection, where the inputs must be encoded in a polynomially-sized domain. Note that in many applications set intersection protocol might run on short inputs anyway, e.g. names, social security numbers, etc. However, extending the domain of this OPRF construction would allow secure computation of set intersection on larger input domains.

Another very useful feature of our OPRF construction is that it is an efficient *committed* OPRF, i.e. our secure computation protocol for extended version of the F_{OPRF} functionality which checks whether both parties contribute previously committed inputs into the protocol.

Technical Roadmap: We give a brief intuition for constructing an OPRF protocol for the Dodis-Yampolskiy PRF $f_k(x) = g^{1/(k+x)}$ in group $\langle \mathsf{g} \rangle$ of composite order n, and an encryption scheme which is additively homomorphic on message domain $\mathbb{Z}_n$, like Paillier encryption. We assume that this encryption allows for shared decryption. Namely, one can form a "joint key" $pk = pk_s \cdot pk_r$ from two public keys pk_s, pk_r, and the corresponding private keys sk_s, sk_r allow for shared decryption of ciphertext encrypted under pk. This is true of the Camenisch-Shoup version [9] of Paillier encryption. Using expressions like $C_v^{(r)}$, $C_v^{(s)}$, and C_v, to denote ciphertexts which encrypt variable v under, respectively, pk_r, pk_s, or pk, we have that partial decryption of C_v under sk_r creates $C_v^{(s)}$, and a partial decryption of C_v under sk_s creates $C_v^{(r)}$.

The idea of our construction goes like this: Sender S and receiver R exchange public keys pk_s and pk_r and encrypt their inputs k and x under the joint key $pk = pk_s \cdot pk_r$ as C_k and C_x. Then by the homomorphism of the encryption $C_k \cdot C_x = C_\alpha$ where $\alpha = k + x$. Player R then randomizes the encrypted value α by picking random a in $\mathbb{Z}_n$ and computing $C_\beta = (C_\alpha)^a$, for $\beta = a \cdot \alpha$. R also encrypts a under pk, as C_a, partially decrypts C_β into $C_\beta^{(s)}$, and sends $(C_a, C_\beta^{(s)})$ to S. Sender S decrypts $C_\beta^{(s)}$ to get β and computes $C_\sigma = (C_a)^{1/\beta}$, for $\sigma = a/\beta = 1/\alpha = 1/(k+x)$. Finally, S picks an additive share σ_s of σ, computes $v_s = \mathsf{g}^{\sigma_s}$, encrypts it as C_{σ_s} and computes $C_{\sigma_r} = C_\sigma / C_{\sigma_s}$, for $\sigma_r = \sigma - \sigma_s$. S also partially decrypts C_{σ_r} into $C_{\sigma_r}^{(r)}$ and sends $(v_s, C_{\sigma_r}^{(r)})$ to R. R decrypts $C_{\sigma_r}^{(r)}$ and uses σ_r to compute $v = v_s \cdot \mathsf{g}^{\sigma_r} = \mathsf{g}^{\sigma_s + \sigma_r} = \mathsf{g}^{1/(k+x)}$.

Our actual protocol streamlines these operations, and uses solely keys pk_r, pk_s instead of the joint key pk, but the above sketch is an idea behind our construction.

Related Concurrent Work: We note that independently from our work, Belenkiy et al. [2] recently showed a different protocol for oblivious computation of the same function $f_k(x) = g^{1/(k+x)}$, also using Paillier encryption. Their protocol is somewhat similar to ours but it uses multiplicative rather than additive sharing of the crucial exponent value $\sigma = 1/(k+x)$, and it is about twice faster than our protocol as a result. Moreover, the protocol of [2] can work on groups with a 160-bit prime order unrelated to the RSA modulus n, instead of a composite order we need, leading to further speed-ups in the applications of this OPRF. While the initial version of this protocol published in [2] is not a secure computation of the OPRF functionality, it is not difficult to modify this protocol to secure computation in the CRS model using techniques similar to ours.

Organization: We introduce our notation, security assumptions, and important definitions in Subsection 2.1. In Subsection 2.2 we show the main tool in our efficient OPRF protocol construction, an additively homomorphic encryption scheme with verifiable encryption and decryption, and in Subsection 2.3 we show an efficient instanciation of such scheme, i.e. the Camenisch-Shoup encryption scheme. In Section 3 we present our main contribution, a construction of an OPRF protocol. Finally we show two applications of a committed OPRF, to Adaptive OT and to secure computation of the Set Intersection problem, in Sections 4 and 5 respectively.

2 Preliminaries and Tools

2.1 Notation, Definitions, and Security Assumptions

Notation. We use $a \leftarrow A$ to denote that a is the output of the (randomized) algorithm A and $a \leftarrow_R S$ if a is chosen uniformly at random from set S. If A is an algorithm then we will sometimes use $A(x)$ to denote a set of all possible outputs of A on input x. We use $P\{b\}$ to denote a zero-knowledge proof system that statement b holds, and $PoK\{a \mid \phi(a)\}$ to denote a zero-knowledge proof of knowledge of value a that satisfies a publicly computable relation ϕ. Finally, most numerical operations in the paper are group operations unless specifically noted otherwise.

Factoring Assumption (Definition). Let RSAGen denote an algorithm that picks safe RSA moduli with a given security parameter. Namely, $\mathsf{RSAGen}(1^\kappa)$ chooses two random primes p'_1, p'_2 s.t. $|p'_1| = |p'_2| = \kappa$ and $p_1 = 2p'_1 + 1$ and $p_2 = 2p'_2 + 1$ are also primes, and outputs $n = p_1 \cdot p_2$. We say that *factoring safe RSA moduli is hard* if for every efficient algorithm A the probability $\Pr[A(n) \in \{p_1, p_2\} \mid n \leftarrow \mathsf{RSAGen}(1^\kappa)]$ is a negligible function of κ.

Decisional q-Diffie-Hellman Inversion (q-DHI) Problem (Definition). The *computational* q-DHI problem in a group with generator g and order n is to compute $\mathsf{g}^{1/\alpha}$ given the tuple $(\mathsf{g}, \mathsf{g}^\alpha, \ldots, \mathsf{g}^{(\alpha^q)})$, for random α in $\mathbb{Z}_n^*$. We define the hardness of the *decisional* version of this problem for any fixed constant q as follows: Let gGen be an algorithm which on input a security parameter κ picks a modulus n and a generator g of a multiplicative group $\mathbb{G}$ of order n. For example gGen can be a composition of

RSAGen and an algorithm gGen' which on input n output by RSAGen finds the first prime p s.t. $n|p-1$, and sets g as any element of order n in $\mathbb{Z}_p^*$. We say that *the Decisional q-DHI Assumption holds on group (family)* $\mathbb{G}$, if for every efficient algorithm $\mathcal{A}$ function $\epsilon_A(\kappa) = |\mathsf{Real}_{\mathcal{A}}(\kappa) - \mathsf{Random}_{\mathcal{A}}(\kappa)|$ is negligible, where:

$$\mathsf{Real}_{\mathcal{A}}(\kappa) = \Pr\left[\mathcal{A}(\mathsf{g}, \mathsf{g}^{\alpha}, \ldots, \mathsf{g}^{(\alpha^q)}, \mathsf{g}^{1/\alpha}) = 1 \mid (\mathsf{g}, n) \leftarrow \mathsf{gGen}(1^{\kappa});\ \alpha \leftarrow \mathbb{Z}_n^*\right]$$

$$\mathsf{Random}_{\mathcal{A}}(\kappa) = \Pr\left[\mathcal{A}(\mathsf{g}, \mathsf{g}^{\alpha}, \ldots, \mathsf{g}^{(\alpha^q)}, \mathsf{h}) = 1 \quad \middle| \begin{array}{l}(\mathsf{g}, n) \leftarrow \mathsf{gGen}(1^{\kappa});\ \alpha \leftarrow \mathbb{Z}_n^*; \\ \mathsf{h} \leftarrow \mathbb{G}\end{array}\right]$$

Pseudorandom Function (Definition). For notational simplicity we consider a version of the general PRF notion which is custom-made to fit the PRF implementation we consider in this paper. Namely, the PRF function f_k maps $|q|$-bit strings to elements of group $\mathbb{G}$. We say that *function (family) f_k defined by the key generation algorithm* KGen *is a PRF* if $|\mathsf{Real}_{\mathcal{A}}(\kappa) - \mathsf{Random}_{\mathcal{A}}(\kappa)|$ is a negligible function of κ, where:

$$\mathsf{Real}_{\mathcal{A}}(\kappa) \ = \ \Pr\left[\mathcal{A}^{f_k(\cdot|\neg x)}(v, \mathsf{st}) = 1 \,\middle|\, \begin{array}{l}(k, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^{\kappa}); \\ (x, \mathsf{st}) \leftarrow A^{f_k(\cdot)}(\mathsf{pk});\ \ v \leftarrow f_k(x)\end{array}\right]$$

$$\mathsf{Random}_{A}(\kappa) \ = \ \Pr\left[\mathcal{A}^{f_k(\cdot|\neg x)}(v, \mathsf{st}) = 1 \,\middle|\, \begin{array}{l}(k, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^{\kappa}); \\ (x, \mathsf{st}) \leftarrow A^{f_k(\cdot)}(n, \mathsf{g}, \mathsf{pk});\ \ v \leftarrow \mathbb{G}\end{array}\right]$$

In the above experiments $f_k(\cdot|\neg x)$ denotes an oracle $f_k(\cdot)$ modified to output $\perp$ on x.

Dodis-Yampolskiy's PRF and Boneh-Boyen's Function in Composite-Order Groups. Our OPRF construction relies on a variant of the PRF scheme of Dodis-Yampolskiy [13], based on the Boneh-Boyen unpredictable function [4], with the sole modification being a substitution of a prime-order group with a group whose order is a safe RSA modulus. The Boneh-Boyen function [4] is $f_k(x) = \mathsf{g}^{1/(k+x)}$ where g generates a group $\mathbb{G}$ of prime order p and k is a random element in $\mathbb{Z}_p$. This function is unpredictable under the *computational* q-DHI assumption on $\mathbb{G}$. Dodis-Yampolskiy [13] considered the same function as a source of a verifiable pseudorandom function (VRF) in a group with a bilinear map, and showed that it is secure under a decisional version of the q-DHI assumption on the target group of the bilinear map. However, the same argument also shows that the decisional q-DHI assumption on group $\mathbb{G}$ itself implies that the Boneh-Boyen function is a PRF. Morevoer, the same arguments which were done by [4, 13] for prime-order groups also imply that (1) the Boneh-Boyen function in a composite-order group remains a PRF under the decisional q-DHI assumption on such groups (and hardness of factoring) and (2) the same generic-group argument which motivated trust in the q-DHI assumption on the prime-order groups carries to composite-order groups as well.

Specifically, for security parameter κ we define the following PRF function family: The key generation algorithm picks $n \leftarrow \mathsf{RSAGen}(1^{\kappa})$, $\mathsf{g} \leftarrow \mathsf{gGen}'(n)$, $k \leftarrow \mathbb{Z}_n^*$, and computes $\mathsf{pk} \leftarrow \mathsf{g}^k$. We define $f_k(x)$ for each $x \in \{0,1\}^{|q|}$ as follows:

$$f_k(x) = \begin{cases} \mathsf{g}^{1/(k+x)} \text{ if } gcd(k+x, n) = 1 \\ 1 \ \text{ otherwise} \end{cases}$$

Claim 1: The Decisional q-DHI Assumption holds on a generic group (family) with a safe RSA order.

Argument Sketch: This claim can be verified by inspecting the generic-group argument for the computational q-DHI problem on a prime-order group given in [4], because generic arguments for decisional rather than computational problems are identical, and all that this specific argument requires is that the group order has only large factors.

Claim 2: The function (family) f_k defined above is a PRF on the domain of q-domain strings if factoring safe RSA moduli is hard and if the Decisional q-DHI Assumption holds on group (family) $\mathbb{G}$.

Argument Sketch: Assuming decisional q-DHI holds in group $\mathbb{G}$ of order a safe RSA modulus n, the argument why $f_k(x)$ in $\mathbb{G}$ is a PRF follows from the argument of Theorem 1 in [13] and the fact that under the assumption of hardness of safe RSA moduli, efficient algorithms can encounter elements v s.t. $gcd(v, n) \neq 1$ only with negligible probability. The reduction given in the proof of Theorem 1 in [13] uses inverses $(\alpha - x_{i^*} + x_i)$ in the exponent for all x_i in the domain of f_k, where x_{i^*} is the value for which the adversary must distinguish $f_k(x_{i^*})$ from a random element in $\mathbb{G}$. Note that by the factoring assumption the probability that in this game there appears i s.t. $(\alpha - x_{i^*} + x_i)$ is *not* co-prime with n is negligible. Thus the argument for pseudo-randomness of $f_k(x) = g^{1/(k+x)}$ shown in [13] for prime-order groups extends to groups whose order is hard to factor.

2.2 Additively Homomorphic Verifiable Encryption with Additional Properties

In order to construct an oblivious PRF, we need an additively homomorphic encryption scheme with *verifiable encryption* and *verifiable decryption*.

- $\mathsf{Setup}(\kappa)$ on security parameter κ outputs the public parameter par to be used in all subsequent algorithms. This par also defines the plaintext space $\mathcal{M}$, a finite additive group $\mathbb{Z}_n$ for some n.
  For notational simplicity, we omit explicit mention of par as the input to all the following algorithms.
- KGen outputs a random public/secret key pair (pk, sk). Parameters par also defines a relation KVal (for key validity) on all valid (pk, sk). We also require an efficient proof system
$$PoK\{sk \mid (pk, sk) \in \mathsf{KVal}\} \tag{1}$$
- $\mathsf{Enc}_{pk}(m)$ on public key pk and message m outputs a ciphertext C which is a random encryption of m under pk. We require an efficient realization of the following proof system given a public key pk and ciphertext C
$$PoK\{m \mid C \in \mathsf{Enc}_{pk}(m)\} \tag{2}$$
- $\mathsf{Dec}_{sk}(C)$ is a deterministic algorithm on secret key sk and ciphertext C that outputs a message m. We require an efficient realization of the following proof system given a public key pk and ciphertext C
$$P\{\exists\ sk,\ \text{s.t. } m = \mathsf{Dec}_{sk}(C)\ \wedge\ (pk, sk) \in \mathsf{KVal}\} \tag{3}$$

Besides semantic security, we require the following properties of the above encryption scheme:

- **Additive Homomorphism:** We require that there is an efficient operation on ciphertexts, which for convenience we denote as a multiplication, s.t. $\mathsf{Enc}_{pk}(m_0) \cdot \mathsf{Enc}_{pk}(m_1) \in \mathsf{Enc}_{pk}(m_0+m_1)$. We can also define exponentiation and division operations on ciphertexts, and by homomorphism of the encryption $(\mathsf{Enc}_{pk}(m))^a = \mathsf{Enc}_{pk}(a \cdot m)$ and $\mathsf{Enc}_{pk}(m_0)/\mathsf{Enc}_{pk}(m_1) \in \mathsf{Enc}_{pk}(m_0 - m_1)$, for any a and any m, m_0, m_1 in $\mathcal{M}$.
- **Verifiable Encryption:** We require an efficient realization of the following proof system, given public key pk, ciphertext C, and two elements g and y of some multiplicative group of order $|\mathcal{M}| = n$:

$$PoK\{m \mid C \in \mathsf{Enc}_{pk}(m) \ \wedge \ \mathsf{y} = \mathsf{g}^m)\} \tag{4}$$

  In addition, we require an efficient realization of the following proof system, given public keys pk and pk' and ciphertexts C_1, C_2 and C', where C_1 and C_2 are supposed to be ciphertexts under public key pk.

$$PoK\{m \mid \exists\, m' \in \mathbb{Z}_n, \text{ s.t. } C' = \mathsf{Enc}_{pk'}(m') \ \wedge \ C_2 \in (C_1 \cdot \mathsf{Enc}_{pk}(m))^{m'}\} \tag{5}$$

- **Verifiable Decryption:** We require an efficient realization of the following proof system, given public keys pk and pk' and ciphertexts C, C_1 and C_2, where C is supposed to be ciphertext under public key pk and C_1 and C_2 are supposed to be ciphertexts under public key pk':

$$P\left\{\begin{array}{l}\exists\, m, m' \in \mathbb{Z}_n, m'', sk \text{ s.t. } m = \mathsf{Dec}_{sk}(C) \ \wedge \ (pk, sk) \in \mathsf{KVal} \\ \wedge\, C_1 \in (C_2 \cdot \mathsf{Enc}_{pk'}(m'))^{m''} \ \wedge \ \mathsf{y} = \mathsf{g}^{m'} \ \wedge \ m \cdot m'' = 1 \bmod n\end{array}\right\} \tag{6}$$

2.3 Efficient Instantiation Using Camenisch-Shoup Encryption

The above encryption scheme can be efficiently instantiated with just the semantically secure version of Camenisch-Shoup Encryption [9].

- $\mathsf{Setup}(\kappa)$ generates public parameter $\mathsf{par} = (g, n)$, where n is safe RSA modulo, i.e. $n = p_1 \cdot p_2$, $p_1 = 2p_1' + 1$, $p_2 = 2p_2' + 1$, and p_1, p_2, p_1' and p_2' are all primes, and g is of order $p_1'p_2'$. Let $h = n + 1$ and $n' = p_1'p_2'$. The message space $\mathcal{M}$ is the additive group $\mathbb{Z}_n$.
- $\mathsf{KGen}(\mathsf{par})$ picks random x in $[0, \frac{n}{4}]$, computes $y \leftarrow g^x$, and sets $pk = y$ and $sk = x$.
- $\mathsf{Enc}_y(m)$ for $m \in \mathbb{Z}_n$, picks random $r \in [0, \frac{n}{4}]$, and outputs a ciphertext $C = (u, e) = (g^r, y^r h^m)$.
- $\mathsf{Dec}_x(u, e)$ computes $\hat{m} \leftarrow (e/u^x)^2$. If $\hat{m} \notin \langle h \rangle$ (i.e. if n does not divides $\hat{m} - 1$), it rejects the ciphertext. Otherwise, it sets $\hat{m}' \leftarrow \frac{\hat{m}-1}{n}$ (over integers), computes $\gamma \leftarrow \beta^{-1} \bmod n$, and outputs $m = \hat{m}'/2 \cdot \gamma \bmod n$

Semantic security of this encryption holds under the Composite Residuosity Assumption on $\mathbb{Z}_{n^2}^*$ [9]. It also satisfies all other properties listed in Section 2.2.

– This encryption scheme is additively homomorphic. The corresponding operation on the ciphertext is a pair-wise multiplication of the two components, i.e. if $C_1 = (u_1, e_1)$ and $C_2 = (u_2, e_2)$ then $C_1 \cdot C_2 = (u_1 \cdot u_2,\ e_1 \cdot e_2)$. Similarly $C_1/C_2 = (u_1/u_2, e_1/e_2)$ and $(C_1)^a = ((u_1)^a, (e_1)^a)$.
– All the proof systems listed in Section 2.2 can be realized for this encryption scheme by proof systems requiring $O(1)$ exponentiations from each party. We defer description of these proofs to the full version of this paper, but all these proof systems are very similar to the verifiable encryption proof system given by Camenisch-Shoup in [9]. Such HVZK proofs can be converted to ZK proof systems using known compilation techniques, or into Non-Interactive ZK using the Fiat-Shamir heuristic in the Random Oracle Model. We note that the proof systems resulting from the last compilation are as efficient as the underlying HVZK proof systems, and they remain zero-knowledge and simulation sound under parallel composition.

3 Construction of an OPRF Protocol

We show the construction of a secure computation protocol of the ideal OPRF functionality for the PRF defined in the previous section, using the additively homomorphic encryption scheme with verifiable encryption, decryption, and other useful properties listed in Section 2.2. This protocol is illustrated in Figure 1, with all the proof systems denoted non-interactively for notational simplicity.

Theorem 1. *Assuming hardness of factoring of safe RSA moduli, a semantically secure encryption scheme on $\mathbb{Z}_n$ which satisfies the properties listed in Section 2.2, and assuming that each proof (of knowledge) system in Figure 1 is zero-knowledge and (strong) simulation-sound, the protocol in Figure 1 is a secure computation protocol for functionality $\mathcal{F}_{\mathsf{OPRF}}$.*

Proof. Constructing an ideal-world sender SIM_s from a malicious real sender S^*:
First, we show the construction of the ideal-world sender SIM_s which interacts with the real world sender S^* and the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$. SIM_s proceeds as follows:

– If S^* succeeds in the proof π_1, then SIM_s runs the extractor algorithm for π_1 with S^* to extract k, s.t. $\mathsf{pk} = \mathsf{g}^k$.
– Then SIM_s simulates the real-world receiver R as follows:
  1. $(pk_r, sk_r) \leftarrow \mathsf{KGen}$
  2. $a \leftarrow_R \mathbb{Z}_n^*$, $C_a^{(r)} \leftarrow \mathsf{Enc}_{pk_r}(a)$
  3. $\beta \leftarrow_R \mathbb{Z}_n^*$, $C_\beta^{(s)} \leftarrow \mathsf{Enc}_{pk_s}(\beta)$
  4. $C_\beta^{(s)} \leftarrow \mathsf{Enc}_{pk_s}(\beta)$
  5. Send $(pk_r, C_a^{(r)}, C_\beta^{(s)})$ and simulate the proof π_2.
– If the proof π_3 verifies, then SIM_s sends k to $\mathcal{F}_{\mathsf{OPRF}}$. Note that $\mathcal{F}_{\mathsf{OPRF}}$ on SIM_s's input k and ideal-world receiver $\bar{\mathsf{R}}$'s input x outputs $f_k(x)$ to $\bar{\mathsf{R}}$.

Let $\mathcal{Z}$ be a distinguisher that controls the sender S^*, feeds the input of the receiver R, and also sees the output of R. Now we argue that $\mathcal{Z}$'s view in the real world (S^*'s view + R's output) and its view in the idea world (S^*'s view + ideal receiver $\bar{\mathsf{R}}$'s output) are

indistinguishable. This is done by showing a series of games $\mathsf{Game}_0, \ldots, \mathsf{Game}_5$, each interacting with $\mathcal{Z}$, where each Game_{i+1} modifies Game_i slightly, and arguing that $\mathcal{Z}$'s views in Game_i and Game_{i+1} are indistinguishable, where Game_0 runs S^* together with the real receiver R's protocol, while Game_5 runs the above simulator SIM_s (with oracle access to S^*), which simulates an ideal world sender, the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$, and the ideal receiver $\bar{\mathsf{R}}$.

Game_1: Same as Game_0 except that instead of proving π_2, R simulates it. By zero-knowledge of the π_2 proofs system, $\mathcal{Z}$'s views in Game_0 and Game_1 are indistinguishable.

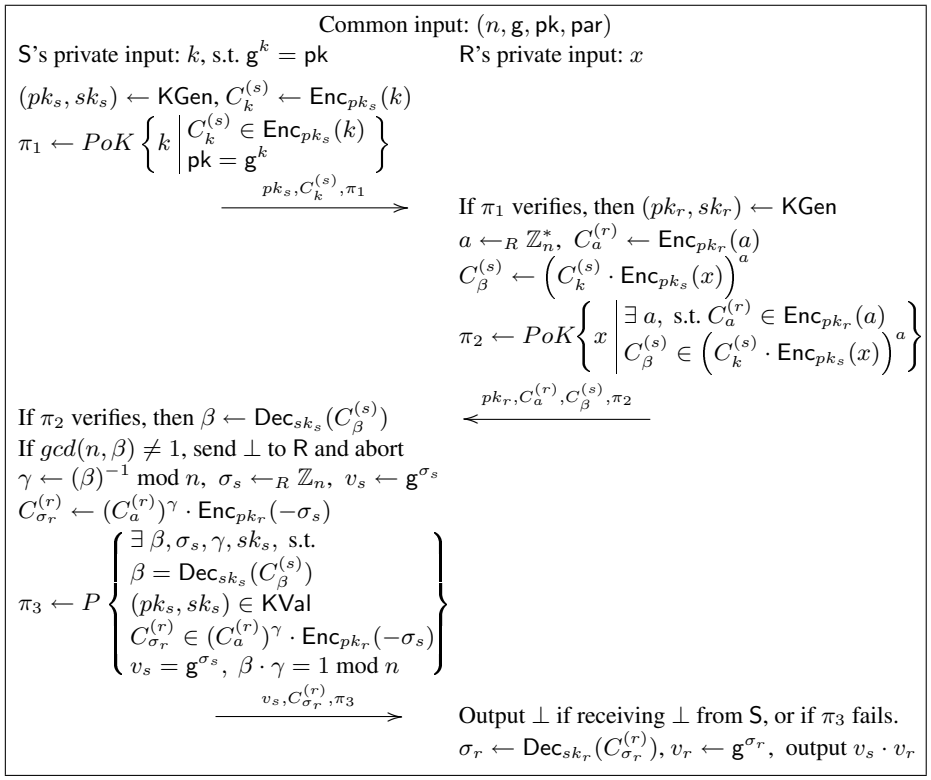

Fig. 1. Construction of an OPRF Protocol

Game_2: Same as Game_1 except that (a.) If S^* succeeds in the proof π_1, then Game_2 runs the extractor algorithm for π_1 with S^* to extract k, s.t. $\mathsf{pk} = \mathsf{g}^k$; and (b) If the proof π_3 verifies, then Game_2 outputs $f_k(x) = \mathsf{g}^{1/(k+x)}$ as its final output (or $\perp$ if $gcd(k+x, n) \neq 1$). Note that Game_2 knows both k and x. For (a), by strong simulation soundness of the proof system π_1, Game_2 extracts k with non-negligible probability. For (b), by simulation soundness of proof system π_3, $\mathcal{Z}$'s views in Game_1 and Game_2 are indistinguishable.

Game_3: Same as Game_2 except that as long as $gcd(k+x, n) = 1$, Game_3 does the following:

1. $(pk_r, sk_r) \leftarrow \mathsf{KGen}$
2. $\beta \leftarrow_R \mathbb{Z}_n^*$, $C_\beta^{(s)} \leftarrow \mathsf{Enc}_{pk_s}(\beta)$
3. $a \leftarrow \beta/(k+x)$, $C_a^{(r)} \leftarrow \mathsf{Enc}_{pk_r}(a)$
4. Simulate the proof π_2.

Note that the probability that $gcd(k+x, n) \neq 1$ is negligible if factoring safe RSA moduli is hard, and if $gcd(k+x, n) = 1$ then the tuple $(pk_r, C_a^{(r)}, C_\beta^{(s)})$ is distributed identically in Game_2 and Game_3, so $\mathcal{Z}$'s view of these two games are indistinguishable.

Game_4: Same as Game_3 except that in line 3 above, value a is replaced by random $a' \in \mathbb{Z}_n^*$. We claim that by semantic security of the encryption scheme, $\mathcal{Z}$'s view in Game_3 and Game_4 are indistinguishable. A reduction Red can be constructed as follows. Getting the challenger's public key pk^*, it sets $pk_r = pk^*$ and follows Game_3 except that when C_a is to be computed, it sends (a, a') to the challenger and gets back the challenge C^*. It sets $C_a^{(r)} \leftarrow C^*$, and continue following Game_3. If $\mathcal{Z}$ distinguishes Game_3 and Game_4, then Red breaks the semantic security of the encryption scheme.

Game_5: Game_5 is the ideal world game between SIM_s (with access to S^*), $\mathcal{F}_{\mathsf{OPRF}}$, and the ideal world receiver $\bar{\mathsf{R}}$. Instead of computing $v = f_k(x)$ as the last step of Game_4 (this modification is shown in the description of Game_2 Step b), in Game_5, let SIM_s follow the protocol in Game_4 until $f_k(x)$ is to be computed, then send k to $\mathcal{F}_{\mathsf{OPRF}}$. $\mathcal{F}_{\mathsf{OPRF}}$ computes $v = f_k(x)$ on k given by SIM_s and x given by $\bar{\mathsf{R}}$, and sends v to $\bar{\mathsf{R}}$. It is easy to see that $\mathcal{Z}$'s view in Game_4 and Game_5 are identical.

Constructing an ideal-world receiver SIM_r from a malicious real-world receiver R^*:
We first describe the construction of SIM_r:

- SIM_r picks $(pk_s, sk_s) \leftarrow_R \mathsf{KGen}$, random k' in $\mathcal{M}$, sets $C_k^{(s)} \leftarrow \mathsf{Enc}_{pk_s}(k')$, sends pk_s and $C_k^{(s)}$ to R^*, and simulate the proof π_1.
- If the proof π_2 verifies, SIM_r runs the extractor algorithm of π_2 with R^* to extract x and sends it to $\mathcal{F}_{\mathsf{OPRF}}$.
- Getting $v = f_k(x)$ from $\mathcal{F}_{\mathsf{OPRF}}$, which computes v on ideal-world sender $\bar{\mathsf{S}}$'s input k and SIM_r's input x, SIM_r does the following:
  1. If $f_k(x) = 1$, then SIM_r sends $\perp$ to R^* and aborts.
  2. $\sigma_r \leftarrow_R \mathbb{Z}_n$
  3. $v_s \leftarrow v/\mathsf{g}^{\sigma_r}$
  4. $C_{\sigma_r}^{(r)} \leftarrow \mathsf{Enc}_{pk_r}(\sigma_r)$
  5. send $(v_s, C_{\sigma_r}^{(r)})$ and simulate the proof π_3.

Let $\mathcal{Z}$ be a distinguisher that controls the receiver R^*, feeds the input of the sender S, and also sees the output of S. We still show a series of games to argue that the environment $\mathcal{Z}$'s view in the real protocol and its view in the ideal world protocol is indistinguishable. Let Game_0 be the protocol executed by the real world sender, and let Game_4 be the ideal world game.

Game_1*:* Game_1 is the same as Game_0 except that S instead of proving π_1 and π_3, it simulates the two proofs. By zero-knowledge (simulatability) of these proof systems, $\mathcal{Z}$'s view in Game_0 and its view in Game_1 are indistinguishable.

Game_2*:* Game_2 is the same as Game_1 except that if the proof π_2 verifies, Game_2 extracts x from R^* (The probability that Game_2 extracts x is non-negligible because of simulation soundness of the proof system π_2). $\mathcal{Z}$'s views in Game_1 and Game_2 are indistinguishable.

Game_3*:* Game_3 is the same as Game_2 except it does the following after extracting x:

1. $v = f_k(x)$; if $v = 1$, send $\perp$ to R^* and abort.
2. $\sigma_r \leftarrow_R \mathbb{Z}_n$, $v_s \leftarrow v/\mathsf{g}^{\sigma_r}$
3. $C^{(r)}_{\sigma_r} \leftarrow \mathsf{Enc}_{pk_r}(\sigma_r)$
4. send $(v_s, C^{(r)}_{\sigma_r})$ and simulate the proof π_3.

Note that $f_k(x) = 1$ if and only if $gcd(k + x, n) \neq 1$, and if $gcd(k + x, n) \neq 1$, then $\beta = a(k + x)$ for any a is not co-prime with n, i.e. $gcd(\beta, n) \neq 1$, so the real sender (as well as Game_2) in this case will also send $\perp$ to R^* and abort. If $v \neq 1$, the pair $(v_s, C^{(r)}_{\sigma_r})$ distributes identically in Game_2 and Game_3. Therefore, $\mathcal{Z}$'s view in Game_2 and Game_3 are identical.

Game_4*:* In Game_4, we let the simulator SIM_r to follow the protocol in Game_3 except that when v is to be computed, SIM_r sends the extracted x to the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$ which also gets input k from the ideal world sender $\bar{\mathsf{S}}$, and gets the value $v = f_k(x)$ from the functionality $\mathcal{F}_{\mathsf{OPRF}}$ instead. Then SIM_r continues following the protocol in Game_3. It is easy to see that $\mathcal{Z}$'s view in Game_2 and Game_3 are identical, and Game_3 is in fact the ideal world game among ideal sender S, the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$ and the ideal receiver SIM_r who has oracle access to R^* and simulates R^* in the ideal world.

Extension to secure computation of $\mathcal{F}_{\mathsf{OPRF}}$ on *committed* inputs. Note that in our implementation of the OPRF functionality the sender's input k is already committed in the key $\mathsf{pk} = g^k$, but only a slight modification to the protocol is needed if we extend the OPRF functionality so that the receiver's input x is committed as well. The receiver can commit to x using Pedersen commitment [27] in group $\langle \mathsf{g} \rangle$ of order n, i.e. $\mathsf{Com}_x = \mathsf{g}^x \mathsf{h}^{r_x}$ for random $\mathsf{h} \in \langle \mathsf{g} \rangle$ specified in the CRS (or picked by the sender) and random $r_x \in \mathbb{Z}_n$. The proof system π_2 in receiver's first step must then be extended to a proof of knowledge of not just x but also r_x s.t. $\mathsf{Com}_x = g^x \mathsf{h}^{r_x}$. Using the hiding and binding properties of Pedersen commitment it is not difficult to extend our proof of security of the basic OPRF protocol in Figure 1 to this case. Finally, such as proof system is easy to realize because the order of group $\langle \mathsf{g} \rangle$ is the same n which acts on the plaintext in encryption $\mathsf{Enc}_{pk}(x)$. Using techniques for proving equality between exponents in groups of different orders, e.g. [5, 7], one can extend it to accommodate commitments to x using more standard groups of smaller (and prime) order.

Parallel composition and re-using sender's first message. This OPRF scheme can be composed sequentially because it is a secure computation protocol. This directly implies security of the adaptive OT protocol based on such OPRF, as explained in Section 4 below. However, by inspection of the proof, using standard hybrid arguments one can argue that this OPRF scheme can also be composed in parallel, provided that the ZK proof systems it uses remain zero-knowledge and simulation-sound under parallel composition. The resulting protocol is a secure computation of the "parallel OPRF"

functionality, where the receiver enters a sequence of data items $(x_1, \ldots, x_t)$ and gets a sequence of PRF values $f_k(x_1), \ldots, f_k(x_n)$ in return. The secure computation for this parallel OPRF functionality enables constant-round secure computation protocol for set intersection, as we explain in Section 5.

It is also easy to see that one can reduce the round complexity and/or the sender's computation's time in both of these applications if the sender re-uses the same pk_s, sk_s, and $C_k^{(s)}$ values in each instance of the OPRF subprotocol. One can think of this as a case of sender's re-using the same randomness when executing the first step of the protocol. Since we cannot prevent a malicious sender from doing so anyway, such modified protocol remains secure against a malicious sender. For malicious receiver, one can see by inspection of the proof of Theorem 1 that this re-use does not change anything in the security proof. Note moreover that in the ROM model one can realize the proof π_1 non-interactively as a tuple of a few group elements. Therefore in ROM the sender's first message can be included as a common input to the protocol, which results in a 2-round OPRF protocol in ROM.

4 Adaptive Oblivious Transfer from an OPRF Scheme

In this section, we show a construction of t-out-of-N OT protocol from an OPRF protocol, where the sender takes N messages $(m_1, \ldots, m_N)$ as input, while receiver takes t indices $(i_1, \ldots, i_t)$ as input. At the end of the protocol, receiver gets $\{m_{i_j}\}_{j=1,\ldots,t}$, while sender gets nothing. An OT protocol is adaptive if receiver can query on indices $i_1, \ldots, i_t$ adaptively one after another. Using an OPRF protocol, we show below the construction of an adaptive OT protocol and in Figure 2.

- Let (g, n) be the common input.
- Sender picks random $k \in \mathbb{Z}_n$, sets $\mathsf{pk} = \mathsf{g}^k$, and then computes $c_i = m_i \cdot f_k(i)$ for every $i = 1, \ldots, N$. It sends pk, $\{c_i\}_{i=1,\ldots,N}$ to the receiver.
- For $j = 1, \ldots, t$:
  - Sender and receiver interacts in an OPRF with sender's input k and receiver's input i_j;
  - Receiver gets $v_{i_j} = f_k(i_j)$ and recovers m_{i_j} by computing c_{i_j}/v_{i_j}.

As we point out at the end of Section 3, the OPRF protocol of that section takes only two-rounds in ROM if the sender re-uses the first message of the OPRF protocol in each instance. Therefore the combination of these two protocols results in an optimal two-round adaptive OT.

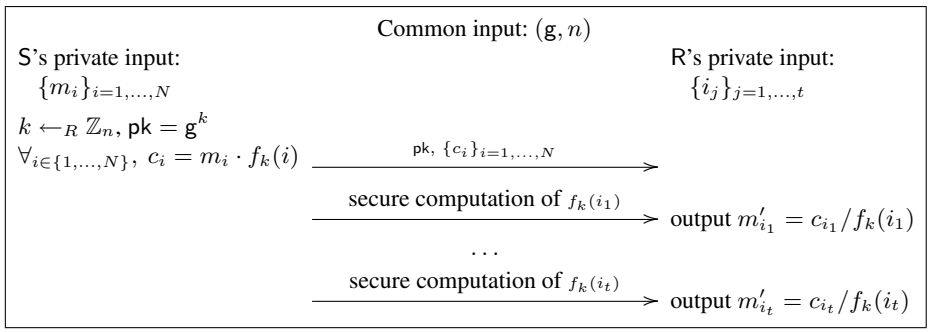

Fig. 2. Adaptive OT Protocol from an OPRF Protocol

Theorem 2. *The above construction of OT is a secure computation of the ideal t-out-of-N OT functionality*

$$\mathcal{F}_{\mathsf{OT}}\left(\{m_i\}_{i=1,\ldots,N}\ ,\ \{i_j\}_{j=1,\ldots,t}\right) = (\perp\ ,\ \{m_{i_j}\}_{j=1,\ldots,t})$$

Proof. We argue this theorem in the hybrid model using the secure computation of OPRF as a blackbox.

Constructing ideal world sender SIM_s from a malicious sender S^* in the real world: Getting k from S^*, SIM_s computes $v_i = f_k(i)$ for each $i = 1, \ldots, N$, and sends $\{\bar{m}_i = c_i/v_i\}_{i=1,\ldots,N}$ to $\mathcal{F}_{\mathsf{OT}}$, which on $\{\bar{m}_i\}_{i=1,\ldots,N}$ from SIM_s and $\{i_j\}_{j=1,\ldots,t}$ from the ideal world receiver R, outputs $\{\bar{m}_{i_j}\}_{j=1,\ldots,t}$ to the receiver. In the real world, what the receiver R outputs is $\{c_{i_j}/f_k(i_j)\}_{j=1,\ldots,t}$, which is also $\{\bar{m}_{i_j}\}_{j=1,\ldots,t}$. S^* learns nothing either interacting with the real world receiver R via the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$ or interacting with SIM_s in the ideal world. Therefore, the environment $\mathcal{Z}$'s views in the real world and ideal world are indistinguishable.

Constructing ideal world receiver SIM_r from a malicious receiver R^* in the real world: SIM_r first sends to R^* a random pk' in $\langle \mathsf{g} \rangle$ as well as a tuple of $\{r_i\}_{i=1,\ldots,N}$ where each r_i is random in the range of $\mathcal{F}_{\mathsf{OPRF}}$. On getting each i_j for $j = 1, \ldots, t$, SIM_r sends $\{i_j\}_{j=1,\ldots,t}$ to $\mathcal{F}_{\mathsf{OT}}$ and gets back $\{m_{i_j}\}_{j=1,\ldots,t}$. Then SIM_r sends $\{r_{i_j}/m_{i_j}\}_{j=1,\ldots,t}$ to R^*. As $\mathcal{F}_{\mathsf{OPRF}}$ is a pseudorandom function, R^*'s view in the real protocol ($\{c_i\}_{i=1,\ldots,N}$, $\{f_k(i_j)\}_{j=1,\ldots,t}$) is indistinguishable from ($\{r_i\}_{i=1,\ldots,n}$, $\{r_{i_j}/m_{i_j}\}_{j=1,\ldots,t}$) for each r_i chosen at random in the range of $\mathcal{F}_{\mathsf{OPRF}}$. Since both the real world sender S and ideal world sender $\bar{\mathsf{S}}$ output a $\perp$, the environment $\mathcal{Z}$'s views in the real world and ideal world are indistinguishable.

5 Secure Computation of Set Intersection from an OPRF Scheme

Using the Hazay-Lindell construction [19] one can easily convert a secure computation protocol for OPRF functionality into secure computation of the set intersection problem. Let $M_s = \{m_i^{(s)}\}_{i=1,\ldots,N}$ and $M_r = \{m_i^{(r)}\}_{j=1,\ldots,N}$ denote, respectively, the sender's and the receiver's data sets. The secure set intersection protocol should allow the receiver to compute $M_s \cap M_r$ while the sender gets nothing from the interaction. Using a secure computation protocol for parallel OPRF functionality for the PRF from Section 3), the construction goes as follows, on common inputs (g, n):

- Sender picks random $k \in \mathbb{Z}_n$ and sets $\mathsf{pk} = \mathsf{g}^k$. Then it computes $v_i^{(s)} = f_k(m_i^{(s)})$ for every $i = 1, \ldots, N$, sets $V_s = \{v_i^{(s)}\}_{i=1,\ldots,N}$ and sets $V_s' = \Pi(V_s)$, where Π is a random permutation. It sends pk and V_s' to the receiver.
- Sender and receiver interact in a parallel OPRF protocol with sender's input k and receiver's inputs $({m_1}^{(r)}, ..., {m_N}^{(r)})$. Let $({v_1}^{(r)}, ..., {v_N}^{(r)})$ be receiver's outputs;
- Receiver outputs the set $\left\{m_j^{(r)} \text{ s.t. } v_j^{(r)} \in V_s'\right\}$

Theorem 3. *The above construction is a secure computation protocol for functionality*

$$\mathcal{F}_{\mathsf{SI}}(M_s,\ M_r) = (\perp,\ M_s \cap M_r)$$

Proof. We argue this theorem in a hybrid model, where the sender and receiver communicate using the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$ when they invoke the OPRF protocol.

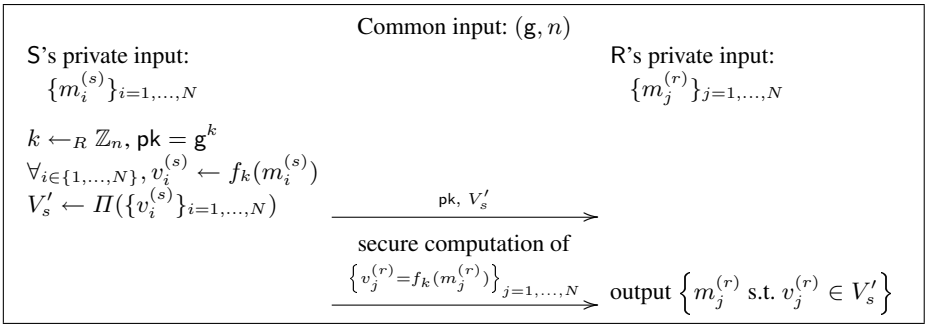

Fig. 3. Computing Set Intersection from an OPRF protocol

Constructing ideal world sender SIM_s *from a malicious sender* S^* *in the real world:* SIM_s first gets pk and the set $V_s' = \{v_i^{(s)}\}_{i=1,\ldots,N}$ from S^*. Then when getting the key k from S^*, SIM_s tries every possible input in the range of the PRF to reconstruct the set $\bar{M}_s = \{\bar{m}_i^{(s)}\}_{i=1,\ldots,N}$ from $V_s' = \{v_i^{(s)}\}_{i=1,\ldots,N}$, and sends $\bar{M}_s$ to $\mathcal{F}_{\mathsf{SI}}$, which computes $\bar{M}_s \cap M_r$ on SIM_s's input $\bar{M}_s$ and the ideal world receiver $\bar{\mathsf{R}}$'s input M_r, and sends $\bar{M}_s \cap M_r$ to $\bar{\mathsf{R}}$. The real world receiver R in the hybrid model gets $v_i^{(s)} = f_k(m_i^{(s)})$ for every $\bar{m}_i^{(s)} \in \bar{M}_s$ as well as $v_j^{(r)} = f_k(m_j^{(r)})$ for every $j = 1, \ldots, N$. So R's output is also $\bar{M}_s \cap M_r$. S^* learns nothing either interacting with the real world receiver R via the ideal functionality $\mathcal{F}_{\mathsf{OPRF}}$ or interacting with SIM_s in the ideal world. Therefore, the environment $\mathcal{Z}$'s views in the real world and ideal world are indistinguishable.

Constructing ideal world receiver SIM_r *from a malicious receiver* R^* *in the real world:* SIM_r first sends to R^* a random pk' in $\langle \mathsf{g} \rangle$ as well as a tuple of $\{r_i\}_{i=1,\ldots,N}$ where each r_i is random in the range of the pseudorandom function. On getting $M_r = \{m_i^{(r)}\}_{j=1,\ldots,N}$ from the receiver R^*, it sends M_r to the ideal functionality $\mathcal{F}_{\mathsf{SI}}$, which computes $M_s \cap M_r$ on the ideal world sender S's input M_s and SIM_r's input M_r, and sends $M_s \cap M_r$ to SIM_r. For each $m_j^{(r)} \in M_r$, if $m_j^{(r)} \in M_s \cap M_r$, then SIM_r sets $\bar{v}_j^{(r)} = r_i$ for some (previously not picked) i; otherwise it sets $\bar{v}_j^{(r)}$ at random in the range of the pseudorandom function. SIM_r sends $\{\bar{v}_j^{(r)}\}_{j=1,\ldots,N}$ to R^*. Because of $\mathcal{F}_{\mathsf{OPRF}}$ is a pseudorandom function, R^*'s view in the real protocol $(\{\Pi(v_i^{(s)}\}_{i=1,\ldots,N}), \{\bar{v}_i^{(r)}\}_{i=1,\ldots,N})$ is indistinguishable from $(\{r_i\}_{i=1,\ldots,n},\ \{\bar{v}_i^{(r)}\}_{i=1,\ldots,N})$ in the simulated game above. Since both the real world sender S and ideal world sender $\bar{\mathsf{S}}$ output a $\perp$, the environment $\mathcal{Z}$'s views in the real world and ideal world are indistinguishable.

Extension to computing on committed inputs: As in the OPRF protocol in Section 3, it is easy to extend this protocol so both parties execute on committed inputs. The receiver can be forced to execute on committed inputs if we replace the basic OPRF functionality by the committed OPRF, as sketched as the end of Section 3. And if $\{\mathsf{Com}_i\}_{i=1,\ldots,N}$ are Pedersen commitments on the sender's inputs in the same group $\langle \mathsf{g} \rangle$ of order n, then it is easy to extend the sender's first message with a ZK proof of knowledge of values m_i, for each i, s.t. $v_i^{(s)} = f_k(m_i) = g^{1/(k+m_i)}$ where $pk = g^k$ and m_i is committed in Com_i. Note that since we use the same commitment for sender's and receiver's inputs, both parties can securely (but not fairly) compute the set intersection

if they run the same protocol twice, with the roles reversed. Another use of computing on committed data is for the sender to be able to verify that the receiver holds some authorization on the values it enters into the computation, e.g. in the form of a signature, or an unlinkable credential [6], on the commitments to these values.

Extension to transfer of associated data, or "Index OT": In addition to letting the receiver discover any item ${m_i}^{(s)}$ which it shares with the sender, the receiver can also get some data d_i which the sender associates with this item. If the receiver has only a single item then this protocol problem is known as "index OT". The advantage of the protocol given here is that it has $O(N)$ complexity if both parties contribute N items. Assume that each d_i is an element of group $\langle \mathsf{g} \rangle$. The sender generates two PRF keys k_1 and k_2 and publishes $(f_{k_1}({m_i}^{(s)}), f_{k_2}({m_i}^{(s)}) * d_i)$ for each i. For each receiver's item ${m_j}^{(r)}$ the two parties engage in *two* instances of the OPRF protocol, on two sender's keys k_1 and k_2. The receiver then uses $f_{k_1}({m_j}^{(r)})$ to decide if there exists ${m_i}^{(s)} = {m_j}^{(r)}$, and $f_{k_2}({m_j}^{(r)})$ to retrieve the data d_i associated with ${m_i}^{(s)}$.

5.1 Efficiency Estimation of the Set Intersection Protocol

It is interesting to compare the set intersection protocol resulting from the oblivious PRF protocol of Figure 1 with the FNP protocol by Friedman et al. [16], both in the honest-but-curious and the malicious models. We compare only straightforward implementations of both schemes, and summarize this comparison in table 4. Clearly, it is possible that either algorithm can be further optimized.

Let N_s, N_r denote the number of entries in, respectively, the sender's and the receiver's data sets, and let k be the number of bits needed for representing each entry. Assume that the FNP protocol is implemented using ElGamal encryption over a 160-bit subgroup of $\mathbb{Z}^*_{p'}$ for a 1024-bit prime p'. Note that multiplications in the OPRF protocol in Figure 1 are either modulo a 2048-bit Paillier modulus n^2 or modulo the first prime p s.t. n divides $p-1$. Therefore if m is the cost of a single multiplication (or squaring) modulo a 1024-bit modulus then we can assume that these two different mults in our scheme cost about $3\mathsf{m}$ and m, respectively.

In the honest-but-curious model the main bottleneck of the FNP protocol is the oblivious evaluation of the receiver's encrypted polynomial, with N_r coefficients, on N_s points in the sender's database. Using Horner's rule, this would take $2N_sN_r$ of k-bit exponentiations, i.e. $3kN_sN_r\mathsf{m}$. In the malicious model, using standard commitments and zero-knowledge proofs of arithmetic relationships between committed values, it seems that each k-bit exponentiation would have to be replaced by at least two 160-bit exponentiations, so the total cost would grow by at least a factor of $320/k$.

In the set intersection protocol of Figure 3, using the OPRF of Figure 1, the sender's costs in the honest-but-curious model consist of N_s evaluations of the PRF (N_s fixed-base 1000-bit exponentiations, contributing $500N_s\mathsf{m}$), N_r Paillier decryptions (N_r 1000-bit exponentiations mod n^2, contributing $4500N_r\mathsf{m}$), and N_r computations of v_s values ($500N_r\mathsf{m}$) and $C^{(r)}_{\delta_r}$ values (this is a mixture of variable and fixed-base exponentiations which we estimate to amount to under $13000N_r\mathsf{m}$), for the total cost of $18000N_r\mathsf{m}$. For the receiver we have N_r computations of $C^{(r)}_a$ and $C^{(s)}_\beta$ values and N_r Paillier decryptions and computations of v_r values. This is a mixture of variable and fixed-base exponentiations modulo n^2 and p, for the estimated total cost of $14000N_r\mathsf{m}$. Thus we

estimate the total cost of this protocol in the passive model as $(500N_s + 32000N_r)\mathsf{m}$. In the malicious model this cost grows by only a factor of 2.

	FNP [16]	our protocol
honest-but-curious model	$3kN_sN_r\mathsf{m}$	$(500N_s + 32000N_r)\mathsf{m}$
malicious model	$\geq 960N_sN_r\mathsf{m}$	$(1000N_s + 64000N_r)\mathsf{m}$

Fig. 4. Comparison of Efficiency between the FNP protocol [16] and the one proposed here.

Summarizing these estimations, it seems that if N_s and N_r are comparable then the new protocol is faster than FNP in the honest-but-curious model for N_s on the order of $11000/k$, while in the malicious model the new protocol should be faster for $N_s \geq 67$.

References

1. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, 2001.
2. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Delegatable anonymous credentials. Cryptology ePrint Archive, Report 2008/428, 2008.
3. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, 2003.
4. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, 2004.
5. Fabrice Boudot and Jacques Traor. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *ICICS*, 1999.
6. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *SCN*, 2002.
7. Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO*, 1999.
8. Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, 2007.
9. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, 2003.
10. David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.
11. Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *EUROCRYPT*, 2006.
12. Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. *Cryptology ePrint Archive, 1998/003*, 1998.
13. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography*, 2005.
14. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of ACM*, 28(6), 1985.
15. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, 2005.
16. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

17. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4), 1986.
18. Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, 2007.
19. Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, 2008.
20. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, 2003.
21. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *STOC*, 1999.
22. Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO*, 1999.
23. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, 2001.
24. Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5), 2006.
25. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2), 2004.
26. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. EUROCRYPT*, 1999.
27. T.P. Pedersen. Non-interactive and information-theoretical secure verifiable secret sharing. In *Proceedings of Crypto'91*, 1991.
28. Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Harvard University, 1981.