Partial Garbling Schemes and Their Applications

Yuval Ishai [⋆] and Hoeteck Wee [⋆⋆]

[1] Technion, Haifa, Israel
[2] ENS, Paris, France

Abstract. Garbling schemes (aka randomized encodings of functions) represent a function F by a "simpler" randomized function F such that $F(x)$ reveals $F(x)$ and no additional information about x. Garbling schemes have found applications in many areas of cryptography. Motivated by the goal of improving the efficiency of garbling schemes, we make the following contributions:

- We suggest a general new notion of partial garbling which unifies several previous notions from the literature, including standard garbling schemes, secret sharing schemes, and "conditional disclosure of secrets". This notion considers garbling schemes in which part of the input is public, in the sense that it can be leaked by F.
- We present constructions of partial garbling schemes for (boolean and arithmetic) formulas and branching programs which take advantage of the public input to gain better efficiency.
- We demonstrate the usefulness of the new notion by presenting applications to efficient attribute-based encryption, delegation, and secure computation. In each of these applications, we obtain either new schemes for larger classes of functions or efficiency improvements from quadratic to linear. In particular, we obtain the first ABE scheme in bilinear groups for arithmetic formulas, as well as more efficient delegation schemes for boolean and arithmetic branching programs.

1 Introduction

There are many situations in cryptography where one is interested in computing some function F of a sensitive input x but the computational model is restricted so that only "simple" functions F can be directly computed. For instance, the entries of x may be encrypted so that only affine functions can be computed, or they may be distributed between multiple non-interacting parties so that only local functions can be computed.

A common approach for handling more complex functions F in such situations is to relax the usual notion of computation. This is done by: (1) settling for computing some function F whose output encodes the output of F (and reveals nothing else about the input), and (2) allowing the latter encoding to be randomized. That is, the randomized function $F(x)$ should satisfy the simplicity constraint (e.g., being affine) for every fixed choice of the randomness,[3] and moreover its output distribution on an input x should reveal $F(x)$ and no additional information about x. The function F is often referred to as a randomized encoding or a garbling scheme for F.

To give a simple example, let $F(a,b) = ab$ where a and b are elements of a finite field. Then an affine garbling scheme for F can be defined by $F(a,b) = ((a - r_a),(b - r_b), ar_b + br_a - r_ar_b)$, where r_a and r_b are

[⋆] . Research received funding from the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC. Also supported by ISF grant 1361/10 and BSF grant 2012378.

[⋆⋆] . CNRS (UMR 8548) and INRIA. Supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and NSF Awards CNS-1237429 and CNS-1319021. Part of this work was done while visiting the Technion, supported by ERC-CaC.

[3] Some applications require that F be "simple" even as a function of both x and the randomness; however, in this work we will be mainly interested in complexity in terms of x.

random and independent field elements. Note that F is an affine function of the input (a,b) for every fixed choice of r_a, r_b. Moreover, $F(a,b)$ can be recovered from the output (c,d,e) of $F(a,b)$ by computing $cd - e$. Finally, the output (c,d,e) of F is distributed uniformly subject to the constraint that $cd - e = ab$ and hence it reveals no additional information about (a,b) other than ab.

Garbling schemes have found applications in many areas of cryptography and elsewhere (see [38, 16, 25, 3, 2, 7, 33] and references therein). Starting with Yao's celebrated garbled circuit construction, different constructions of garbling schemes have been proposed for circuits and other representation models. However, these constructions still have theoretical and practical limitations. In particular, they do not efficiently generalize to arithmetic computations (despite progress in [5]) and even in the boolean case their asymptotic and concrete efficiency leave much to be desired. Furthermore, some of the best garbling schemes do not satisfy all of the structural properties that are needed by applications.

1.1 New notion: Partial garbling

This work is motivated by the observations that (1) in many applications of garbling schemes, most of the input is already known to the designated receiver of the encoded output, and (2) known constructions do not take advantage of this fact for improving efficiency.

We suggest a general new notion of partial garbling which relaxes standard garbling by allowing part of the input to be public. This notion can be viewed as unifying several previous notions from the literature, including standard garbling schemes, secret sharing schemes [35, 27], and conditional disclosure of secrets [20].

Consider for instance the case of secret sharing. Here, we want to disclose a secret if and only if the attributes of the parties satisfy a given predicate referred to as the access structure; however, it is okay to leak information about the individual attributes, which we think of as being "public". Note that "public" does not mean "known to everyone" or "must be revealed by the garbling" but rather "okay to leak by the garbling" and known to the reconstruction algorithm. For this reason, partial garbling cannot be viewed as a special case of standard garbling.

As another example, consider a client who has her data x distributed between several servers. Suppose that the servers wish to disclose a secret s to the client only if $P(x) = 1$ for some publicly known predicate P. That is, they would like to reveal to the client the function $F(x,s) = P(x) \cdot s$. Note that from the client's point of view, x is public input for F whereas s is a secret input. Now, suppose we are given a local partial garbling scheme $F(x,s)$ for F, namely one where each output depends only on the view of a single server. The partial garbling F can be used by the servers to conditionally disclose s to the client by each sending her a single message. This "conditional disclosure" primitive serves as a useful building block in both two-party and multi-party cryptographic protocols, where it can often serve as a light-weight, non-interactive substitute for zero-knowledge proofs [20, 1, 37, 15, 11, 12].

Typical applications of garbling schemes, including those illustrated above, require that the garbling be "affine" and/or "local" with respect to the public inputs. This additional requirement rules out the trivial solution of garbling a restriction of F to the private inputs.

Garbling arithmetic branching programs (ABP). We present unconditional constructions of partial garbling schemes for boolean and arithmetic formulas and branching programs which take advantage of the public input to gain better efficiency. Our constructions satisfy all of the useful structural properties of garbling schemes required by natural applications.

In cases where the private input is small, as is the case for essentially all of our motivating applications, we improve the size of the garbling from quadratic to linear in the size of the formula or branching program; we also obtain a corresponding efficiency improvement in the applications. Boolean and arithmetic formulas and branching programs capture many functions of interest, including arithmetic computations like sparse polynomials, mean, and variance, as well as combinatorial computations like string-matching, finite automata and decision trees.

Our partial garbling schemes are "affine" and "local" with respect to the public inputs; indeed, all of our applications exploit this property. As mentioned earlier, this means that we cannot simply garble the original function hardwired with the private inputs. Instead, we start with the randomized encoding scheme for ABPs in [26]. Roughly speaking, this prior construction works by multiplying the adjacency matrix for the branching program by random upper triangular matrices on both the left and the right. Our construction uses a subgroup of upper triangular matrices with fewer non-zero entries (corresponding to the private inputs). This means that the size of the garbling is roughly the number of private inputs times the size of the branching program. Therefore, when the number of private inputs is constant (or "local"), the size of the garbling improves from quadratic to linear. In particular, we achieve linear-size garbling for functions of the form $F(x,s) := P(x)\cdot s$ and $F(x,(s_1,s_2)) := s_1 P(x)^\top s_2$ where P is an ABP acting on a public input x; these are precisely the functions we consider for several of our applications.

Comparison with standard garbling techniques. We note that standard garbling is a special case of partial garbling where all input is private. In the other direction, there is a general reduction from partially garbling $F(x,z)$ where x is public and z is private to garbling $F'(x,z) := (x, F(x,z))$. However, the cost of the reduction can be significant. Take the example where F is identically 0. Then, a partial garbling scheme can output nothing whereas a standard garbling of F' must reveal x.

Unlike Yao's garbling technique and its variants, our constructions cannot handle general circuits and are restricted to weaker computational models. However, they do offer a number of significant advantages: (i) they do not rely on computational assumptions and can be used in the context of information-theoretic cryptography; (ii) they work in an arithmetic model of computation, where the number of field operations is independent of the field size; (iii) they satisfy the "linear reconstruction" property required by several applications below; (iv) they have better concrete efficiency for natural functions F that admit compact representations by formulas and branching programs.

1.2 Applications

We demonstrate the usefulness of our new notion and constructions by presenting applications to efficient attribute-based encryption (ABE) [34, 23], delegation [21, 19], and secure computation [24, 20] in Sections 5, 6, 7. More broadly, partial garbling can be applicable in many cryptographic settings in which there are computations that mix public inputs with private inputs. In each of the applications we consider, we obtain either new schemes for larger classes of functions or efficiency improvements from quadratic to linear:

- For ABE, we extend prior pairing-based schemes for boolean formulas and branching programs [23, 22] to arithmetic branching programs.
- For delegation, conditional disclosure of secrets and generalized oblivious transfer, we obtain efficiency improvements for arithmetic branching programs from quadratic to linear; prior to this work, constructions with linear complexity were only known for boolean formulas [19, 20, 36].

We proceed with an overview of the applications to delegation and ABE.

Delegation and verifiable computation. In verifiable computation (VC), a computationally weak client with input x wishes to delegate a complex computation f to an untrusted server, with the assurance that the server cannot convince the client to accept an incorrect computation [21, 19, 4, 9]. We focus on the online/offline setting, where the protocol proceeds in two phases. In the offline phase, the client sends to the server a possibly long message that may be expensive to compute. Later on, in the online phase (when the input x arrives), the client sends a short message to the server, and receives the result of the computation together with a certificate for correctness. We are interested in protocols where the client's communication and computational complexity in the online phase depend only on the input and output lengths and is independent of the complexity of f.

Our VC schemes build upon the garble+MAC paradigm in [4], which derives a VC protocol by garbling the function obtained by composing f with a one-time MAC. Our key observation is that it suffices to use a partial garbling scheme where the public input is x and the private input is the MAC key. Using our partial garbling schemes, we then derive more efficient online/offline VC protocols for arithmetic branching programs (ABPs), reducing the complexity of previous protocols from quadratic to linear in the size of the program. We note that ABPs simultaneously capture several classes of functions considered in the literature on delegation, including boolean formulas in [19, 32] and sparse arithmetic polynomials in [9].

> **Theorem 1** (informal). *Assuming the existence of a PRG, there is an online/offline VC protocol for arithmetic branching programs with the following efficiency features. The complexity of the server and the client's offline phase is $s \cdot \mathrm{poly}(\lambda)$ and that of the client's online phase is $n \cdot \mathrm{poly}(\lambda)$, where n is the input length, s is the size of the ABP, and λ is the security parameter.*

In [4], the complexity of the server and the client's offline phase is $s^2 \cdot \mathrm{poly}(\lambda)$. We also obtain a smaller improvement for boolean formulas; see Section 6 for details.

Attribute-based encryption. Attribute-based encryption (ABE) [34, 23] is a new paradigm for public-key encryption that enables fine-grained access control for encrypted data. In ABE, ciphertexts are associated with descriptive values x in addition to a plaintext, secret keys are associated with predicates P, and a secret key decrypts the ciphertext if and only if $P(x) = 1$. Here, P may express an arbitrarily complex access policy, which is in stark contrast to traditional public-key encryption, where access is all or nothing. The security requirement for ABE enforces resilience to collusion attacks, namely any group of users holding secret keys for different functions learns nothing about the plaintext if none of them is individually authorized to decrypt the ciphertext.

We present the first ABE that directly handles a large class of predicates over arithmetic domains as described by arithmetic branching programs. This is particularly useful in settings where identities or attributes come from a universe of exponential size, since we can avoid the overhead from using bit encodings, as with the case for the Boneh-Boyen identity-based encryption [13].

> **Theorem 2** (informal). *Suppose the decisional bilinear Diffie-Hellman assumption holds. Then, there exists a (selectively secure) ABE scheme for the class of arithmetic branching programs.*

Note that there are two natural ways to associate an ABP with a predicate, namely whether its output is zero (Z-ABP), or non-zero (N-ABP). We obtain ABE schemes for both via a single construction for "arithmetic span programs," which simultaneously generalizes boolean branching programs in [23] as well as (public-index) inner product and non-zero inner product predicates [29, 6]. Prior to this work, we do not know any ABE schemes in bilinear groups supporting the class of ABPs. We could of course appeal to lattice-based ABE for general circuits [22, 17], though simulating an ABP using a boolean circuit incurs a substantial overhead in concrete efficiency.

At a high level, our construction follows the approach of Goyal et al. [23] for building ABE for monotone Boolean formula from linear secret-sharing schemes. The difficulty with extending this approach to arithmetic branching programs is that there is no natural analogue of LSSS for arithmetic functionalities. Instead, we observe that it suffices to use partial garbling with linear reconstruction, which we do obtain for arithmetic branching programs. Intuitively, the descriptive value x on the ABE ciphertext corresponds to the public input in partial garbling, and the plaintext/master secret key corresponds to the private input.

The running time of the encryption algorithm depends only on the input length to the ABP and not the size of the ABP. As such, exploiting the connection between ABE and delegation in [32] and using the fact that we handle both Z-ABP and N-ABP, we obtain a publicly verifiable delegation scheme for ABPs. This scheme requires a stronger assumption than the offline/online VC in Theorem 1, but achieves a stronger soundness requirement with reusability.

Finally, our construction yields an unconditionally secure witness encryption scheme [18] for algebraic languages corresponding to vectors of group elements g^{w} such that $P(w) = 0$ for a fixed ABP P. For instance, this captures ElGamal public key and ciphertext pair $(\ ,C)$ such that C is an encryption of 0 or 1; see [12, 8, 10] for additional examples of such languages. The construction follows essentially from conditional disclosure of secrets schemes for the same predicate, along with the fact that reconstruction is linear.

Related work. In an independent work, Boneh et al. [14] constructed ABE for arithmetic circuits under the LWE assumption; they only handle the "is zero" predicate (i.e., decryption is possible exactly when the output of the circuit is zero), whereas our construction also handles the "is non-zero" predicate. Handling a class that is closed under complement is useful for applications such as publicly verifiable delegation [32], as noted in the preceding paragraph.

2 Preliminaries

Notation. We denote by $s \leftarrow_R S$ the fact that s is picked uniformly at random from a finite set S and by $x, y, z \leftarrow_R S$ that all x, y, z are picked independently and uniformly at random from S. By PPT, we denote a probabilistic polynomial-time algorithm. Throughout, we use 1^{λ} as the security parameter.

Arithmetic branching programs. A branching program is defined by a directed acyclic graph (V, E), two special vertices $v_0, v_1 \in V$ and a labeling function ℓ. An arithmetic branching program (ABP)[4] over a finite field F_q computes a function $f : F_q^n \to F_q$. Here, ℓ assigns to each edge in E an affine function in some input variable or a constant, and $f(x)$ is the sum over all v_0-v_1 paths of the product of all the values along the path. We refer to $|V| + |E|$ as the size of the ABP. Ishai and Kushilevitz [24, 26] showed how to relate an ABP computation to that of computing the determinant of a matrix (see Figure 1 for an example).

[4] Also referred to as mod-q counting branching programs in [26].

Lemma 1 ([26, Lemma 1]). Given an ABP $\Gamma = (V, E, v_0, v_1, \ell)$ computing $f : \mathbb{F}_q^n \to \mathbb{F}_q$, we can efficiently (and deterministically) compute a function $L(x)$ mapping an input $x \in \mathbb{F}_q^n$ to a $(|V|-1) \times (|V|-1)$ matrix over $\mathbb{F}_q$, such that:

- $\det(L(x)) = f(x)$;
- each entry of $L(x)$ is a degree one polynomial in a single variable x_i;
- $L(x)$ contains only -1's in the second diagonal (the diagonal below the main diagonal) and 0's below the second diagonal.

Specifically, L is obtained by removing the column corresponding to v_0 and the row corresponding to v_1 in the matrix $A - I$, where A is the adjacency matrix for Γ.

We note that there is a linear-time algorithm that converts any boolean formula, boolean branching program or arithmetic formula to an arithmetic branching program with a constant blow-up in the representation size. Thus, ABPs can be viewed as a stronger computational model than all of the above.

3 Partial Garbling Schemes (PGS)

We consider garbling schemes (aka randomized encodings of functions) [38, 16, 25, 3, 7] in which part of the input is public; we refer to this as partial garbling. Take a function F where the input (x, z) comprises a public value x and a private value z. In a standard garbling $\hat{F}$ of F, the function F is randomized and the output distribution $\hat{F}(x, z)$ "encodes" $F(x, z)$ and leaks no additional information about the input (x, z). In a partial garbling $\hat{F}$ of F, the value x is public, and the privacy requirement only applies to z. Again, we require that $\hat{F}(x, z)$ "encodes" $F(x, z)$, and that it leaks no additional information about the private input z beyond what is revealed by x and $F(x, z)$. More formally, we require that there be two efficiently computable maps Sim and Rec where Sim is randomized and Rec is deterministic such that

- the distributions $\mathsf{Sim}(x, F(x, z))$ and $\hat{F}(x, z)$ are identical;
- $\mathsf{Rec}(x, \hat{F}(x, z; r)) = F(x, z)$ for all inputs (x, z) and randomness r.

We want constructions where $\hat{F}$ is a "simple" function of (x, z). We are also particularly interested in constructions where the reconstruction algorithm $\mathsf{Rec}(x, \cdot)$ is a "simple" function of $\hat{F}(x, z)$, e.g. a function of total degree 1 whose coefficients may depend arbitrarily on x.

Definition 1 ((affine) partial garbling scheme). Let $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ be a function. We say that a randomized function $\hat{F} : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q^m$ is a partial garbling scheme (PGS) of F if it satisfies the following properties:

- (correctness) There exists a deterministic reconstruction algorithm Rec such that for all $(x, z) \in \mathbb{F}_q^n \times \mathbb{F}_q^{n'}$, $\Pr[\mathsf{Rec}(x, \hat{F}(x, z)) = F(x, z)] = 1$ (where the probability is taken over the coin tosses of $\hat{F}$).
- (privacy) There exists a randomized algorithm Sim, called a simulator, such that for all $(x, z) \in \mathbb{F}_q^n \times \mathbb{F}_q^{n'}$, $\mathsf{Sim}(x, F(x, z))$ and $\hat{F}(x, z)$ are identically distributed.

In addition, we say that the garbling scheme is affine if for all indices $j \in [m]$, $\hat{F}(x, z)_j$ is an affine function of the form $a_j x_i + b_j$ or $a_j z_i + b_j$ where the coefficients $a_j, b_j \in \mathbb{F}_q$ depend only on the randomness of $\hat{F}$.

We will also say that the garbling scheme is x-affine if each $F(x,z)_j$ is an affine function of the form $a_j x_i + b_j$ where the coefficients $a_j, b_j \in \mathbb{F}_q$ depend only on z and the randomness of F. We may define z-affine analogously. Note that an affine garbling scheme is both x-affine and z-affine.

We note that the above definition extends naturally to functions F with longer outputs. When considering an infinite family of F, we also require that there is an efficient deterministic garbling algorithm for computing the description of F, Rec and Sim from that of F.

Examples. As a warm-up, we describe several partial garbling schemes for functions with $n' = 1$ and $n' = 2$, some of which were implicit in prior works. All of these schemes are captured by our more general construction in Section 4.

Example 1 (non-zero product). Consider the function

$$F((x_1,x_2,x_3),z) = x_1x_2x_3z.$$

This corresponds to disclosing a secret z subject to the condition x_1,x_2,x_3 are all non-zero. Consider the affine garbling scheme:

$$F((x_1,x_2,x_3),z;r_1,r_2,r_3) = (r_1x_1, r_1 - r_2x_2, r_2 - r_3x_3, r_3 - z))$$

Reconstruction has degree 1 and is given by a dot product with the vector

$$(-1, x_1, x_1x_2, x_1x_2x_3).$$

Example 2 (sum is zero). Consider the function $F : \mathbb{F}_q^n \times \mathbb{F}_q^2 \to \mathbb{F}_q$ given by

$$F((x_1,\ldots,x_n),(z,z')) = z' \cdot (x_1 + \cdots + x_n) + z.$$

This corresponds to disclosing a secret z subject to the condition $x_1 + \ldots + x_n = 0$; for $n = 2$, this captures "disclose z if x_1, x_2 are equal". Consider the x-affine encoding:

$$F((x_1,\ldots,x_n),(z,z');r_1,\ldots,r_n) = (z'x_1 - r_1,\ldots,z'x_n - r_n, r_1 + \cdots + r_n + z)$$

Reconstruction has degree 1 and is given by summing the values in the encoding. This is essentially the scheme given in [20, Lemma 2].

Example 3 (non-zero inner product). Consider the function $F_{y_1,\ldots,y_n} : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q$ given by

$$F_{y_1,\ldots,y_n}(x_1,\ldots,x_n,z) = (x_1y_1 + \cdots + x_ny_n)z$$

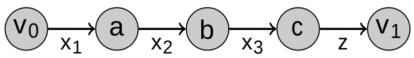
This corresponds to disclosing a secret z subject to the condition $x_1y_1 + \cdots + x_ny_n \neq 0$, that is, the inner product is non-zero. Consider the z-affine encoding:

$$F_{y_1,\ldots,y_n}(x_1,\ldots,x_n,z;w_1,\ldots,w_n) = (y_1z - w_1,\ldots,y_nz - w_n, x_1w_1 + \cdots + x_nw_n)$$

Reconstruction is given by a dot product with the vector $(x_1y_1 + \cdots + x_ny_n)^{-1}(x_1,\ldots,x_n,1)$. This encoding is not affine, but can be readily transformed into an affine encoding (at the price of increasing the randomness complexity and output length by $n - 1$). A variant of this encoding was used implicitly in [6, 31] for constructing revocation and negated inner product cryptosystems with short ciphertexts (the first n outputs of F are associated with the secret key, whereas the last output is associated with the ciphertext).

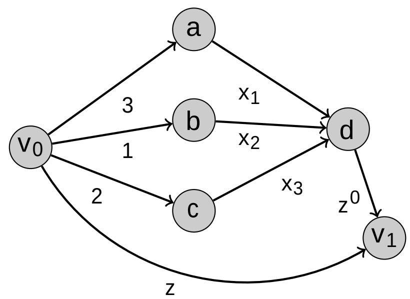
4 Partially Garbling Arithmetic Branching Programs

In this section, we present partial garbling schemes for arithmetic branching programs, with a restriction on where the private inputs are used in the computation. Specifically, we consider functions $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ that are computed by an ABP such that the variables in the private input z appear only on the edges leading into the last vertex v_1 (or more generally, into the last t vertices in V). For instance, this class captures read-once branching programs on $n + n'$ inputs where the first n inputs are public and the last n' inputs are private. Figure 1 shows that this class also captures the first two examples in Section 3.



$$L(x_1,x_2,x_3,z) = \begin{pmatrix} x_1 & 0 & 0 & 0 \\ -1 & x_2 & 0 & 0 \\ 0 & -1 & x_3 & 0 \\ 0 & 0 & -1 & z \end{pmatrix}$$

$$F(x_1,x_2,x_3,z) = x_1x_2x_3z$$



$$F((x_1,x_2,x_3),(z,z')) = z'(3x_1 + x_2 + 2x_3) + z$$

Fig. 1. ABPs for examples in Section 3 with $t = 1$

4.1 Statement of results

We now formally state our results on partial garbling.

Theorem 3 (partially garbling ABP). Consider a function $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ which is computed by an ABP $\Gamma = (V, E, v_0, v_1, \ell)$ taking as input (x, z), where the variables in the private input z appear only on the edges leading into the last t vertices in V. Then, there is a partial garbling scheme $\hat{F}$ of F with the following properties:

- the output length of $\hat{F}$ is $t \cdot (|V| - 1)$;
- each entry of $\hat{F}$ is a polynomial of total degree one in the variables x and z;
- each entry of $\hat{F}$ is a polynomial of total degree one in the randomness for $t = 1$, and total degree two in the randomness in the general case.
- the reconstruction algorithm $\mathsf{Rec}(x, \cdot)$ has degree t in the output of $\hat{F}$.

In the interesting special case where $t = 1$ (which captures the applications to CDS and secret-sharing), we obtain an encoding of linear size and degree one reconstruction. For the case $t = |V| - 1$, we achieve the standard requirement for garbling schemes and recover a variant of the construction in [26, Theorem 1]. We note here that the degree in the randomness r is important in MPC applications where the generation of r is distributed between multiple parties, whereas the degree of the reconstruction algorithm is important in ABE schemes where reconstruction happens *in the exponent*.

Affine partial garbling. Combined with the locality lemma in [3, Lemma 4.17], we obtain an affine PGS for arithmetic branching programs:

Corollary 1 (affine PGS for ABP)**.** *Consider a function $F : \mathbb{F}_q^n \times \mathbb{F}_q^{n'} \to \mathbb{F}_q$ which is computed by an ABP $\Gamma = (V, E, v_0, v_1, \ell)$ taking as input (x, z) where the variables in the private input z appear only on the edges leading into the last t vertices in V. Then, there is an affine partial garbling scheme $\hat{F}$ of F with the following properties:*

- *$\hat{F}$ has the output length $t^2 \cdot |E|$;*
- *the reconstruction algorithm $\mathsf{Rec}(x, \cdot)$ has degree t.*

The locality lemma tells us that we can garble a polynomial of d variables of total degree one using d affine functions of a single variable (e.g. we garble $x_1 + 2x_2 + x_3$ using $(x_1 - r_1, 2x_2 - r_2, x_3 + r_1 + r_2)$), while increasing the randomness complexity by $d - 1$ and without affecting the degree of the reconstruction algorithm. That is, we will replace each polynomial in d variables in $\hat{F}$ with d affine functions in one variable. This increases the output length of $\hat{F}$ from $t \cdot (|V| - 1)$ to $t^2 \cdot |E|$.

4.2 Our construction

Following prior garbling schemes for ABP due to Ishai and Kushilevitz in [25, 26], the starting point of our construction is the matrix representation $L(x, z)$ of the ABP in Lemma 1. Since the variables in z appear only on the edges leading into the last t vertices in V, this means that they only appear in the last t columns of the matrix $L(x, z)$. Garbling proceeds similarly to that in [26, Section 4] by randomizing the last t columns of this matrix while preserving its determinant — we achieve this by multiplying $L(x, z)$ on the left and on the right by random matrices with a prescribed structure. The efficiency improvement over the prior construction comes from using matrices with fewer random entries; in particular, only the last t columns of the randomizing matrices contain random entries.

A digression into matrices. We consider a set $\mathcal{H}$ of matrices which contains $L(x, z)$, along with two groups of matrices $\mathcal{G}_1, \mathcal{G}_2$ (see examples in Figure 2) which would be used to randomize $L(x, z)$ as outlined above.

Definition 2. *Let $\mathcal{H}$ denote the set of $\ell \times \ell$ matrices over $\mathbb{F}_q$ containing only -1's in their second diagonal (the diagonal below the main diagonal), and 0's below the second diagonal. For a fixed parameter t, define two matrix groups $\mathcal{G}_1$ and $\mathcal{G}_2$ as follows:*

- *$\mathcal{G}_1$ is the subset of $\ell \times \ell$ matrices over $\mathbb{F}_q$ with 1's on the main diagonal and 0's in all of the remaining entries except the right-most $t - 1$ entries in the top row;*
- *$\mathcal{G}_2$ is the subset of $\ell \times \ell$ matrices over $\mathbb{F}_q$ with 1's on the main diagonal and 0's in all of the remaining entries except for those above the main diagonal in the t right-most columns.*

It is straight-forward to verify that $\mathcal{G}_1, \mathcal{G}_2$ are both closed under multiplication and inverse; that is, both $\mathcal{G}_1$ and $\mathcal{G}_2$ are subgroups of the multiplicative group of invertible $\ell \times \ell$ matrices. Next, we establish additional properties of $\mathcal{H}, \mathcal{G}_1, \mathcal{G}_2$ which would be used to establish correctness and privacy respectively:

Lemma 2. *For any $H \in \mathcal{H}, G_1 \in \mathcal{G}_1, G_2 \in \mathcal{G}_2$, the first $\ell - t$ columns in $G_1 H G_2$ are the same as those in H.*

Proof. Consider two types of matrix operations:

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & * & * \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \in \mathcal{G}_1, \quad
\begin{pmatrix}
1 & 0 & 0 & * & * & * \\
0 & 1 & 0 & * & * & * \\
0 & 0 & 1 & * & * & * \\
0 & 0 & 0 & 1 & * & * \\
0 & 0 & 0 & 0 & 1 & * \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \in \mathcal{G}_2
$$

Fig. 2. Examples for $\mathcal{G}_1, \mathcal{G}_2$ with $\ell = 6, t = 3$

(a) Add to the top row some linear combination of the bottom $t - 1$ rows, as given by left-multiplication by a matrix from $\mathcal{G}_1$;

(b) Add to the j'th column where $j > \ell - t$ some linear combination of the first $j - 1$ columns, as given by right-multiplication by a matrix from $\mathcal{G}_2$;

Clearly, (b) operations leave the first $\ell - t$ columns in H unchanged. In each of the bottom $t - 1$ rows of H, the first $\ell - t$ entries are all 0's and therefore (a) operations also leave the first $\ell - t$ columns in H unchanged. $\square$

The following lemma (generalizing [26, Lemma 3]) shows that a matrix H from $\mathcal{H}$ can be brought into a canonical form, uniquely defined by its first $\ell - t$ columns and its determinant, by multiplying it from the left by some $G_1 \in \mathcal{G}_1$ and from the right by some $G_2 \in \mathcal{G}_2$.

Lemma 3. For any $H \in \mathcal{H}$, there exists $G_1 \in \mathcal{G}_1$ and $G_2 \in \mathcal{G}_2$ such that $G_1 H G_2$ satisfies the following properties (that is, the "canonical form"):

- the entries in the first $\ell - t$ columns of $G_1 H G_2$ are the same as those in H;
- $G_1 H G_2$ contains -1's in its second diagonal;
- it contains 0's elsewhere except the value $\det(H)$ in the top-right entry.

Note that the canonical form for H is unique and is completely determined by the first $\ell - t$ columns of H and $\det(H)$.

Proof. Again, consider the matrix operations as described in the proof of Lemma 2. As illustrated in Figure 3, a matrix $H \in \mathcal{H}$ can be transformed, using a sequence of (a) and (b) operations, to a matrix H' satisfying the properties in the Lemma. In particular, we modify the top row in the last t columns of H using (a) operations, and the remaining rows in the last t columns of H using (b) operations. Note that matrices in $\mathcal{G}_1$ and $\mathcal{G}_2$ have determinant 1 and therefore $\det(H') = \det(H)$. Finally, a simple calculation says that $\det(H')$ is equal to its top-right entry. $\square$

Partially garbling F. We may now specify our PGS F: start with the $\ell \times \ell$ matrix representation $L(x, z)$ for F, where $\ell = |V| - 1$, and output the last t columns of the matrix $R_1 L(x, z) R_2$, where $R_1 \leftarrow_R \mathcal{G}_1$ and $R_2 \leftarrow_R \mathcal{G}_2$. We proceed to analyze the construction:

$$\begin{pmatrix} * & * & * & * & * & * \\ -1 & * & * & * & * & * \\ 0 & -1 & * & * & * & * \\ 0 & 0 & -1 & * & * & * \\ 0 & 0 & 0 & -1 & * & * \\ 0 & 0 & 0 & 0 & -1 & * \end{pmatrix} \xrightarrow{(a)} \begin{pmatrix} * & * & * & 0 & 0 & * \\ -1 & * & * & * & * & * \\ 0 & -1 & * & * & * & * \\ 0 & 0 & -1 & * & * & * \\ 0 & 0 & 0 & -1 & * & * \\ 0 & 0 & 0 & 0 & -1 & * \end{pmatrix} \xrightarrow{(b)} \begin{pmatrix} * & * & * & 0 & 0 & * \\ -1 & * & * & 0 & 0 & 0 \\ 0 & -1 & * & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

Fig. 3. Bringing a matrix H to canonical form with $\ell = 6, t = 3$

(correctness) Reconstruction proceeds as follows: Given x and the last t columns of $R_1L(x,z)R_2$, we may recover the entire matrix $R_1L(x,z)R_2$, since the first $\ell - t$ columns are the same as those in $L(x,z)$ (cf. Lemma 2) and depend only on x. Then, we compute $\det(R_1L(x,z)R_2)$, which is a degree t computation over the output of F. Correctness follows from the fact that $\det(R_1L(x,z)R_2) = \det(L(x,z)) = F(x,z)$ since $\det(R_1) = \det(R_2) = 1$.

(privacy) Given x and $F(x,z)$, we can compute the canonical form H' of the matrix $L(x,z)$ as defined in Lemma 3, namely $H' = G_1L(x,z)G_2$ for some $G_1 \in \mathcal{G}_1, G_2 \in \mathcal{G}_2$ (we do not need to compute G_1, G_2). Since $\mathcal{G}_1$ and $\mathcal{G}_2$ are both matrix groups, it follows that

$$R_1L(x,z)R_2 \quad \text{and} \quad R_1'H'R_2',$$

where $R_1, R_1' \leftarrow_R \mathcal{G}_1, R_2, R_2' \leftarrow_R \mathcal{G}_2$, are identically distributed. Simulation proceeds by outputting the last t columns of $R_1'H'R_2'$.

Theorem 3 then follows readily.

5 Applications to Secure Computation

5.1 Conditional disclosure of secrets

We describe an application to conditional disclosure of secrets (CDS), already outlined in the introduction, and an implication for secure multiparty computation.

Relation to CDS and secret-sharing. Conditional disclosure of secrets (CDS) [20] allows a set of n players $P_1, \ldots, P_n$ to disclose a secret $z \in F_q$ to an external party Carol, subject to a given condition on their joint inputs. In the original setting, Carol knows all the inputs held by the players except for the secret to be conditionally disclosed, so she knows whether the condition holds and whether she will obtain the secret. Each player on the other hand only sees its portion of the input and does not necessarily know whether Carol will obtain the secret. Following [20], we consider the shared randomness model where all players have access to the same random string which is hidden from Carol. For simplicity, we also assume that all players know z. The protocol involves only a unidirectional communication from the players to Carol. Note that CDS generalizes secret-sharing by considering the special case where each party holds a boolean input and the message sent by P_i corresponds to its share (c.f. [20, Section 3.2.1]).

Now, consider the scenario where party P_i holds $x_i \in \mathbb{F}_q$ and the parties want to disclose the secret z subject to the condition $f(x_1,\ldots,x_n) \neq 0$, where $f : \mathbb{F}_q^n \to \mathbb{F}_q$. Let $F : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q$ denote the function

$$F((x_1,\ldots,x_n),z) = f(x_1,\ldots,x_n) \cdot z$$

and let F be an x-affine PGS for F. We can then derive a CDS protocol as follows:

- the shared random string is the randomness used for F;
- for each $j \in [m]$, if $F(x,z)_j$ is of the form $a_j x_i + b_j$, then party i computes and sends $a_j x_i + b_j$ to Carol.

Suppose $f(x_1,\ldots,x_n) \neq 0$. Then, upon receiving the messages, Carol can then recover $f(x_1,\ldots,x_n) \cdot z$ and thus z using Rec since she knows $x_1,\ldots,x_n$. On the other hand, if $f(x_1,\ldots,x_n) = 0$, then PGS privacy guarantees that Carol learns nothing about the secret. We note that the above construction generalizes non-monotonic secret sharing: player i holds the share $F(x,z)_j = a_j x_i + b_j$. In the case $x_i \in \{0,1\}$, then the 0-share is b_j and the 1-share is $a_j + b_j$.

Improved CDS protocols. First, we obtain the first CDS protocols for arithmetic branching programs where the communication complexity is linear in the branching program size. The prior constructions in [20] achieve linear communication complexity for boolean branching programs, and quadratic complexity for arithmetic branching programs. Furthermore, our protocols only make a black-box use of the underlying field.

Non-zero ABP. Consider the scenario as before where party P_i holds $x_i \in \mathbb{F}_q$ and the parties want to disclose the secret subject to the condition $f(x_1,\ldots,x_n) \neq 0$, where $f : \mathbb{F}_q^n \to \mathbb{F}_q$ is computed by an ABP $\Gamma_f = (V,E,v_0,v_1,\ell)$. It suffices to construct an affine PGS for the function

$$F((x_1,\ldots,x_n),z) = f(x_1,\ldots,x_n) \cdot z$$

Observe that we can build an ABP Γ_F for F from that for f by adding an edge labeled z connecting the original sink v_1 to a new sink (c.f. example on the left in Fig 1). Clearly, Γ_F satisfies the conditions for Corollary 1 with $t = 1$, and the ensuing protocol also achieves communication complexity linear in the size of the ABP for f and linear reconstruction of the secret z.

Zero ABP. We can also handle the condition $f(x_1,\ldots,x_n) = 0$. Here, we want to construct an affine PGS for the function

$$F'((x_1,\ldots,x_n),(z,z')) = f(x_1,\ldots,x_n) \cdot z' + z$$

We can build an ABP $\Gamma_{F'}$ for F' from that for f by adding two edges: one labeled z' connecting the original sink v_1 to the new sink and another labeled z connecting the original source v_0 to the new sink (c.f. example on the right in Fig 1). Again, $\Gamma_{F'}$ satisfies the conditions for Corollary 1 with $t = 1$, and the ensuing protocol also achieves communication complexity linear in the size of the ABP for f and linear reconstruction of the secret z.

Applications to secure multiparty computation. Consider a multiparty protocol in which a client interacts with multiple servers. For instance, the client may wish to make a query q to a database D held by k servers while hiding q from each set of t servers. When the client is semi-honest, this can often be achieved via an efficient 2-round protocol in which the client distributes q between the servers using a suitable secret sharing scheme, and each server responds by applying some local computation to D and its share q_i of q. Such protocols are susceptible attacks by a malicious client, who distributes inconsistent shares q_i between the servers. The traditional approach of preventing such inconsistencies is via the use of interactive protocols for verifiable secret sharing or zero-knowledge proofs. However, these protocols require additional interaction.

An alternative methodology suggested in [20] is to use CDS: make each server send only a single message to the client, such that these messages reveal the messages of the original protocol only under the condition that the client's messages q_i are consistent with the protocol. The information-theoretic CDS constructions given in [20] apply to Boolean formulas and do not efficiently apply in the context of arithmetic computations required for, say, verifying the consistency of the client's messages with Shamir's secret sharing scheme. This can be handled using our CDS protocols above for handling arithmetic predicates.

5.2 Generalized oblivious transfer

In generalized oblivious transfer (GOT) [24] (which includes priced oblivious transfer [1] as a special case), a sender holds n pairs of secret bits $(z_{i,0}, z_{i,1}), i = 1,\ldots,n$ and a receiver holds input $x \in \{0,1\}^n$. In addition, there is some fixed boolean function $f : \{0,1\}^n \to \{0,1\}$. The receiver should learn the secrets indexed by x (namely $z_{1,x_1},\ldots,z_{n,x_n}$) whenever $f(x) = 1$ and nothing otherwise, whereas the sender should learn nothing about x. That is, the receiver learns

$$F_i(x,(z_{i,0},z_{i,1})) = f(x)\cdot(x_i z_{i,1} + (1-x_i)z_{i,0})$$

for $i = 1,\ldots,n$ and $F_i : \{0,1\}^n \times \{0,1\}^2 \to \{0,1\}$.

Suppose f is computed by an ABP $\Gamma_f = (V,E,v_0,v_1,\ell)$ over $\mathbb{F}_2$ of size s. It is easy to see that we can build an ABP Γ_i for F_i with $|V|+3$ vertices and $|E|+4$ edges so that the two edges labeled with the private input $(z_{i,0},z_{i,1})$ both lead into the sink node in Γ_i. This fulfills the requirements for our partial garbling scheme, and by Corollary 1, we can construct a z-affine PGS F_i for F_i whose output length is $O(s)$. This allows us to realize GOT for branching programs using $O(ns)$ parallel invocations of standard bit OT in the semi-honest setting. This improves upon the prior construction in [24] which requires $O(ns^2)$ parallel invocations.

6 Verifiable Computation

We consider the online/offline verifiable computation (VC) where the client wants to delegate the computation of a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ on an input $x \in \mathbb{F}_q^n$, where f is computed by an ABP Γ of size s. For simplicity, we restrict the client's input to be binary, that is, $x \in \{0,1\}^n \subseteq \mathbb{F}_q^n$; we show later how to remove this assumption with an $O(\log q)$ multiplicative overhead. In addition, we may assume that $\log q$ is larger than the statistical security parameter, by taking a sufficiently large field extension of $\mathbb{F}_q$.

The basic construction. Our construction follows the garble+MAC paradigm in [4]. We begin with the function $F : \mathbb{F}_q^n \times \mathbb{F}_q^2 \to \mathbb{F}_q$:

$$F(x,(z_1,z_2)) = z_1 f(x) + z_2$$

That is, we apply the standard pairwise-independent MAC to authenticate the value $f(x)$ under the key (z_1, z_2). Next, we want to partially garble the function F. We observe that for the security proof in [4], it suffices to keep z_1, z_2 private and there is no need to hide x. As with the CDS protocols in Section 5.1, we can apply Corollary 1 to obtain an x-affine PGS $\hat{F}$ for F where $\hat{F}$ has output length $O(s)$. Indeed, if we want to fully garble the ABP for F, the overhead will be quadratic in s (c.f. [26]). Since $\hat{F}$ is x-affine, we may think of $\hat{F}$ as defining n pairs of vectors (a_i, b_i) over $\mathbb{F}_q$ and the output $\hat{F}(x)$ is determined by $(x_i a_i + b_i), i = 1, \ldots, n$. The VC protocol for delegating the computation of f on x then proceeds as follows:

- In the offline phase, the client picks $z_1, z_2 \leftarrow_R \mathbb{F}_q$ and computes the n vectors (a_i, b_i). In addition, it selects n pairs of seeds $(\sigma_{i,0}, \sigma_{i,1})$ for a PRG G, and sends the server n pairs of values

$$G(\sigma_{i,0}) \oplus b_i, G(\sigma_{i,1}) \oplus (a_i + b_i), \ i = 1, \ldots, n$$

  The client only needs to store z_1, z_2 and the n pairs of seeds for the next phase.

- In the online phase, upon receiving $x \in \{0,1\}^n$, the client sends the server x, along with the n seeds

$$\sigma_{i,x_i}, \ i = 1, \ldots, n$$

  The server can then compute $x_i a_i + b_i$ and thus both $\hat{F}(x, (z_1, z_2))$ and $F(x, (z_1, z_2))$. It sends $(y, \tau) = (f(x), F(x, (z_1, z_2)))$ back to the client. The client accepts if $\tau = z_1 y + z_2$ and rejects otherwise.

The intuition for soundness is as follows. Fix an ABP f and an input $x \in \{0,1\}^n$ and we want to bound the probability that a cheating server S^* convinces an honest client to accept $y \neq f(x)$. First, by the security of the PRG, at the end of the online phase, S^* only learns $\hat{F}(x, (z_1, z_2))$ and nothing else about a_i, b_i. Then, by the privacy of PGS, S^* only learns $z_1 f(x) + z_2$ and nothing else about (z_1, z_2). This means that by the one-time security of the pairwise-independent MAC, the client accepts a tag on $y \neq f(x)$ with probability at most $1/q$. Following [4, Lemma 1], we can indeed show that the ensuing VC protocol has soundness error at most $1/q + \text{negl}(\kappa)$.

Removing the binary restriction. It is easy to see that we can transform any $f : \mathbb{F}_q^n \to \mathbb{F}_q$ into another function $f' : \mathbb{F}_q^{n\lceil \log q \rceil} \to \mathbb{F}_q$ so that:

- for all $x \in \mathbb{F}_q^n$, $f'(\bar{x}) = f(x)$, where $\bar{x} \in \{0,1\}^{n\lceil \log q \rceil}$ denote the binary decomposition of x;
- if f is computed by an ABP Γ of size s, then f' is computed by an ABP Γ' of size $O(s \log q)$, obtained by replacing each edge in Γ with $O(\log q)$ edges which recovers a variable from its binary decomposition.

Now, we just apply the preceding construction to the ABP f', with $\bar{x}$ as the client's input in the online phase.

Handling boolean formula. We note that our construction may be applied to boolean formula, by converting a formula of size s to an ABP of size $O(s)$ over a field of size $q \approx 2^{\lambda}$ where λ is the statistical security parameter. We then obtain a VC protocol where the offline complexity and the client's online complexity are dominated by $O(s\lambda)$ and $O(n\lambda)$ respectively, improving upon that based on Yao's garbled circuits in [19, 4] in that we replace the dependency on a computational security parameter with that on a statistical security parameter. Here, instead of our partial garbling scheme from Corollary 1, we may also use a direct construction based on secret-sharing for boolean formula implicit in [32, 23].

7 Attribute-Based Encryption for ABPs

In this section, we present our ABE schemes for ABPs. Here, ciphertexts are associated with $x \in \mathbb{F}_q^n$, and secret keys are associated with an ABP $f : \mathbb{F}_q^n \to \mathbb{F}_q$. In Z-ABP, decryption is possible iff $f(x) = 0$, and in N-ABP, decryption is possible iff $f(x) \neq 0$. Roughly speaking, instead of using secret-sharing schemes as in the literature on ABE for boolean formula [23, 30], we will use CDS for ABP given in Section 5.1, which support linear reconstruction. The latter is important since we need to reconstruct the secret α in the exponent. As with ABE for boolean formula, it is more convenient to work with span programs [28] for the proof of security; a side-benefit is that we can simultaneously capture both Z-ABP and N-ABP.

7.1 Arithmetic span programs

We define arithmetic span programs, a generalization of (boolean) span programs [28].

Definition 3 (arithmetic span program [28]). An arithmetic span program $(\mathbf{V}, \rho)$ is a collection of vectors $\mathbf{V} = \{(\mathbf{y}_j, \mathbf{z}_j) : j \in [m]\}$ in $\mathbb{F}_q^\ell$ and $\rho : [m] \to [n]$. We say that

$$x \in \mathbb{F}_q^n \text{ satisfies } (\mathbf{V}, \rho) \text{ iff } \mathbf{e}_\ell \in \text{span}\langle x_{\rho(j)}\mathbf{y}_j + \mathbf{z}_j \rangle,$$

where $\mathbf{e}_\ell := (0, 0, \ldots, 0, 1) \in \mathbb{F}_q^\ell$ and span refers to linear span of a collection of column vectors.

We relate Z-ABP and N-ABP to arithmetic span programs. That is, given an ABP Γ for $f : \mathbb{F}_q^n \to \mathbb{F}_q$ on ℓ vertices, we construct an arithmetic span program $(\mathbf{V}, \rho)$ so that

- for Z-ABP: $x \in \mathbb{F}_q^n$ satisfies $(\mathbf{V}, \rho)$ iff $f(x) = 0$;
- for N-ABP: $x \in \mathbb{F}_q^n$ satisfies $(\mathbf{V}, \rho)$ iff $f(x) \neq 0$.

In both cases, we start with an ABP Γ' for f on $\ell + 1$ vertices, obtained by adding an edge labeled 1 connecting the original sink in Γ to a new sink in Γ' (an analogous construction appears in Section 5.1). Next, we apply the transformation in Lemma 1 to Γ' to obtain a $\ell \times \ell$ matrix $\mathbf{L}$ such that $\det(\mathbf{L}(x)) = f(x)$, where $\mathbf{L}(x)$ has the following form:

$$\begin{pmatrix} * & * & * & \cdots & * & * & 0 \\ -1 & * & * & \cdots & * & * & 0 \\ 0 & -1 & * & \cdots & * & * & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & * & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix}$$

Observe that the first $\ell - 1$ columns of $\mathbf{L}(x)$ are linearly independent. We will actually need to first modify Γ (by replacing each edge e with a pair of edges labeled $\sigma(e)$ and 1) so that every vertex has at most one incoming edge of degree 1, so that each column of $\mathbf{L}$ depend on the same variable and can be written as $x_{\rho(j)}\mathbf{y}_j + \mathbf{z}_j$. We may then derive the arithmetic span program $(\mathbf{V}, \rho)$ as follows:

for Z-ABP: Here,

$$f(x) = \det(L(x)) = 0 \iff e_\ell \text{ lies in the column span of the first } \ell - 1 \text{ columns of } L(x)$$

We then take $(V, \mathbf{‰})$ to describe the column span of the first $\ell - 1$ columns of $L(x)$.

for N-ABP: Here,

$$f(x) = \det(L(x)) \neq 0 \iff e_1 \text{ lies in the column span of } L(x)$$

We take $(V, \mathbf{‰})$ to describe the column span of $L(x)$, with the first and ℓ'th row switched.

7.2 Computational assumptions

We now briefly recall bilinear pairing groups and then state the decisional bilinear Diffie-Hellman (DBDH) assumption that are required in our security proof. A generator $\mathcal{G}$ which takes as input a security parameter 1^λ and outputs a description $\mathbb{G} := (p, G, G_T, e)$, where p is a prime of $\Theta(\lambda)$ bits, G and G_T are cyclic groups of order p, and $e : G \times G \to G_T$ is a non-degenerate bilinear map. We require that the group operations in G and G_T as well the bilinear map e are computable in deterministic polynomial time with respect to λ. Furthermore, the group descriptions of G and G_T include generators of the respective cyclic groups.

Assumption 1 (DBDH: Decisional Bilinear Diffie-Hellman Assumption) Given a group generator $\mathcal{G}(1^\lambda)$, we define the following distribution:

$$\begin{aligned}
&\mathbb{G} := (p, G, G_T, g, e) \leftarrow_R \mathcal{G}(1^\lambda),\\
&a, b, s, z \leftarrow_R \mathbb{Z}_q,\\
&T_0 := g^{abs},\ T_1 := g^{abs+z},\\
&D := (\mathbb{G}; g^a, g^b, g^s).
\end{aligned}$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$\mathsf{Adv}^{\mathrm{DBDH}}_{\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|$$

is negligible in the security parameter λ.

7.3 Attribute-Based Encryption

We present the definitions for attribute-based encryption [34, 23].

Syntax. A attribute-based encryption (ABE) scheme for arithmetic span programs with message space $\mathcal{M}$ consists of four algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda, 1^n) \to (\quad , \quad)$. The setup algorithm gets as input the security parameter λ, the length parameter 1^n and outputs the public parameter , and the master key .

- $\mathsf{Enc}(\quad , x, m) \to {}_x$. The encryption algorithm gets as input , an attribute $x \in \mathcal{X}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext ${}_x$.

KeyGen(, $(V,\rho)) \to {}_{V,\rho}$. The key generation algorithm gets as input and an arithmetic span program (V,ρ). It outputs a secret key $_{V,\rho}$.

Dec($_{V,\rho}$, $_{x}$) $\to$ m. The decryption algorithm gets as input $_{V,\rho}$ and $_{x}$ and outputs a message m.

Correctness. If x satisfies (V,ρ), for all messages $m \in M$, we have

$$\Pr[\mathsf{Dec}({}_{V,\rho}, \mathsf{Enc}(, x, m)) = m] = 1.$$

where the probability is taken over (,) $\leftarrow$ Setup$(1^{\lambda}, 1^n)$; $_{y} \leftarrow$ KeyGen(, (V,ρ)) and the coins of all the algorithms in the expression above.

7.4 Security Model

We consider the following selective CPA security game for attribute-based encryption played between a challenger and an adversary A.

Selective. The adversary A submits the challenge attribute x^*.

Setup. The challenger generates (,) $\leftarrow$ Setup$(1^{\lambda}, 1^n)$ and gives to the adversary A.

Phase 1. The adversary A adaptively requests keys for any (V,ρ) not satisfied by x^*. The challenger responds with the corresponding secret key $_{V,\rho} \leftarrow$ KeyGen(, (V,ρ)).

Challenge. The adversary A submits two messages m_0, m_1 of equal length. The challenger picks $\beta \leftarrow_R \{0,1\}$ and sends A the ciphertext $^* \leftarrow$ Enc(, x, m_β).

Phase 2. A continues to request keys for any (V,ρ) of its choice, subject to the same constraints as before.

Guess. The adversary A outputs a guess β' for β.

We define the advantage function $\mathsf{Adv}_A^{\mathsf{ABE}}(,)$ to be $|\Pr[\beta' = \beta] - 1/2|$.

7.5 Construction

We present our ABE scheme for arithmetic span programs:

Setup$(1^{\lambda}, 1^n)$: generate $G := (q, G, G_T, e) \leftarrow_R \mathcal{G}(1^{\lambda})$, pick $\alpha \leftarrow_R \mathbb{Z}_q, a \leftarrow_R \mathbb{Z}_q^*, v \leftarrow_R \mathbb{Z}_q^n$, and output

$$:= \left(G, e(g,g)^{\alpha}, g, g^{a}, g^{v} \right) \quad \text{and} \quad := (\alpha, a, v)$$

Enc(, x, m): pick $s \leftarrow_R \mathbb{Z}_q$ and output

$$_{x} := \left(C_0 := g^{s}, C_i := g^{(ax_i - v_i)s} : i \in [n], C' := e(g,g)^{\alpha s} \cdot m \right)$$

KeyGen(, , (V,ρ)): pick $u \leftarrow_R \mathbb{Z}_q^{\ell}$ subject to the constraint $\langle u, e_1 \rangle = \alpha$, and compute

$$\beta_j := \langle u, y_j \rangle \quad \text{and} \quad \gamma_j := \langle u, z_j \rangle.$$

Output

$$\mathsf{sk}_{V,\rho} := \left((K_{j,0}, K_{j,1}) := (g^{\frac{\beta_j}{a}}, g^{\gamma_j - \frac{\beta_j v_{\rho(j)}}{a}}) : j \in [m] \right).$$

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{V,\rho}, \mathsf{ct}_x)$: If $e_1 \in \mathrm{span}\langle x_{\rho(j)} y_j + z_j \rangle$, first compute $\omega_1, \ldots, \omega_m \in \mathbb{Z}_q$ such that

$$e_1 = \sum_{j=1}^{n} \omega_j (x_{\rho(j)} y_j + z_j)$$

Parse the ciphertext as $(C_0, C_1, \ldots, C_n, C')$ and output

$$m \leftarrow C' \cdot \Big(\prod_{j=1}^{n} \big(e(C_0, K_{j,1}) \cdot e(C_{\rho(j)}, K_{j,0}) \big)^{\omega_j} \Big)^{-1}.$$

Theorem 4. Under DBDH assumption (described in Section 7.2), our ABE scheme defined in Section 7.5 is selectively secure (in the sense of Definition 7.4).

Correctness. Observe that

$$e(C_0, K_{j,1}) \cdot e(C_{\rho(j)}, K_{j,0}) = e(g,g)^{s(x_{\rho(j)} \beta_j + \gamma_j)}.$$

Correctness then follows readily from the fact that

$$\alpha = \sum_{j=1}^{n} \omega_j (x_{\rho(j)} \beta_j + \gamma_j).$$

7.6 Proof of ABE Security

We prove the following lemma:

Lemma 4. For any adversary $\mathcal{A}$ against the ABE scheme, there exist an adversary $\mathcal{B}$ against the DBDH assumption whose running time is roughly that of $\mathcal{A}$ such that

$$\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{B}}(\lambda) - 1/p.$$

Proof. Recall that $\mathcal{B}$ is given

$$D := (\mathbb{G}; g, g^a, g^b, g^s),$$

along with T, where T equals $e(g,g)^{abs}$ or is drawn uniformly from G_T. Here, we assume that $a \leftarrow_R \mathbb{Z}_q^*$, which yields a $1/p$ negligible difference from DBDH assumption in the advantage. $\mathcal{B}$ proceeds as follows:

Setup. On input selective challenge x^*, pick $v \leftarrow_R \mathbb{Z}_q^n$ and implicitly set

$$\alpha := ab \quad \text{and} \quad v := v - a x^*.$$

Observe that we can compute mpk given g, g^a, g^b as follows:

$$\mathsf{mpk} := (\mathbb{G}; e(g^a, g^b), g, g^a, g^{v} (g^a)^{-x^*}).$$

Key Queries. On input a key query (V, ρ) not satisfied by x^*, by duality, we can efficiently compute u^* such that

$$\langle u^*, e_1 \rangle = 1 \quad \text{and} \quad \langle u^*, x^*_{\rho(j)} y_j + z_j \rangle = 0, \ \forall j \in [m].$$

In addition, we pick $u \leftarrow_R \mathbb{Z}_q^\ell$ subject to the constraint $\langle u, e_\ell \rangle = 0$ and implicitly set

$$u := abu^* - au.$$

It suffices to show how to simulate

$$(K_{j,0}, K_{j,1}) = \left(g^{\frac{\beta_j}{a}}, g^{x^*_{\pi(j)}\beta_j - \gamma_j + \frac{\beta_j v_{\pi(j)}}{a}}\right)$$

for each j, where

$$\beta_j := ab\langle u^*, y_j\rangle - a\langle u, y_j\rangle, \qquad \gamma_j := ab\langle u^*, z_j\rangle - a\langle u, z_j\rangle$$

Now, observe that we can readily simulate $K_{j,0}$ given g^b, since

$$K_{j,0} = g^{b\langle u^*, y_j\rangle - \langle u, y_j\rangle}$$

Next, observe that

$$x^*_{\pi(j)}\beta_j - \gamma_j = ab\langle u^*, x^*_{\pi(j)}y_j - z_j\rangle - a\langle u, x^*_{\pi(j)}y_j - z_j\rangle = a\langle u, x^*_{\pi(j)}y_j - z_j\rangle$$

since $\langle u^*, x^*_{\pi(j)}y_j - z_j\rangle = 0$. We can then simulate $K_{j,1}$ given $g^a, v_{\pi(j)}$, since

$$K_{j,1} = (g^a)^{\langle u, x^*_{\pi(j)}y_j - z_j\rangle} \cdot K_{j,0}^{v_{\pi(j)}}$$

Challenge Ciphertext. Upon receiving m_0 and m_1 from $\mathcal{A}$, $\mathcal{B}$ picks $\beta \leftarrow_R \{0,1\}$ and outputs the challenge ciphertext as:

$$\mathsf{ct}_{x^*} := \left(g^s, (g^s)^v, T \cdot m_\beta\right).$$

Now, if T equals $e(g,g)^{abs}$, then ct_{x^*} is indeed an encryption of m_β. On the other hand, if $T \leftarrow_R G_T$, then ct_{x^*} is an encryption of a random message in G_T and thus independent of β.

Guess. When $\mathcal{A}$ halts with output β', $\mathcal{B}$ outputs 1 if $\beta = \beta'$ and 0 otherwise.

We may therefore conclude that $\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda) \geq \mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{B}}(\lambda) - 1/p$. ∎

References

[1] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In EUROCRYPT, pages 119–135, 2001.

[2] B. Applebaum. Randomly encoding functions: A new cryptographic paradigm - (invited talk). In ICITS, pages 25–31, 2011.

[3] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. SIAM J. Comput., 36(4):845–888, 2006.

[4] B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In ICALP (1), pages 152–163, 2010.

[5] B. Applebaum, Y. Ishai, and E. Kushilevitz. How to garble arithmetic circuits. In FOCS, pages 120–129, 2011.

[6] N. Attrapadung and B. Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In Public Key Cryptography, pages 384–402, 2010.

[7] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In ACM CCS, 2012. Also Cryptology ePrint Archive, Report 2012/265.

[8] F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Public Key Cryptography, pages 272–291, 2013.

[9] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In CRYPTO, page 110, 2011. Cryptology ePrint Archive, Report 2011/132.

[10] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In CRYPTO (1), pages 449–475, 2013.

[11] N. Bitansky and O. Paneth. Point obfuscation and 3-round zero-knowledge. In TCC, pages 190–208, 2012.

[12] O. Blazy, D. Pointcheval, and D. Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In TCC, pages 94–111, 2012.
[13] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In EUROCRYPT, pages 223–238, 2004.
[14] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In EUROCRYPT, 2014. To appear.
[15] N. Chandran, V. Goyal, R. Ostrovsky, and A. Sahai. Covert multi-party computation. In FOCS, pages 238–248, 2007.
[16] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In STOC, pages 554–563, 1994.
[17] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In CRYPTO (2), pages 479–499, 2013. Also, Cryptology ePrint Archive, Report 2013/128.
[18] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In STOC, pages 467–476, 2013. Also, Cryptology ePrint Archive, Report 2013/258.
[19] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In CRYPTO, pages 465–482, 2010.
[20] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. J. Comput. Syst. Sci., 60(3):592–629, 2000.
[21] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In STOC, pages 113–122, 2008.
[22] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In STOC, pages 545–554, 2013. Also, Cryptology ePrint Archive, Report 2013/337.
[23] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In ACM Conference on Computer and Communications Security, pages 89–98, 2006.
[24] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In ISTCS, pages 174–184, 1997.
[25] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In FOCS, pages 294–304, 2000.
[26] Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In ICALP, pages 244–256, 2002.
[27] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structure. In GLOBECOM, pages 99–102, 1987.
[28] M. Karchmer and A. Wigderson. On span programs. In Structure in Complexity Theory Conference, pages 102–111, 1993.
[29] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In EUROCRYPT, pages 146–162, 2008.
[30] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In EUROCRYPT, pages 62–91, 2010.
[31] A. B. Lewko, A. Sahai, and B. Waters. Revocation systems with very small private keys. In IEEE Symposium on Security and Privacy, pages 273–285, 2010.
[32] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In TCC, pages 422–439, 2012.
[33] M. Prabhakaran and A. Sahai. Secure Multi-Party Computation. IOS Press, 2003.
[34] A. Sahai and B. Waters. Fuzzy identity-based encryption. In EUROCRYPT, pages 457–473, 2005.
[35] A. Shamir. How to share a secret. Commun. ACM, 22(11):612–613, 1979.
[36] T. Tassa. Generalized oblivious transfer by secret sharing. Des. Codes Cryptography, 58(1):11–21, 2011.
[37] L. von Ahn, N. J. Hopper, and J. Langford. Covert two-party computation. In STOC, pages 513–522, 2005.
[38] A. C.-C. Yao. Theory and applications of trapdoor functions. In FOCS, pages 80–91, 1982.