Arithmetic Garbling from Bilinear Maps

Nils Fleischhacker[1], Giulio Malavolta[2], and Dominique Schröder[2]

[1] Ruhr University Bochum
[2] Friedrich Alexander Universität Erlangen-Nürnberg

Abstract. We consider the problem of garbling arithmetic circuits and present a garbling scheme for inner-product predicates over exponentially large fields. Our construction stems from a generic transformation from predicate encryption which makes only blackbox calls to the underlying primitive. The resulting garbling scheme has practical efficiency and can be used as a garbling gadget to securely compute common arithmetic subroutines. We also show that inner-product predicates are complete by generically bootstrapping our construction to arithmetic garbling for polynomial-size circuits, albeit with a loss of concrete efficiency.
In the process of instantiating our construction we propose two new predicate encryption schemes, which might be of independent interest. More specifically, we construct (i) the first pairing-free (weakly) attribute-hiding non-zero inner-product predicate encryption scheme, and (ii) a key-homomorphic encryption scheme for linear functions from bilinear maps. Both schemes feature constant-size keys and practical efficiency.

1 Introduction

Garbled circuits were introduced by Yao in an oral presentation about secure function evaluation [42]. Given a function f and an input x, Yao's machinery generates a garbled circuit $\tilde{f}$ and an encoded input $\tilde{x}$. Anyone holding $\tilde{x}$ can then evaluate the garbled circuit $\tilde{f}$ to recover the output $f(x)$ and nothing beyond that. Garbled circuits have been cast in the more generic framework of randomized encodings [29] and were first recognized as a cryptographic primitives on its own in the work of Bellare, Hoang, and Rogaway [11]. Since their introduction, garbled circuits have found an enormous range of application to problems such as computation over encrypted data [18], parallel cryptography [5], functional encryption [39], and many others.

Yao's classical construction assumes a binary encoding of the inputs and operates over boolean gates. Since arithmetic computation appear often in real-life scenarios, a natural question to ask is whether one can garble arithmetic circuits over fields of exponential size. This is motivated by efficiency constraints, as operating directly on the arithmetic representation of the input avoids the costly bit decomposition step. Furthermore, certain cryptographic models [4] only admit access to inputs as atomic ring elements. Consider an input encrypted under a linearly homomorphic encryption scheme [22], in this case there is no efficient algorithm to decompose it to its binary representation and the only admissible operations are field addition and scalar multiplication.

The first milestone in this regard was set by Applebaum, Ishai, and Kushilevitz [6] when they proposed the first non-trivial garbling scheme for arithmetic computations over the integers. The scheme is based on the hardness of the learning with errors (LWE) problem [38]. They also show several information-theoretic garbling gadgets for arithmetic branching programs (or, equivalently, arithmetic formulae). The price to pay for perfect security is a quadratic blowup in the input encodings, i.e., to evaluate a circuit of size s one needs $O(s^2)$ space to encode its inputs. In this sense the former construction has better asymptotics.

1.1 Our Results

In this work we continue the study of arithmetic garbling and we propose a simple construction from number-theoretic assumptions. Specifically, we construct a garbling scheme for inner-product predicates over exponentially large fields, assuming the hardness of standard problems in bilinear groups. We shall note that, in general, input vectors are linear in the size of the predicate being computed so the cost of evaluating the

encoding function might be comparable to evaluating the function itself. There are however two important differences that make the problem of garbling inner-products non-trivial: (i) Part of the input vector might be fixed in advance, in which case some steps of the encoding circuit can be precomputed, and (ii) the encoding function does not depend on the garbled function. This means that the garbling algorithm and the encoding algorithm can be executed non-interactively by two different parties.

Our construction (Section 3) consists of a compiler that transforms a predicate encryption scheme into an arithmetic garbling scheme for inner products. The garbling algorithm makes only blackbox use of the underlying primitives and thus has practical efficiency. In contrast with information-theoretic garbling schemes, the size of the encoding depends exclusively on the decryption keys of the predicate encryption. To instantiate our construction we propose a zero (ZIPE) and a non-zero (NIPE) inner-product predicate encryption scheme from standard assumptions, more specifically:

- A NIPE scheme from DDH or LWE with constant-size keys (Section 4).
- A ZIPE scheme with small keys (2 group elements) and with ciphertexts consisting of exactly n group elements (where n is the length of the vectors) from standard assumptions in bilinear groups (Section 5).

Combined with the DDH-based NIPE, the input encodings of our garbling scheme add only two group elements and two integers (in $\mathbb{Z}_p$) to the size of the original vector, regardless of the vector length. This is asymptotically optimal and it is comparable with the work of Applebaum et al. [7], where they construct short encodings for classical boolean garbling. Finally we show that arithmetic garbling for inner-product predicates is complete in the sense that it can be generically bootstrapped to the evaluation of polynomial-size circuit (Section 6), albeit with a loss in efficiency.

1.2 Applications

Our garbling scheme for inner-products is motivated by practical scenarios where one needs to efficiently garble simple arithmetic circuits. Such a garbling gadget can also be used as a building block within a larger protocol to boost the efficiency of certain arithmetic subroutines, while retaining all of the advantages of garbling schemes (e.g., low latency). Here we highlight some tasks of interest which can be solved with a garbling scheme for inner-product predicates.

FHE Decryption. Most recent fully-homomorphic encryption schemes [16, 17, 25] feature a decryption algorithm of the form

$$\langle \boldsymbol{sk}, \boldsymbol{c} \rangle \in \left[\frac{p}{2} - N, \frac{p}{2} + N\right] \pmod p$$

where $\boldsymbol{sk}$ and $\boldsymbol{c}$ are vectors in $\mathbb{Z}_p^n$, and N is a polynomially large noise range. The decryption algorithm returns 1 if the above condition is satisfied and 0 otherwise. This relation can be easily encoded as an inner-product predicate by the so called lazy-OR trick [27]

$$\exists \eta \in \left[\frac{p}{2} - N, \frac{p}{2} + N\right] : \langle \boldsymbol{sk}, \boldsymbol{c} \rangle - \eta \equiv 0 \pmod p.$$

It is clear that whenever $\boldsymbol{c}$ encrypts a 0 the garbling scheme returns 0 in all position, whereas if $\boldsymbol{c}$ is an encryption of 1 then the garbling gadget will return 0 everywhere except in exactly one position where it returns 1. To hide the exact position of the 1 – as this would leak the value of η – we can randomly permute the order in which the inner-product relation is tested against each η.

Private Set Intersection Predicate. A private set intersection predicate refers to the problem of securely computing whether two vectors $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z}_p^{2n}$ have a common element, i.e., whether there exists an i such that $x_i = y_i$. A standard approach to perform this task is to encode y as an n degree polynomial $p_{\boldsymbol{y}}$ with roots in $(y_1, \ldots, y_n)$ and evaluate

$$\exists i \in \{1, \ldots, n\} : p_{\boldsymbol{y}}(x_i) \equiv 0 \pmod p.$$

If we consider the vector of coefficients of $p_{\boldsymbol{y}}$ and the n vectors $(x_i, x_i^2, \ldots, x_i^n)$, then this reduces to the computation of inner-product predicates, which we can efficiently garble.

Matrix Multiplication. It is easy to see that the multiplication of an $n \times n$ matrix with an n size vector reduces to n instances of inner-products. It follows that matrix multiplication predicates (and in general secure linear algebra computations) can be generically reduced to the task of computing inner-products. Our scheme can garble predicates of the form $A \cdot B = C$, where $A \in \mathbb{Z}_p^{n \times m}$, $B \in \mathbb{Z}_p^{m \times n}$, and C is a publicly known $\mathbb{Z}_p^{n \times n}$ matrix.

Statistics on Private Data. Our scheme can also be used to garble several interesting measures over a private dataset x. Among others, we can securely check whether the weighted mean x is equal to a certain known value μ by garbling the vector $(x\| - 1)$ and issuing encoding for the vector $\left(\frac{w_1}{n}, \ldots, \frac{w_n}{n}, \mu\right)$. Then we have

$$\left\langle (x\| - 1), \left(\frac{w_1}{n}, \ldots, \frac{w_n}{n}, \mu\right) \right\rangle = \frac{\sum_{i=1}^{n} w_i x_i}{n} - \mu = 0,$$

where $(w_1, \ldots, w_n)$ is a vector of weights. As a further example, we can garble the square Euclidean distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ by expanding the equation

$$d_{\boldsymbol{x},\boldsymbol{y}} = (x_1 - y_1)^2 + \ldots + (x_n - y_n)^2$$

and encoding the mixed terms as inner-products.

1.3 Concrete Efficiency

In the following we discuss the concrete efficiency of our garbling scheme for inner-product predicates, when instantiated with our ZIPE and the DDH-based NIPE. We measure the computational efficiency of our main algorithms in terms of number of modular exponentiations and pairings and we omit routine calculations such as additions and multiplications. The costs for an inner-product predicate over $\mathbb{F}^n$ are detailed below.

- *Garbling*: The cost of the garbling algorithm is dominated by a call of the setup and encryption algorithm of the NIPE and ZIPE scheme, for a total of $(5n + 11)$ exponentiations. The size of the garbled circuit is $(n + 2)$ elements of the source group $\mathbb{G}_1$, an element of the target group $\mathbb{G}_T$, and $(n + 3)$ elements of a DDH-hard group $\mathbb{G}$.
- *Encoding*: The encoding algorithm runs in time comparable to that of $(n + 2)$ exponentiations. The size of the encoded input is two element of source group $\mathbb{G}_2$ and two integers in $\mathbb{Z}_p$.
- *Evaluation*: The circuit evaluation consists of one call each to the decryption algorithms of the NIPE and ZIPE scheme. This accounts for $(2n + 4)$ exponentiations and 2 pairings. Note that the number of pairings (the most expensive operation) is independent of the vector length.

When compared with information-theoretic constructions, our scheme brings down the size of the encodings from quadratic to independent of the circuit size. This is an asymptotic improvement. The efficiency comparison with the construction of [6] is less clear since it is lattice-based. A concrete comparison would require an implementation of their scheme, which is beyond the scope of this work. Regardless, we believe that broadening the set of assumptions that suffice to construct arithmetic garbling with non-trivial efficiency is an interesting goal on its own. Finally we remark that there exist tailor-made protocols to solve each individual task, among the applications that we suggested above, which are highly optimized for the goal and in practice outperform our solution. However our construction gives a unified and generic approach to securely compute a large family of functions and inherits all the distinguishing features of garbling schemes, such as the low round complexity.

1.4 Our Techniques

In this section we outline the main ideas behind our work and we give an overview of the techniques developed throughout the paper.

Arithmetic Garbling from Predicate Encryption. Our initial observation is an interesting connection between garbling schemes and predicate encryption for inner-products. A predicate encryption scheme allows one to encrypt a message m under a vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ and issue decryption keys for vectors $\boldsymbol{y} \in \mathbb{Z}_p^n$. The decrypter can recover m if and only if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. Furthermore, no information about $\boldsymbol{x}$ is leaked beyond the fact that it is orthogonal to $\boldsymbol{y}$. Given such a predicate encryption, we can construct an arithmetic garbling scheme for inner-product predicates in a very natural way: Garbling a vector $\boldsymbol{x}$ corresponds to encrypting a fixed message m^* under $\boldsymbol{x}$, whereas encoding an input $\boldsymbol{y}$ consists of generating a key for $\boldsymbol{y}$. The evaluator can test whether the decryption algorithm returns m^* and learn whether $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. Since the predicate encryption is attribute-hiding, this does not reveal any further information about $\boldsymbol{x}$.

However a garbling scheme must satisfy some additional properties. Among others, the scheme must guarantee the authenticity of its output, i.e., the evaluator can efficiently show a proof of the result of the computation. This property has been proved useful in the context of zero-knowledge protocols [30] and verifiable computation [24]. A naive approach would be to substitute m^* with a random message r^* which constitutes a valid proof that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, as otherwise the decryption would have failed. However for the complementary case ($\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$) the evaluator is left with nothing. Standard techniques, such as garbling the complement of $\boldsymbol{x}$, do not seem to apply here since we would need to find a vector $\bar{\boldsymbol{x}}$ orthogonal to any $\boldsymbol{y}$ such that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$, which may not exist.

We resolve this issue by shifting the complement to the cryptographic primitive: We additionally deploy a non-zero inner-product predicate encryption scheme, which returns the encrypted message if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$. Given such a scheme, the solution consists in encrypting another random message under the same $\boldsymbol{x}$, which can be used to certify that the two vectors are not orthogonal. Since our garbling algorithm makes only blackbox use of the underlying primitives, the resulting instantiations are simple and practical.

Arithmetic Garbling for Circuits. The connection between garbling and predicate encryption is not limited to inner-products predicates but generalizes to polynomial-size circuits using universal encodings [41]. However, current instantiations of attribute-hiding predicate encryption [23] for circuits are not yet in the domain of practicality and are based on newly crafted assumptions.

Fortunately inner-product predicates are a very powerful tool and we show that our garbling scheme for inner-product predicates can be generically bootstrapped into an arithmetic garbling scheme for poly-size circuit with small (poly-size) domain. If we assume a relaxed model on the arithmetic representation of the inputs, then we can leverage standard techniques to extend our scheme to an exponentially large input domain. The remainder of this work focuses on constructing efficient inner-product encryption schemes to instantiate our arithmetic garbling scheme. Our schemes aim at minimizing the size of the decryption keys, as their size determines the size of the input encodings.

Non-Zero Inner-Product Encryption. We propose an efficient predicate encryption scheme for non-zero inner-products with small keys. The construction consists of a generic transformation from inner-product functional encryption, a primitive introduced by Abdalla et al. [1]. Our transformation makes only two calls to the encryption (resp. decryption) algorithm of the inner-product functional encryption scheme and preserves all the properties of the underlying construction. This transformation yields:

- the first pairing-free (weakly) attribute-hiding predicate encryption for inner-products,
- the first (weakly) attribute-hiding scheme based on the DDH assumption, and
- the first adaptively payload hiding scheme with a tight security reduction.

In the DDH-based instance, a key for a vector of any length corresponds to the vector itself and 2 elements of $\mathbb{Z}_p$.

Zero Inner-Product Encryption. To construct a zero inner-product predicate encryption scheme, we start from the same design paradigm as the lattice-based scheme of Boneh et al. [15]. In [15] the authors propose a fully key-homomorphic public-key encryption scheme. Such a primitive allows anyone to transform an encryption under attribute $\boldsymbol{x}$ into an encryption under $(f(\boldsymbol{x}), [f])$, where $f(\boldsymbol{x}) \in \mathbb{F}$ and $[f]$ is an encoding of the circuit computing f. Predicate encryption can then be instantiated in a very natural way: On input

a predicate f, one can issue a key for $(1, [f])$ and the decryptor can publicly apply the transformation from above to any ciphertext and decrypt if and only if $f(\boldsymbol{x}) = 1$. The advantage of this class of schemes is that the complex operations are pushed to public algorithms and therefore the resulting decryption keys are typically small.

We apply the same idea to the bilinear maps settings: Our scheme can be seen as a key-homomorphic encryption for *linear functions*. The private key of a vector consists of two group elements, regardless of the vector length. For a vector of length n, a ciphertext consists of exactly n group elements. In contrast, previous solutions with similar key sizes (such as [10,19,35]), require at least $2n$ elements. A crucial difference with respect to the lattice-based scheme [15] is that our scheme achieves a (weak) notion of attribute hiding. The resulting scheme is only selectively secure, which is however enough for our purposes.

1.5 Related Work

The first formal treatment of arithmetic garbling is due to Applebaum, Ishai, and Kushilevitz [6]. They proposed (i) a construction for any circuit based on lattices and (ii) an information-theoretic construction for branching programs (or, equivalently, NC1 circuits). A shortcoming of the latter scheme is that the input encoding grows quadratically with the circuit size, i.e., the encodings for a circuit of size s have length $O(s^2)$. A followup work of Ball, Malkin, and Rosulek [9] investigates the concrete efficiency of arithmetic garbling gadgets in the context of secure computation. Their constructions generalize the free-XOR technique [33] and therefore assume the existence of correlation robust hashes.

Attribute-Based Encryption was first introduced in the seminal work of Goyal et al. [40] and refined in [28,36]. Predicate encryption (PE) is a special case of ABE where ciphertexts hide the encoded policy, in addition to the message. This notion was proposed by Katz, Sahai and Waters in [32] where they constructed a scheme for inner-products: The owner of a key for a vector $\boldsymbol{x}$ can decrypt a ciphertext generated over a vector $\boldsymbol{y}$ only if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. ZIPE and NIPE schemes have been first identified in the work of Attrapadung and Libert [8] where the authors also showed the utility of public-index inner-product PE. The efficiency of inner-product-based schemes was improved by Chen, Gay and Wee in [19] and the first NIPE with constant-size ciphertexts has been recently proposed in [20]. To the best of our knowledge, all of the known instances of ZIPE and NIPE rely on bilinear maps or on the intractability of lattice problems.

In a different line of research Abdalla et al. [1] suggested a functional encryption scheme for inner-products (IPFE) where ciphertexts encode some vector $\boldsymbol{x}$ and users with keys for a vector $\boldsymbol{y}$ can learn the inner-product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. This result has been extended to achieve full security [3], function privacy [12], and support quadratic functions [10]. In the latter work, Baltico et al. also proposed a generic transformation to predicate encryption for bilinear map evaluation. However, such a transformation retains security only for those vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \{0, 1\}$. In particular, this is insufficient to instantiate a fully-fledged NIPE, where the range of the inner-product spans the whole $\mathbb{Z}_p$. Following a similar paradigm, a trace-and-revoke scheme has been recently proposed by Agrawal et al. [2]. A related work by Parno et al. [37] identifies a surprising connection between attribute-based encryption and verifiable computation.

Concurrent Work. In concurrent and independent work, Katsumata and Yamada [31] show a similar compiler from inner-product functional encryption to NIPE, thus also obtaining a NIPE from the DDH assumption. However, the crucial difference with respect to our work is that their NIPE does not achieve any form of attribute-hiding, which is necessary to instantiate our construction of arithmetic garbling for inner-product predicates.

2 Preliminaries

In this section we introduce the notation and some basic definitions that we will use throughout our work. We denote by $\lambda \in \mathbb{N}^+$ the security parameter and by $\mathsf{poly}(\lambda)$ any function that is bounded by a polynomial in λ. We call a function $\mathsf{negl}(\lambda)$ if for every $c \in \mathbb{Z}$, there exists some $N \in \mathbb{Z}$ such that for all $\lambda > N$ it holds that $\mathsf{negl}(\lambda) < \frac{1}{\lambda^c}$. We say that an algorithm is PPT if it is modeled as a probabilistic Turing machine whose

$\mathsf{ExpPriv}^{\mathcal{A}}_{\mathsf{GC},\Phi}(1^\lambda)$

$(C, x) \leftarrow \mathcal{A}(1^\lambda)$
if $x \notin \{0,1\}^n$ **return** $\bot$
$b \leftarrow\!\!\$\ \{0,1\}$
if $b = 0 : (\tilde{C}, e, d) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$
$\quad\quad \tilde{X} \leftarrow \mathsf{GC.Enc}(e, x)$
if $b = 1 : y \leftarrow \mathsf{GC}.ev(C, x)$
$\quad\quad (\tilde{C}, \tilde{X}, d) \leftarrow \mathcal{S}(1^\lambda, y, \Phi(C))$
$b' \leftarrow \mathcal{A}(\tilde{C}, \tilde{X}, d)$
return $b = b'$

(a) Privacy game.

$\mathsf{ExpAuth}^{\mathcal{A}}_{\mathsf{GC}}(1^\lambda)$

$(C, x) \leftarrow \mathcal{A}(1^\lambda)$
if $x \notin \{0,1\}^n$ **return** $\bot$
$(\tilde{C}, e, d) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$
$\tilde{X} \leftarrow \mathsf{GC.Enc}(e, x)$
$\tilde{Y} \leftarrow \mathcal{A}(\tilde{C}, \tilde{X})$
return $(\tilde{Y} \neq \mathsf{GC.Eval}(\tilde{C}, \tilde{X})$
$\quad$ **and** $\mathsf{GC.Dec}(d, \tilde{Y}) \neq \bot)$

(b) Authenticity game.

Fig. 1: The description of the security games for garbling schemes.

running time is bounded by some function $\mathsf{poly}(\lambda)$. Given a set S, we denote by $x \leftarrow\!\!\$\ S$ the sampling of an element uniformly at random from S. Vectors $(x_1, \ldots, x_n)$ are written as $\boldsymbol{x}$ and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes the inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$.

2.1 Garbling Schemes

We recall the notion of garbling schemes as presented in [11].

Definition 1 (Garbling Scheme). *A garbling scheme* $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Enc}, \mathsf{GC.Dec}, \mathsf{GC.Eval}, \mathsf{GC.ev})$ *is a tuple of algorithms, where* C *describes the circuit* $\mathsf{GC.ev}(C, \cdot) : \mathbb{F}^n \mapsto \mathbb{F}^m$ *that we want to garble and where the remaining algorithms follow.*

$\underline{(\tilde{C}, e, d) \leftarrow \mathsf{GC.Garble}(1^\lambda, C) :}$ *On input* 1^λ *and* f*, the probabilistic garbling algorithm outputs a garbled circuit description* $\tilde{C}$*, the input encoding information* e*, and the output decoding information* d*.*

$\underline{\tilde{X} \leftarrow \mathsf{GC.Enc}(e, x) :}$ *The (possibly) probabilistic encoding algorithm takes the encoding information* e *as input, and an initial input* $x \in \mathbb{F}^n$*, and outputs a garbled input* $\tilde{X}$*.*

$\underline{\tilde{Y} \leftarrow \mathsf{GC.Eval}(\tilde{C}, \tilde{X}) :}$ *On input a garbled circuit* $\tilde{C}$ *and an garbled input* $\tilde{X}$*, evaluate* $\tilde{C}$ *on* $\tilde{X}$ *to produce a deterministic garbled output* $\tilde{Y}$*.*

$\underline{y \leftarrow \mathsf{GC.Dec}(d, \tilde{Y}) :}$ *The deterministic decoding algorithm takes as input the decoding information* d*, and a garbled output* $\tilde{Y}$*, and outputs a final output* $y \in \mathbb{F}^m$*.*

The security notions we consider are *privacy*, *obliviousness*, and *authenticity*. Privacy means that seeing the garbled circuit $\tilde{C}$ with a garbled input $\tilde{X}$ and decoding information d does not reveal more about C and x than $C(x)$. We denote by $\Phi(C)$ some side information that the garbled circuit leaks about the original circuit C, like its size and topology. To define this formally, we resort to the simulation-based definition, which can be shown to be equivalent to the game-based one via a standard argument (see [11]).

Definition 2 (Φ-Privacy). *Let* $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Enc}, \mathsf{GC.Dec}, \mathsf{GC.Eval}, \mathsf{GC.ev})$ *be a garbling scheme and* Φ *a side information function.* GC *achieves* Φ*-privacy if there exists a* PPT *simulator* $\mathcal{S}$ *such that for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$2 \cdot \Pr[\mathsf{ExpPriv}^{\mathcal{A}}_{\mathsf{GC},\Phi}(1^\lambda) = 1] - 1 \leq \mathsf{negl}(\lambda),$$

where $\mathsf{ExpPriv}^{\mathcal{A}}_{\mathsf{GC},\Phi}(1^\lambda)$ *is defined in Figure 1a.*

Authenticity prevents the adversary from learning any output labels other than the ones it can learn itself from evaluating the circuit. This prevents a malicious adversary from claiming the evaluation yielded a different result.

Definition 3 (Authenticity). *Let* $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Enc}, \mathsf{GC.Dec}, \mathsf{GC.Eval}, \mathsf{GC.ev})$ *be a garbling scheme.* GC *achieves authenticity if for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{ExpAuth}^{\mathcal{A}}_{\mathsf{GC}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{ExpAuth}^{\mathcal{A}}_{\mathsf{GC}}(1^\lambda)$ *is defined in Figure 1b.*

Obliviousness is formalized in Appendix A.

2.2 Predicate Encryption

We recall the concept of *predicate encryption* (PE) [32]. A PE scheme allows to encrypt messages for certain attributes and to encode Boolean predicates in the decryption keys. We denote by Σ an arbitrary set of attributes and by $\mathcal{F}$ an arbitrary family of predicates over Σ, which may depend on the security parameter λ and/or on the public parameters of the scheme. We assume the message space of the scheme to be a field $\mathbb{F}$.

Definition 4 (Predicate Encryption). *A* predicate encryption *scheme for a class of predicates* $\mathcal{F}$ *over the set of attributes* Σ *consists of four* PPT *algorithms* $(\mathsf{Setup}_{\mathsf{PE}}, \mathsf{KGen}_{\mathsf{PE}}, \mathsf{Enc}_{\mathsf{PE}}, \mathsf{Dec}_{\mathsf{PE}})$ *such that:*

$\underline{\mathsf{Setup}_{\mathsf{PE}}(1^\lambda)}$: *The setup algorithm outputs a master public key* ek *and a corresponding private key* msk.

$\underline{\mathsf{KGen}_{\mathsf{PE}}(\mathsf{msk}, f)}$: *The key generation algorithm takes as input the master secret key and a predicate* $f \in \mathcal{F}$. *It outputs a key* dk_f.

$\underline{\mathsf{Enc}_{\mathsf{PE}}(\mathsf{ek}, \mathsf{A}, m)}$: *The encryption takes as input the public key* ek, *an attribute* $\mathsf{A} \in \Sigma$, *and a message* m *in some associated message space. It returns a ciphertext* c.

$\underline{\mathsf{Dec}_{\mathsf{PE}}(\mathsf{dk}_f, c)}$: *The decryption algorithm takes as input a secret key* dk_f *and a ciphertext* c. *It outputs either a message* m *or a special symbol* $\perp$.

For correctness, we require that for all $\lambda \in \mathbb{N}$, all $(\mathsf{ek}, \mathsf{msk}) \in \mathsf{Setup}_{\mathsf{PE}}(1^\lambda)$, all $f \in \mathcal{F}$, all $\mathsf{dk}_f \in \mathsf{KGen}_{\mathsf{PE}}(\mathsf{msk}, f)$, all $m \in \mathbb{F}$, and all $\mathsf{A} \in \Sigma$:

If $f(\mathsf{A}) = 1$ then $\mathsf{Dec}_{\mathsf{PE}}(\mathsf{dk}_f, \mathsf{Enc}_{\mathsf{PE}}(\mathsf{ek}, \mathsf{A}, m)) = m$.
If $f(\mathsf{A}) = 0$ then $\mathsf{Dec}_{\mathsf{PE}}(\mathsf{dk}_f, \mathsf{Enc}_{\mathsf{PE}}(\mathsf{ek}, \mathsf{A}, m)) = \perp$ with all but negligible probability.

A predicate encryption is *attribute-hiding* if the ciphertexts hide the payload and the attribute. In this work we only consider a very weak version of this property, which is however sufficient for our purposes. In our game, the adversary must specify the challenge attributes $(\mathsf{A}^0, \mathsf{A}^1)$ together with the queries to the $\mathsf{KGen}_{\mathsf{PE}}$ oracle before being provided with the public parameters. Furthermore, the adversary can request only one key to the $\mathsf{KGen}_{\mathsf{PE}}$ oracle. We call this property *static attribute-hiding* and provide a formal definition below.

Definition 5 (Static Attribute-Hiding). *A predicate encryption scheme is* statically attribute-hiding *with respect to* $\mathbb{F}$ *if for all* PPT *adversaries* $\mathcal{A}$, *the advantage of* $\mathcal{A}$ *in the following experiment is negligible in the security parameter* λ.

1. *On input* 1^λ, $\mathcal{A}$ *outputs a pair of attributes* $(\mathsf{A}^0, \mathsf{A}^1) \in \Sigma^2$ *and a predicate* $f \in \mathcal{F}$. *If* $f(\mathsf{A}^0) \neq f(\mathsf{A}^1)$ *then the challenger aborts.*
2. $\mathsf{Setup}_{\mathsf{PE}}(1^\lambda)$ *is run to generate* ek *and* msk *and the adversary is given* ek *and* $\mathsf{KGen}_{\mathsf{PE}}(\mathsf{msk}, f)$.
3. $\mathcal{A}$ *outputs two equal-length messages* (m_0, m_1). *If* $f(\mathsf{A}^0) = f(\mathsf{A}^1) = 1$, *then it is required that* $m_0 = m_1$. *A random bit* b *is chosen and* $\mathcal{A}$ *is given the ciphertext* $c \leftarrow \mathsf{Enc}_{\mathsf{PE}}(\mathsf{ek}, \mathsf{A}^b, m_b)$.
4. $\mathcal{A}$ *outputs a bit* b' *and succeeds if* $b = b'$.

A predicate encryption scheme achieves *payload-hiding* if the message of a ciphertext is hidden from users whose predicate does not satisfy the encoded vector $\boldsymbol{x}$. In the literature this notion can be either presented in its *selective* or *adaptive* variant: In the former the adversary is required to commit to the challenge attribute A before even seeing the public parameters of the PE, while in the latter the adversary can specify A in the challenge phase, together with the challenge message pair (m_0, m_1). Clearly, selective payload-hiding is a strictly weaker notion, however it is still meaningful in certain contexts. In the following we formally define the notion of selective payload-hiding PE, the adaptive version comes as a straightforward modification of such a definition.

Definition 6 (Selective Payload-Hiding). *A PE scheme is* selectively payload-hiding *with respect to $\mathcal{F}$ and Σ if for all* PPT *adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter λ.*

1. *On input 1^λ, $\mathcal{A}$ outputs an attribute $\mathsf{A} \in \Sigma$.*
2. *$\mathsf{Setup}_{\mathsf{PE}}(1^\lambda)$ is run to generate ek and msk and the adversary is given ek.*
3. *$\mathcal{A}$ may adaptively request keys for any predicates $f_1, \ldots, f_q \in \mathcal{F}$. In response, $\mathcal{A}$ is given the corresponding keys $\mathsf{dk}_{f_i} \leftarrow \mathsf{KGen}_{\mathsf{PE}}(\mathsf{msk}, f_i)$.*
4. *$\mathcal{A}$ outputs two equal-length messages (m_0, m_1). If there is an i such that $f_i(\mathsf{A}) = 1$, then it is required that $m_0 = m_1$. A random bit b is chosen and $\mathcal{A}$ is given the ciphertext $c \leftarrow \mathsf{Enc}_{\mathsf{PE}}(\mathsf{ek}, \mathsf{A}, m_b)$.*
5. *The adversary may continue to request keys for additional predicates, subject to the same restrictions as before.*
6. *$\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$.*

Zero Inner-Product Encryption. Zero inner-product encryption (ZIPE) is a specific type of predicate encryption first explored by Katz, Sahai, and Waters [32] in 2008. In ZIPE the attributes are vectors of field elements and the predicates are functions indexed by another vector of field elements that evaluate to 1 exactly if the inner-product of the two vectors is 0. Formally this can be defined as follows.

Definition 7 (Zero Inner-Product Encryption). *A zero inner-product encryption scheme for vectors of length n over a field $\mathbb{F}$ is a PE scheme for the class of attributes $\Sigma = \mathbb{F}^n$ and the class of predicates $\mathcal{F} = \{ f_{\boldsymbol{y}} \mid \boldsymbol{y} \in \mathbb{F}^n \}$ where $f_{\boldsymbol{y}} : \mathbb{F}^n \to \{0, 1\}$ is defined as*

$$f_{\boldsymbol{y}}(\boldsymbol{x}) := \begin{cases} 0 & \text{if } \langle \boldsymbol{y}, \boldsymbol{x} \rangle \neq 0 \\ 1 & \text{otherwise.} \end{cases}$$

Non-Zero Inner-Product Encryption. This specific type of predicate encryption was first introduced by Attrapadung and Libert [8]. In a non-zero inner-product encryption (NIPE) scheme the attributes are vectors of field elements and the predicates are indexed by another vector of field elements and return 1 exactly if the inner-product of the two vectors is $\neq 0$. Formally it can be defined as follows.

Definition 8 (Non-Zero Inner-Product Encryption). *A zero inner-product encryption scheme for vectors of length n over a field $\mathbb{F}$ is a PE scheme for the class of attributes $\Sigma = \mathbb{F}^n$ and the class of predicates $\mathcal{F} = \{ f_{\boldsymbol{y}} \mid \boldsymbol{y} \in \mathbb{F}^n \}$ where $f_{\boldsymbol{y}} : \mathbb{F}^n \to \{0, 1\}$ is defined as*

$$f_{\boldsymbol{y}}(\boldsymbol{x}) := \begin{cases} 0 & \text{if } \langle \boldsymbol{y}, \boldsymbol{x} \rangle = 0 \\ 1 & \text{otherwise.} \end{cases}$$

2.3 Inner-Product Functional Encryption

Here we provide a formal definition of an Inner-Product Functional Encryption (IPFE) scheme. This primitive was introduced in the work of Abdalla et. al [1]. Such a scheme allows one to encode a vector field elements $\boldsymbol{v} \in \mathbb{F}^n$ in a ciphertext and to produce decryption keys for a vector $\boldsymbol{u}$ of the same family. Intuitively, security demands that one can learn nothing beyond the *inner-product* of $\boldsymbol{u}$ and $\boldsymbol{v}$. More formally:

Definition 9 (Inner-Product Functional Encryption). *Let $\mathbb{F}$ be a field. An* inner-product functional encryption *scheme consists of four* PPT *algorithms* $(\mathsf{Setup}_{\mathsf{IPFE}}, \mathsf{KGen}_{\mathsf{IPFE}}, \mathsf{Enc}_{\mathsf{IPFE}}, \mathsf{Dec}_{\mathsf{IPFE}})$ *such that:*

$\underline{\mathsf{Setup}_{\mathsf{IPFE}}(1^\lambda):}$ *The setup algorithm outputs a master public key* ek *and a corresponding private key* msk.

$\underline{\mathsf{KGen}_{\mathsf{IPFE}}(\mathsf{msk}, \boldsymbol{y}):}$ *The key generation algorithm takes as input the master secret key and a vector* $\boldsymbol{y} \in \mathbb{F}^n$. *It outputs a key* $\mathsf{dk}_{\boldsymbol{y}}$.

$\underline{\mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, \boldsymbol{x}):}$ *The encryption takes as input the public key* ek *and a vector* $\boldsymbol{x} \in \mathbb{F}^n$. *It returns a ciphertext* c.

$\underline{\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{dk}_{\boldsymbol{y}}, c):}$ *The decryption algorithm takes as input a secret key* $\mathsf{dk}_{\boldsymbol{y}}$ *and a ciphertext* c. *It outputs either a message* $m \in \mathbb{F}$ *or a special symbol* $\perp$.

We define correctness as: for all λ, all $(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IPFE}}(1^\lambda)$, all $\boldsymbol{y} \in \mathbb{F}^n$, all $\mathsf{dk}_{\boldsymbol{y}} \leftarrow \mathsf{KGen}_{\mathsf{IPFE}}(\mathsf{msk}, \boldsymbol{y})$, and all $\boldsymbol{x} \in \mathbb{F}^n$ it holds that $\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{dk}_{\boldsymbol{y}}, \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, \boldsymbol{x})) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. An IPFE scheme is semantically secure if an adversary holding a key for a vector $\boldsymbol{y}$ and a ciphertext under a vector $\boldsymbol{x}$ does not learn anything except for $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. The formal definition is elaborated below.

Definition 10 (Semantic Security). *An IPFE scheme is* semantically secure *if for all* PPT *adversaries* $\mathcal{A}$, *the advantage of* $\mathcal{A}$ *in the following experiment is negligible in the security parameter* λ.

1. $\mathsf{Setup}_{\mathsf{IPFE}}(1^\lambda)$ *is run to generate* ek *and* msk *and the adversary is given* $(1^\lambda, \mathsf{ek})$.
2. $\mathcal{A}$ *may adaptively request keys for vector* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_q \in \mathbb{F}^n$. *In response,* $\mathcal{A}$ *is given the corresponding keys* $\mathsf{dk}_{\boldsymbol{y}_i} \leftarrow \mathsf{KGen}_{\mathsf{IPFE}}(\mathsf{msk}, \boldsymbol{y}_i)$.
3. $\mathcal{A}$ *outputs two equal-length messages* $(\boldsymbol{x}_0, \boldsymbol{x}_1) \in \mathbb{F}^{2n}$. *If there is an* i *such that* $\langle \boldsymbol{y}_i, \boldsymbol{x}_0 \rangle \neq \langle \boldsymbol{y}_i, \boldsymbol{x}_1 \rangle$, *then the challenger aborts. A random bit* b *is chosen and* $\mathcal{A}$ *is given the ciphertext* $c \leftarrow \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, \boldsymbol{x}_b)$.
4. *The adversary may continue to request keys for additional vectors, subject to the same restrictions as above.*
5. $\mathcal{A}$ *outputs a bit* b' *and succeeds if* $b = b'$.

2.4 Bilinear Maps and Complexity Assumptions

Here we recall the notion of bilinear maps and introduce our complexity assumptions. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of large prime order p. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be respective generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a function that maps pairs of elements in $(\mathbb{G}_1, \mathbb{G}_2)$ to elements of some cyclic group $\mathbb{G}_T$ of order p. Throughout the following sections we write all of the group operations multiplicatively, with identity elements denoted by 1. We require that:

- The map e and all the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are efficiently computable.
- The map e is non degenerate, i.e., $e(g_1, g_2) \neq 1$.
- The map e is bilinear, i.e., $\forall u \in \mathbb{G}_1, \forall v \in \mathbb{G}_2, \forall (a, b) \in \mathbb{Z}^2, e(u^a, v^b) = e(u, v)^{ab}$.

We prove the security of one of our schemes based on the External Diffie-Hellman (XDH) assumption and the Decisional-Bilinear Diffie-Hellman (DBDH) assumption. The latter is the standard extensions of the Decisional Diffie-Hellman assumption [13] in groups with pairing. Such an assumption was introduced in the context of identity-based encryption in the seminal work of Boneh and Franklin [14]. We define the XDH and the DBDH assumption in the following.

Definition 11 (XDH Assumption). *The XDH assumption holds in* $\mathbb{G}_1$ *if, for all* PPT *algorithms* $\mathcal{A}$, *there exists a negligible function* negl *such that*

$$\left| \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(g_1, g_1^a, g_1^b, g_1^{ab}\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(g_1, g_1^a, g_1^b, g_1^c\right)\right] \right| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random choice of the generator $g_1 \in \mathbb{G}_1$, *the random choice of* $(a, b, c) \in (\mathbb{Z}_p^*)^3$, *and the random coins of* $\mathcal{A}$.

Definition 12 (DBDH Assumption). *The DBDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if, for all* PPT *algorithms* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left| \begin{array}{l} \mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b, e(g_1,g_2)^{abc}\right)\right] \\ -\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b, e(g_1,g_2)^{z}\right)\right] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random choice of $g_1 \in \mathbb{G}_1$ *and* $g_2 \in \mathbb{G}_2$*, the random choice of* $(a, b, c, z) \in (\mathbb{Z}_p^*)^4$*, and the random coins of* $\mathcal{A}$*.*

3 Arithmetic Garbling for Inner-Products

In this section we present our compiler that turns predicate encryption schemes into garbling schemes for inner-product predicates. Our first observation is that the ciphertext of an inner-product predicate encryption scheme can already be seen as a garbled circuit if it satisfies some mild attribute-hiding properties. However, to achieve authenticity one needs to encrypt a predicate f and its complement $\overline{f}$. For inner-product predicates, the complement of the predicate defined by a vector $\boldsymbol{x}$ is a vector $\boldsymbol{y}$ such that for all $\boldsymbol{z}$ it holds that $\langle \boldsymbol{x}, \boldsymbol{z} \rangle = 0 \implies \langle \boldsymbol{y}, \boldsymbol{z} \rangle \neq 0$ and vice versa. Thus $\boldsymbol{y}$ is not always efficiently computable. We resolve this issue by lifting the complement to the encryption scheme. Our construction (Figure 2) garbles the family of predicates $\mathcal{F} = \{ f_{\boldsymbol{y}} \mid \boldsymbol{y} \in \mathbb{F}^n \}$ where $f_{\boldsymbol{y}} : \mathbb{F}^n \to \{0,1\}$ is defined as

$$f_{\boldsymbol{y}}(\boldsymbol{x}) := \begin{cases} 0 & \text{if } \langle \boldsymbol{y}, \boldsymbol{x} \rangle = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Our scheme uses a $\mathsf{NIPE} = (\mathsf{Setup}_{\mathsf{NIPE}}, \mathsf{KGen}_{\mathsf{NIPE}}, \mathsf{Enc}_{\mathsf{NIPE}}, \mathsf{Dec}_{\mathsf{NIPE}})$ and $\mathsf{ZIPE} = (\mathsf{Setup}_{\mathsf{ZIPE}}, \mathsf{KGen}_{\mathsf{ZIPE}}, \mathsf{Enc}_{\mathsf{ZIPE}}, \mathsf{Dec}_{\mathsf{ZIPE}})$ for inner products of vectors in $\mathbb{F}^{n+1}$ in a blackbox way.

Theorem 1. *Let* Φ *be the function that takes as input an inner-product predicate and returns the length of its vector. Let* NIPE *be a statically attribute-hiding NIPE scheme and let* ZIPE *be a statically attribute-hiding ZIPE scheme, then the garbling scheme* GC *as described in Figure 2 achieves* Φ*-privacy.*

Proof. We begin by describing the simulator $\mathcal{S}_{\mathsf{GC}}$ in Figure 3. Then we define the following sequence of hybrids:

H_0 : Defined as the experiment described in Definition 2 with the bit $b = 0$.
H_1 : Defined as H_0 except that if $f_{\boldsymbol{y}}(\boldsymbol{x}) = 0$, then a random $\boldsymbol{s} \leftarrow \mathbb{F}^{n+1}$ is sampled under the constraint that $\langle \boldsymbol{x} + \boldsymbol{r} \| 1, \boldsymbol{s} \rangle = 0$. If $f_{\boldsymbol{y}}(\boldsymbol{x}) = 1$, then a random $\boldsymbol{s} \leftarrow \mathbb{F}^{n+1}$ is sampled under the constraint that $\langle \boldsymbol{x} + \boldsymbol{r} \| 1, \boldsymbol{s} \rangle \neq 0$. In both cases c^0 is set to be $\mathsf{Enc}_{\mathsf{ZIPE}}\left(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{s}, r^0\right)$.
H_2 : Defined as H_1 except that c^1 is set to be $\mathsf{Enc}_{\mathsf{NIPE}}\left(\mathsf{ek}_{\mathsf{NIPE}}, \boldsymbol{s}, r^1\right)$.
H_3 : Defined as H_2 except that the keys $\mathsf{dk}_{\mathsf{ZIPE}}$ and $\mathsf{dk}_{\mathsf{NIPE}}$ are computed as $\mathsf{KGen}_{\mathsf{ZIPE}}(\mathsf{msk}_{\mathsf{ZIPE}}, \boldsymbol{r} \| 1)$ and $\mathsf{KGen}_{\mathsf{NIPE}}(\mathsf{msk}_{\mathsf{NIPE}}, \boldsymbol{r} \| 1)$, for a random $\boldsymbol{r} \in \mathbb{F}^n$.
H_4 : Defined as the experiment described in Definition 2 with the bit $b = 1$ and using the simulator described in Figure 3.

We proceed by proving the indistinguishability of each pair of hybrids.

Lemma 1. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_0}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_1}(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 1). Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$ such that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_0}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_1}(1^\lambda)\right]\right| \geq \epsilon(\lambda),$$

for some non-negligible function ϵ. Then we can construct the reduction $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$ in Figure 4 against the static attribute-hiding of the ZIPE scheme ZIPE. The reduction is clearly efficient. Note that when $b = 0$

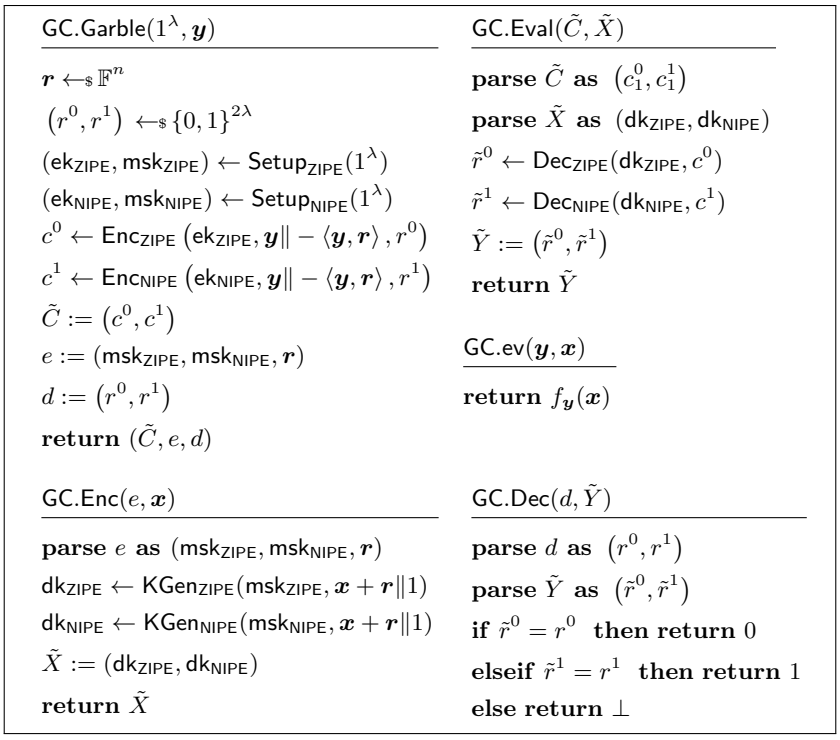

Fig. 2: An arithmetic garbling scheme for inner product predicates.

then c^* is of the form $\mathsf{Enc}_{\mathsf{ZIPE}}(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r} \rangle, r^0)$ which means that the reduction perfectly simulates the view of H_0. On the other hand when $b = 1$ then $c^* \leftarrow \mathsf{Enc}_{\mathsf{ZIPE}}(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{s}, r^0)$ which is exactly the input of H_1. What is left to be shown is that the tuple $((\boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r} \rangle, \boldsymbol{s}), (\boldsymbol{r} + \boldsymbol{x})\|1)$ is an admissible challenge for the static attribute-hiding experiment. We identify two cases:

1. $f_{\boldsymbol{y}}(\boldsymbol{x}) = 0$: In this case we have $\langle \boldsymbol{r} + \boldsymbol{x}\|1, \boldsymbol{s} \rangle = 0$ by definition and therefore

$$\begin{aligned}\langle \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r} \rangle, (\boldsymbol{r} + \boldsymbol{x})\|1 \rangle &= \langle \boldsymbol{y}, (\boldsymbol{r} + \boldsymbol{x}) \rangle - \langle \boldsymbol{y}, \boldsymbol{r} \rangle = \langle \boldsymbol{y}, \boldsymbol{r} \rangle + \langle \boldsymbol{y}, \boldsymbol{x} \rangle - \langle \boldsymbol{y}, \boldsymbol{r} \rangle \\ &= \langle \boldsymbol{y}, \boldsymbol{x} \rangle = 0 = \langle \boldsymbol{r} + \boldsymbol{x}\|1, \boldsymbol{s} \rangle .\end{aligned}$$

2. $f_{\boldsymbol{y}}(\boldsymbol{x}) = 1$: In this case we have $\langle \boldsymbol{r} + \boldsymbol{x}\|1, \boldsymbol{s} \rangle \neq 0$. Furthermore,

$$\langle \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r} \rangle, (\boldsymbol{r} + \boldsymbol{x})\|1 \rangle = \langle \boldsymbol{y}, (\boldsymbol{r} + \boldsymbol{x}) \rangle - \langle \boldsymbol{y}, \boldsymbol{r} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle \neq 0.$$

We can conclude that the challenge tuple is legitimate. By initial assumption we have that

$$\left| \mathsf{Pr}\left[1 \leftarrow \mathcal{B}(1^\lambda) | b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B}(1^\lambda) | b = 1\right] \right| \geq \epsilon(\lambda),$$

which is a contradiction to the static attribute-hiding of the ZIPE scheme ZIPE and proves our lemma. □

Lemma 2. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left| \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_1}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_2}(1^\lambda)\right] \right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 2). The proof is identical to the proof of Lemma 1 except that the reduction is against the static attribute-hiding of NIPE. □

$\mathcal{S}_{\mathsf{GC}}(1^\lambda, \beta, n)$

$(\mathsf{ek}_{\mathsf{ZIPE}}, \mathsf{msk}_{\mathsf{ZIPE}}) \leftarrow \mathsf{Setup}_{\mathsf{ZIPE}}(1^\lambda)$; $(\mathsf{ek}_{\mathsf{NIPE}}, \mathsf{msk}_{\mathsf{NIPE}}) \leftarrow \mathsf{Setup}_{\mathsf{NIPE}}(1^\lambda)$

$\boldsymbol{r} \leftarrow_\$ \mathbb{F}^n$; $\left(r^0, r^1\right) \leftarrow_\$ \{0,1\}^{2\lambda}$; $d := \left(r^0, r^1\right)$

$\mathsf{dk}_{\mathsf{ZIPE}} \leftarrow \mathsf{KGen}_{\mathsf{ZIPE}}(\mathsf{msk}_{\mathsf{ZIPE}}, \boldsymbol{r}\|1)$; $\mathsf{dk}_{\mathsf{NIPE}} \leftarrow \mathsf{KGen}_{\mathsf{NIPE}}(\mathsf{msk}_{\mathsf{NIPE}}, \boldsymbol{r}\|1)$

$\tilde{X} := (\mathsf{dk}_{\mathsf{ZIPE}}, \mathsf{dk}_{\mathsf{NIPE}})$

if $\beta = 0$ **then** $\boldsymbol{s} \leftarrow \mathbb{F}^{n+1}$ s.t. $\langle \boldsymbol{r}\|1, \boldsymbol{s}\rangle = 0$

if $\beta = 1$ **then** $\boldsymbol{s} \leftarrow \mathbb{F}^{n+1}$ s.t. $\langle \boldsymbol{r}\|1, \boldsymbol{s}\rangle \neq 0$

$c^0 \leftarrow \mathsf{Enc}_{\mathsf{ZIPE}}\left(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{s}, r^0\right)$; $c^1 \leftarrow \mathsf{Enc}_{\mathsf{NIPE}}\left(\mathsf{ek}_{\mathsf{NIPE}}, \boldsymbol{s}, r^1\right)$; $\tilde{C} := \left(c^0, c^1\right)$

return $\left(\tilde{C}, \tilde{X}, d\right)$

Fig. 3: Construction of the simulator $\mathcal{S}_{\mathsf{GC}}$.

$\mathcal{B}_1(1^\lambda)$

$(\boldsymbol{y}, \boldsymbol{x}) \leftarrow \mathcal{A}(1^\lambda)$; $\boldsymbol{r} \leftarrow_\$ \mathbb{F}^n$

$(\mathsf{ek}_{\mathsf{NIPE}}, \mathsf{msk}_{\mathsf{NIPE}}) \leftarrow \mathsf{Setup}_{\mathsf{NIPE}}(1^\lambda)$

$\mathsf{dk}_{\mathsf{NIPE}} \leftarrow \mathsf{KGen}_{\mathsf{NIPE}}(\mathsf{msk}_{\mathsf{NIPE}}, \boldsymbol{x} + \boldsymbol{r}\|1)$

if $\langle \boldsymbol{y}, \boldsymbol{x}\rangle = 0$ **then**

  $\boldsymbol{s} \leftarrow \mathbb{F}^{n+1}$ s.t. $\langle (\boldsymbol{r} + \boldsymbol{x})\|1, \boldsymbol{s}\rangle = 0$

if $\langle \boldsymbol{y}, \boldsymbol{x}\rangle = 1$ **then**

  $\boldsymbol{s} \leftarrow \mathbb{F}^{n+1}$ s.t. $\langle (\boldsymbol{r} + \boldsymbol{x})\|1, \boldsymbol{s}\rangle \neq 0$

return $\left((\boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, \boldsymbol{s}), (\boldsymbol{r} + \boldsymbol{x})\|1\right)$

$\mathcal{B}_2(\mathsf{ek}, \mathsf{dk})$

$\left(r^0, r^1\right) \leftarrow_\$ \{0,1\}^{2\lambda}$

return $\left(r^0, r^0\right)$

$\mathcal{B}_3(c^*)$

$c^1 \leftarrow \mathsf{Enc}_{\mathsf{NIPE}}(\mathsf{ek}_{\mathsf{NIPE}}, \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, r^1)$

$d := (r^0, r^1)$; $\tilde{X} := (\mathsf{dk}, \mathsf{dk}_{\mathsf{NIPE}})$

$\tilde{C} := \left(c^*, c^1\right)$; $b' \leftarrow \mathcal{A}\left(\tilde{C}, \tilde{X}, d\right)$

return b'

Fig. 4: Reduction against the static attribute-hiding of ZIPE.

Lemma 3. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_2}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_3}(1^\lambda)\right]\right| = 0.$$

Proof (of Lemma 3). Here the observation is that if $\boldsymbol{r}$ is sampled uniformly at random from a field $\mathbb{F}^n$, then for all $\boldsymbol{x} \in \mathbb{F}^n$ we have that $\boldsymbol{r}$ and $\boldsymbol{x} + \boldsymbol{r}$ are identically distributed. Therefore the modification is only syntactical. □

Lemma 4. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_3}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_4}(1^\lambda)\right]\right| = 0.$$

Proof (of Lemma 4). The experiment H_3 is identical to the simulator $\mathcal{S}_{\mathsf{GC}}$ (Figure 3) and thus the change is only syntactical. □

By Lemma 1, Lemma 2, Lemma 3, Lemma 4 we have that the H_0 is computationally indistinguishable from H_4, which means that the advantage of any PPT adversary $\mathcal{A}$ in the privacy experiment (Definition 2) can be bound to a negligible function in the security parameter. □

Theorem 2. *Let* NIPE *be a statically attribute-hiding NIPE scheme and let* ZIPE *be a statically attribute-hiding ZIPE scheme, then the garbling scheme* GC *as described in Figure 2 achieves authenticity.*

Proof. We begin by defining the following sequence of hybrids.

H_0 : Defined as the experiment described in Definition 3.

H_1 : Same as H_0 except that $f_{\boldsymbol{y}}(\boldsymbol{x}) = 1 : c^0 \leftarrow \mathsf{Enc}_{\mathsf{ZIPE}}\left(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, 0^\lambda\right)$.

H_2 : Same as H_1 except that if $f_{\boldsymbol{y}}(\boldsymbol{x}) = 0 : c^1 \leftarrow \mathsf{Enc}_{\mathsf{NIPE}}\left(\mathsf{ek}_{\mathsf{NIPE}}, \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, 0^\lambda\right)$.

We show that the distance between neighbouring experiments is negligible in the security parameter.

Lemma 5. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_0}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_1}(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 5). Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$ such that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_0}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_1}(1^\lambda)\right]\right| \geq \epsilon(\lambda),$$

for some non-negligible function ϵ. Then we can construct the reduction $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$ in Figure 5 against the static attribute-hiding of the ZIPE scheme ZIPE. The reduction is clearly efficient. Note that when $b = 0$

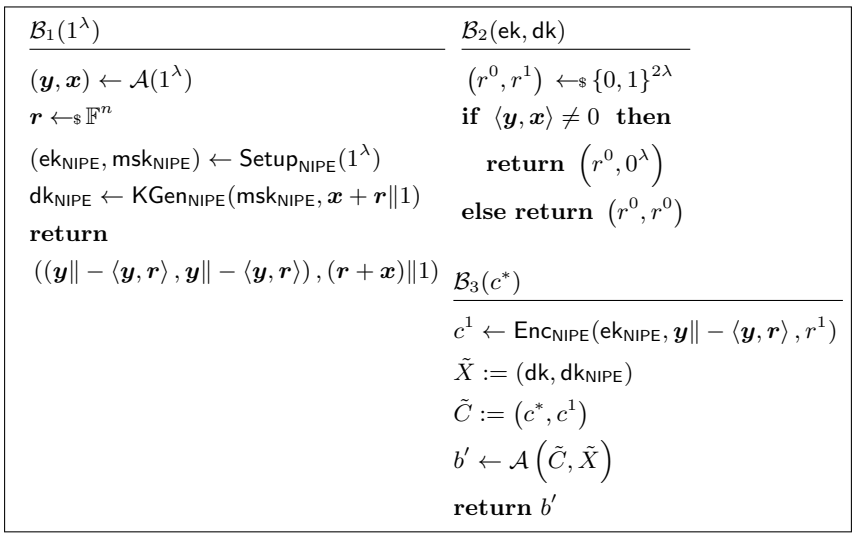

Fig. 5: Reduction against the static attribute-hiding of ZIPE.

then c^* is of the form $\mathsf{Enc}_{\mathsf{ZIPE}}(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, r^0)$ which means that the reduction perfectly simulates the view of H_0. On the other hand when $b = 1$ then $c^* \leftarrow \mathsf{Enc}_{\mathsf{ZIPE}}(\mathsf{ek}_{\mathsf{ZIPE}}, \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, 0^\lambda)$ if and only if $\langle \boldsymbol{y}, \boldsymbol{x}\rangle \neq 0$, which is exactly the input of H_1. We now argue that the reduction $\mathcal{B}$ is an admissible adversary for the static attribute-hiding experiment. Note that the challenge attributes are identical, so the challenger does not abort if the inner product of the challenge and the predicate is different from 0. By assumption we have that

$$\begin{aligned}\langle \boldsymbol{y}\| - \langle \boldsymbol{y}, \boldsymbol{r}\rangle, (\boldsymbol{r} + \boldsymbol{x})\|1\rangle &= \langle \boldsymbol{y}, (\boldsymbol{r} + \boldsymbol{x})\rangle - \langle \boldsymbol{y}, \boldsymbol{r}\rangle = \langle \boldsymbol{y}, \boldsymbol{r}\rangle + \langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, \boldsymbol{r}\rangle \\ &= \langle \boldsymbol{y}, \boldsymbol{x}\rangle \neq 0,\end{aligned}$$

which satisfies the condition for an admissible adversary. It follows that

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{B}(1^\lambda)|b = 0\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B}(1^\lambda)|b = 1\right]\right| \geq \epsilon(\lambda),$$

which is a contradiction to the static attribute-hiding of the ZIPE scheme ZIPE and proves our lemma. □

Lemma 6. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_1}(1^\lambda)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{A}^{H_2}(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 6). The proof is identical to the proof of Lemma 5 except that the reduction is against the static attribute-hiding of the NIPE scheme NIPE. The modifications are obvious and therefore are omitted. □

Finally we show that the advantage of any PPT adversary $\mathcal{A}$ in H_2 is negligibly close to 0.

Lemma 7. *For all* PPT $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\mathsf{Pr}\left[1 \leftarrow H_2^{\mathcal{A}}(1^\lambda)\right] \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 7). To prove this lemma it is enough to observe that, for the case $f_{\boldsymbol{y}}(\boldsymbol{x}) = 1$, the value of r^0 is information theoretically hidden to the eyes of $\mathcal{A}$ and therefore the probability that $\mathcal{A}$ outputs r^0 is negligible in the security parameter. The same argument works also for the case $f_{\boldsymbol{y}}(\boldsymbol{x}) = 0$. □

This concludes our proof. □

Achieving obliviousness requires some modification of the garbling algorithm in a non-blackbox fashion, which we describe in Appendix A.

Input Encoding. The core property of arithmetic garbled circuits is to reduce the task to evaluate a circuit to evaluating an *affine* function over the inputs that depends only on some randomness (and in particular not on the input itself). This corresponds to the encoding function, which in our construction consists of the evaluation of the key generation of the NIPE and ZIPE schemes. Concerning our instantiations, the NIPE's key generation (see Section 4) is identical to the algorithm of the underlying IPFE scheme which in turn, for all the schemes proposed by Agrawal et al. [3], evaluates an affine function over the inputs. For the case of the ZIPE (see Section 5), the key generation corresponds to the computation of an affine function "in the exponent". Both classes of functions are well studied in the context of secure arithmetic computation and admit efficient secure evaluation protocols [34].

4 Non-Zero Inner Product Encryption

Our construction of NIPE is a simple transformation based on inner product functional encryption (see Definition 9). The basic idea is to encrypt the attribute vector $\boldsymbol{x}$ multiplied by the message m with an IPFE scheme. Clearly, trying to decrypt with a vector $\boldsymbol{y}$ such that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, will destroy all information about m, since the decryption algorithm will output $\langle m\boldsymbol{x}, \boldsymbol{y} \rangle = m \langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. On the other hand, decrypting with any vector such that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$ yields a blinded version of the message, where the mask is $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Since $\boldsymbol{x}$ is unknown to the eyes of the decryptor, to enable correctness, we should add a second IPFE ciphertext, of the vector $\boldsymbol{x}$, which decrypted with the same vector $\boldsymbol{y}$ will reveal the blinding factor. However, this simple version does not achieve any form of attribute-hiding, since the decryptor can learn non-trivial information about $\boldsymbol{x}$, in case of correct decryption. To counter this issue, we blind the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ with an additional random scalar r. In the security analysis, this additional degree of freedom will allow us to equivocate the attributes. The scheme is formally described in Figure 6.

It is easy to see that NIPE is correct. By the correctness of IPFE, it holds that $s = \langle mr\boldsymbol{x}, \boldsymbol{y} \rangle = mr \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $t = \langle r\boldsymbol{x}, \boldsymbol{y} \rangle = r \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. If $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0$ we therefore have $st^{-1} = \frac{mr\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{r\langle \boldsymbol{x}, \boldsymbol{y} \rangle} = m$. If on the other hand $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$ then $t = 0$ and the decryption algorithm will always output $\perp$.

Theorem 3. *Let* IPFE *be a semantically secure inner product functional encryption scheme, then the non-zero inner product encryption scheme* NIPE *shown in Figure 6 is adaptively payload-hiding.*

$\underline{\mathsf{Setup}_{\mathsf{NIPE}}(1^\lambda)}$

$(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IPFE}}(1^\lambda)$

return $(\mathsf{ek}, \mathsf{msk})$

$\underline{\mathsf{KGen}_{\mathsf{NIPE}}(\mathsf{msk}, \boldsymbol{y})}$

$\mathsf{dk} \leftarrow \mathsf{KGen}_{\mathsf{IPFE}}(\mathsf{msk}, \boldsymbol{y})$

return dk

$\underline{\mathsf{Enc}_{\mathsf{NIPE}}(\mathsf{ek}, \boldsymbol{x}, m)}$

$r \leftarrow_\$ \mathbb{F}$

$c_1 \leftarrow \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, mr \cdot \boldsymbol{x})$

$c_2 \leftarrow \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, r \cdot \boldsymbol{x})$

return (c_1, c_2)

$\underline{\mathsf{Dec}_{\mathsf{NIPE}}(\mathsf{dk}, c)}$

parse c **as** (c_1, c_2)

$s \leftarrow \mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{dk}, c_1)$

$t \leftarrow \mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{dk}, c_2)$

if $t \neq 0$ **then return** st^{-1}

return $\perp$

Fig. 6: Construction of non-zero inner product encryption.

$\underline{\mathcal{B}_1^{\mathsf{KGen}_{\mathsf{IPFE}}(\mathsf{msk}, \cdot)}(\mathsf{ek})}$

$(m_0, m_1, \boldsymbol{x}) \leftarrow \mathcal{A}^{\mathsf{KGen}_{\mathsf{IPFE}}(\cdot)}(\mathsf{ek})$

$r \leftarrow_\$ \mathbb{F}$

return $(m_0 r \cdot \boldsymbol{x}, m_1 r \cdot \boldsymbol{x})$

$\underline{\mathcal{B}_2^{\mathsf{KGen}_{\mathsf{IPFE}}(\mathsf{msk}, \cdot)}(c^*)}$

$c_2 := \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, r \cdot \boldsymbol{x})$

$b' \leftarrow \mathcal{A}^{\mathsf{KGen}_{\mathsf{IPFE}}(\cdot)}(c^*, c_2)$

return b'

Fig. 7: The reduction $\mathcal{B}$ from the adaptive payload-hiding of NIPE to the semantic security of IPFE.

Proof. Let $\mathcal{A}$ be an algorithm that breaks the adaptive payload-hiding property of NIPE with probability $1/2+\epsilon(\lambda)$. We then construct an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ breaking the semantic security of IPFE with the same probability as depicted in Figure 7. Clearly, the public key passed to $\mathcal{A}$ is distributed exactly as specified by the scheme. Further note that the attacker can only be successful if for all $\boldsymbol{y_i}$ it queries to the key generation oracle it must hold that $\langle \boldsymbol{x}, \boldsymbol{y_i} \rangle = 0$, since otherwise the attacker would have to choose $m_0 = m_1$. Therefore, it holds that for all i, j, $\langle \boldsymbol{x}, \boldsymbol{y_i} \rangle = \langle \boldsymbol{x}, \boldsymbol{y_j} \rangle = 0$ and thus the condition on keys in the semantic security game holds. It follows that the key generation oracle is simulated perfectly. Further, the challenge ciphertext is always a correctly distributed ciphertext of m_b, where b is the challenge bit chosen by the semantic security game. Therefore, whenever $\mathcal{A}$ is successful, so is $\mathcal{B}$. It follows that $\mathcal{B}$ is successful with probability $1/2 + \epsilon(\lambda)$. Since IPFE is semantically secure, $\epsilon(\lambda)$ must be negligible and NIPE is thus adaptively payload-hiding. □

Theorem 4. *Let* IPFE *be a semantically secure inner product functional encryption scheme, then the non-zero inner product encryption scheme* NIPE *shown in Figure 6 is statically attribute-hiding.*

Proof. Let $\mathcal{A}$ be an algorithm that breaks the static attribute-hiding property of NIPE with probability $1/2 + \epsilon(\lambda)$. We then construct an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ breaking the semantic security of IPFE with the same probability as depicted in Figure 8. Note, that we use a notion of semantic security where $\mathcal{B}$ can output pairs of messages. This notion follows from standard semantic security with a simple hybrid argument.

Clearly, the public key passed to $\mathcal{A}$ is distributed exactly as specified by the scheme. Further, the attacker can only ask for a single decryption key and must do so before it outputs the challenge. The choice of r_1 ensures that the condition on keys made by the semantic security game is satisfied. While r_0 and r_1 are related, both uniformly distributed when viewed in isolation. The challenge ciphertext is therefore always a correctly distributed ciphertext of m_b, where b is the challenge bit chosen by the semantic security game. Therefore, whenever $\mathcal{A}$ is successful, so is $\mathcal{B}$. It follows that $\mathcal{B}$ is successful with probability $1/2 + \epsilon(\lambda)$. Since IPFE is semantically secure, $\epsilon(\lambda)$ must be negligible and NIPE is thus statically attribute-hiding. □

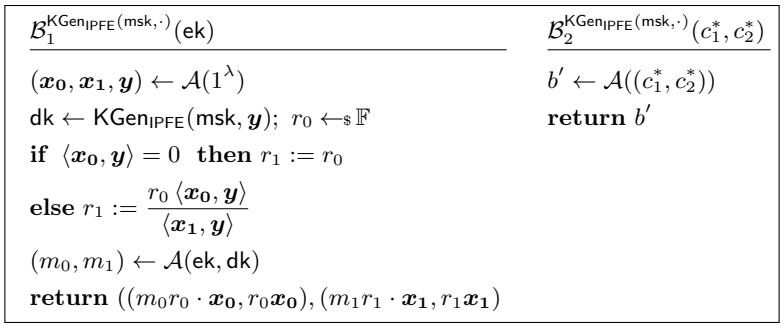

Fig. 8: The reduction $\mathcal{B}$ from the static attribute-hiding of NIPE to the semantic security of IPFE.

4.1 Instantiations

Our transformation relies on a very powerful abstraction of inner-product functional encryption: We require that the inner product of keys and messages is computed over a finite field and that the scheme supports messages from an exponentially large domain. Recent proposals for an adaptively-secure inner-product functional encryption [3] include:

1. A scheme from DDH for small messages for inner product computations over $\mathbb{Z}_p$. The key size is that of two integers in $\mathbb{Z}_p$.
2. A construction from LWE for inner products over $\mathbb{Z}_p$ with a stateful key generation algorithm, where keys are of size $\mathbb{Z}_p^\mu$, where μ depends on the security parameter but not on the vector length n.

A direct application of our transformation to the first scheme would yield a NIPE with an inefficient decryption: The output of the original scheme is of the form g^m, that in our construction would translate into two elements $(g^r, g^{mr}) \in \mathbb{G}^2$, for a randomly distributed r. To retrieve m one would then compute two discrete logarithms. However we can easily solve this, for messages m from a small domain, via a nonblack-box modification of our decryption algorithm. To retrieve m one can compute the discrete logarithm of g^{mr} to base g^r. Assuming a message space sufficiently small, this gives us the first full-fledged NIPE from the DDH assumption. We stress that a small message space does not hinder the applicability of a NIPE to our garbling scheme (Section 3), which requires one to encrypt a message $r^1 \in \{0,1\}^\lambda$ from a large space. We suggest two possible solutions to bypass this problem:

1. Split r^1 in small chunks, e.g., take its binary representation, and encrypt each block separately with fresh randomness. This variant is still secure by a standard hybrid argument.
2. Modify the garbling scheme in a non-black box way to set the decoding label corresponding to 1 to $H(g^r, g^{mr})$, where $H : \mathbb{G}^2 \to \{0,1\}^\lambda$ is any collision-resistant hash function. Note that the garbling algorithm has access to the random coins of the encryption, and therefore (g^r, g^{mr}) is efficiently computable. This allows the evaluation algorithm to recover the correct label without computing any discrete logarithm.

For the second instantiations, our NIPE scheme inherits the stateful key generation algorithm. This is not an issue in the context of garbling schemes since our construction issues only a single decryption key.

5 Zero Inner Product Encryption

Our ZIPE scheme is inspired by the work of Boneh et al. [15] and can be seen as a key-homomorphic public-key encryption for linear functions. Recall that a key-homomorphic public key encryption allows anyone to transform an encryption under attribute $\boldsymbol{x}$ into an encryption under $(f(\boldsymbol{x}), [f])$, where $f(\boldsymbol{x}) \in \mathbb{F}$ and $[f]$ is an encoding of the circuit computing f. This design paradigm offers a very natural way to instantiate an

attribute-based encryption scheme: One can simply issue a key for $(1, [f])$ and the decrypter can publicly apply the transformation of above to any cipher and decrypt if and only if $f(\boldsymbol{x}) = 1$. The advantage of this class of schemes is that the computational complexity is pushed to public operation and therefore the resulting decryption keys are typically small.

We exploit the same idea to construct a predicate encryption with small keys from bilinear maps. Since our public evaluation happens in the exponent, our scheme is key-homomorphic for linear functions only. However, a crucial difference with respect to the work of Boneh et al. [15] is that the structure of bilinear groups allows us to hide the attributes even to the eyes of the evaluator, which applies the function f *obliviously* over $\boldsymbol{x}$. The formal description of our construction is shown in Figure 9.

$\mathsf{Setup}_{\mathsf{ZIPE}}(1^\lambda)$

$(g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow \mathcal{G}(1^\lambda)$
$(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\ \mathbb{Z}_p^n$
$\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$
$\mathsf{msk} := (g_1, g_2, g_2^\alpha, \kappa_1, \ldots, \kappa_n)$
$\mathsf{ek} := (g_1, g_2, g_1^\alpha, g_1^{\kappa_1}, \ldots, g_1^{\kappa_n})$
return $(\mathsf{ek}, \mathsf{msk})$

$\mathsf{KGen}_{\mathsf{ZIPE}}(\mathsf{msk}, \boldsymbol{y})$

parse msk **as** $(g_1, g_2, a_2, \kappa_1, \ldots, \kappa_n)$
$r \leftarrow\!\!\$\ \mathbb{Z}_p$
$d_0 := a_2 \cdot \left(\prod_{i=1}^n g_2^{\kappa_i y_i}\right)^r$
$d_1 := g_2^r$
return $\mathsf{dk} := (d_0, d_1, \boldsymbol{y})$

$\mathsf{Enc}_{\mathsf{ZIPE}}(\mathsf{ek}, \boldsymbol{x}, m)$

parse ek **as** $(g_1, g_2, a_1, h_1, \ldots, h_n)$
$(\kappa_0, t) \leftarrow\!\!\$\ \mathbb{Z}_p^2$
$h_0 := g_1^{\kappa_0}$
$c_0 := m \cdot e(a_1, g_2)^t$
$c_0' := g_1^t$
for $i = \{1, \ldots, n\} : c_i := (h_0^{x_i} h_i)^t$
return $(c_0, c_0', c_1, \ldots, c_n)$

$\mathsf{Dec}_{\mathsf{ZIPE}}(\mathsf{dk}, c)$

parse dk **as** $(d_0, d_1, \boldsymbol{y})$
parse c **as** $(c_0, c_0', c_1, \ldots, c_n)$
$m := c_0 \dfrac{e\left(\prod_{i=1}^n c_i^{y_i}, d_1\right)}{e(c_0', d_0)}$
return m

Fig. 9: Construction of zero inner product encryption.

For correctness, evaluating the decryption algorithm we get the following:

$$
\begin{aligned}
c_0 \frac{e\left(\prod_{i=1}^n c_i^{y_i}, d_1\right)}{e(c_0', d_0)} &= m \cdot e(a_1, g_2)^t \frac{e\left(\prod_{i=1}^n (h_0^{x_i} h_i)^{t y_i}, g_2^r\right)}{e\left(g_1^t, a_2 \cdot (\prod_{i=1}^n g_2^{\kappa_i y_i})^r\right)} \\
&= m \cdot e(g_1, g_2)^{\alpha t} \frac{e\left(\prod_{i=1}^n (h_0^{x_i} h_i)^{y_i}, g_2\right)^{rt}}{e(g_1, g_2)^{\alpha t} e(g_1, \prod_{i=1}^n g_2^{\kappa_i y_i})^{rt}} \\
&= m \cdot \frac{e\left(\prod_{i=1}^n g_1^{y_i(\kappa_0 x_i + \kappa_i)}, g_2\right)^{rt}}{e(g_1, \prod_{i=1}^n g_2^{\kappa_i y_i})^{rt}} \\
&= m \cdot \frac{e\left(g_1^{\kappa_0 \langle \boldsymbol{y}, \boldsymbol{x} \rangle + \sum_{i=1}^n \kappa_i y_i}, g_2\right)^{rt}}{e(g_1, g_2^{\sum_{i=1}^n \kappa_i y_i})^{rt}} \\
&= m \cdot \frac{e(g_1, g_2)^{rt \kappa_0 \langle \boldsymbol{y}, \boldsymbol{x} \rangle + rt \sum_{i=1}^n \kappa_i y_i}}{e(g_1, g_2)^{rt \sum_{i=1}^n \kappa_i y_i}} \\
&= m \cdot e(g_1, g_2)^{rt \kappa_0 \langle \boldsymbol{y}, \boldsymbol{x} \rangle} \qquad (1)
\end{aligned}
$$

Clearly, if $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = 0$, then Equation 1 is equal to m. On the other hand, if $\langle \boldsymbol{y}, \boldsymbol{x} \rangle \neq 0$ then Equation 1 would only be equal to m if $rt\kappa_0 \langle \boldsymbol{y}, \boldsymbol{x} \rangle$ happens to be equal to 0 which will happen only with negligible probability.

Theorem 5. *If the DBDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the zero inner product encryption scheme* ZIPE *shown in Figure 9 is selectively payload-hiding.*

Proof. Let $\mathcal{A}$ be an algorithm that breaks the selective payload-hiding property of ZIPE with probability $1/2 + \epsilon(\lambda)$. We then construct an algorithm $\mathcal{B}$ distinguishing the two cases of the DBDH problem with an advantage of $\epsilon(\lambda)$ as depicted in Figure 10. It is easy to see, that the public key generated by $\mathcal{B}$,

$\mathcal{B}(g_1, A_1, C_1, g_2, A_2, B_2, Z)$	$\mathsf{KGen}'_{\mathsf{ZIPE}}(\boldsymbol{y})$
$b \leftarrow_\$ \{0,1\}$	$\rho \leftarrow_\$ \mathbb{Z}_p$
$\boldsymbol{x} \leftarrow \mathcal{A}(1^\lambda)$; $\boldsymbol{\omega} \leftarrow_\$ \mathbb{Z}_p^n$	$d_0 := B_2^{-\frac{\langle \boldsymbol{\omega}, \boldsymbol{y} \rangle}{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}} \cdot A_2^{r\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g_2^{r\langle \boldsymbol{\omega}, \boldsymbol{y} \rangle}$
$\mathsf{ek} := (g_1, B_2, A_1, g_1^{\omega_1} A_1^{x_1}, \ldots, g_1^{\omega_n} A_1^{x_n})$	$d_1 := g_2^r \cdot B_2^{-\frac{1}{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}}$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KGen}'_{\mathsf{ZIPE}}(\cdot)}(\mathsf{ek})$; $\kappa' \leftarrow_\$ \mathbb{Z}_p$	**return** $\mathsf{dk} := (d_0, d_1, \boldsymbol{y})$
$c^* := (Z \cdot m_b, C_1, C_1^{x_1\kappa' + \omega_1}, \ldots, C_1^{x_n\kappa' + \omega_n})$	
$b' \leftarrow \mathcal{A}(c^*)$	
if $b' = b$ **then return** 1	
else return 0	

Fig. 10: The reduction $\mathcal{B}$ from the selective payload-hiding of ZIPE to DBDH assumption.

$$a_1 = g_1^\alpha \quad \text{and} \quad h_i = g_1^{\omega_i + \alpha x_i}$$

is correctly distributed. By the definition of DBDH α is uniformly distributed in $\mathbb{Z}_p$ and, since ω_i is a uniformly random value, $\kappa_i = \omega_i + \alpha x_i$ is also uniformly distributed. Further, we argue that the secret keys output by $\mathsf{KGen}'_{\mathsf{ZIPE}}(\boldsymbol{y})$ are correctly distributed. We have

$$\begin{aligned} d_0 = B_2^{-\frac{\langle \boldsymbol{\omega}, \boldsymbol{y} \rangle}{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}} \cdot A_2^{\rho\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g_2^{\rho\langle \boldsymbol{\omega}, \boldsymbol{y} \rangle} &= g_2^{-\beta \frac{\sum_{i=1}^n \omega_i y_i}{\sum_{i=1}^n x_i y_i} + \rho(\alpha \sum_{i=1}^n x_i y_i + \sum_{i=1}^n \omega_i y_i)} \\ &= g_2^{\alpha\beta - \beta\left(\frac{\sum_{i=1}^n \omega_i y_i}{\sum_{i=1}^n x_i y_i} + \alpha\right) + \rho(\alpha \sum_{i=1}^n x_i y_i + \sum_{i=1}^n \omega_i y_i)} \\ &= g_2^{\alpha\beta + \left(\rho - \frac{\beta}{\sum_{i=1}^n x_i y_i}\right)\left(\sum_{i=1}^n (\omega_i + \alpha x_i) y_i\right)} = B_2^\alpha \cdot B_2^{\left(\frac{\rho}{\beta} - \frac{1}{\sum_{i=1}^n x_i y_i}\right)\left(\sum_{i=1}^n \kappa_i y_i\right)} \end{aligned}$$

and

$$d_1 = g_2^r \cdot B_2^{-\frac{1}{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}} = B_2^{\frac{\rho}{\beta} - \frac{1}{\sum_{i=1}^n x_i y_i}}.$$

Clearly (d_0, d_1) are a correct decryption key under ek with randomness $r = \frac{\rho}{\beta} - \frac{1}{\sum_{i=1}^n x_i y_i}$. This randomness is uniformly distributed, because r is chosen independently and uniformly at random. Finally, the challenge ciphertext is $c_0 = Z \cdot m_b$, $c'_0 = C_1 = g_1^\gamma$, and

$$c_i = C_1^{x_i\kappa' + \omega_i} = (g_1^{\kappa' x_i + \omega_i})^\gamma = (g_1^{(\kappa' - \alpha)x_i + \omega_i + \alpha x_i})^\gamma = (g_1^{(\kappa' - \alpha)x_i} h_i)^\gamma.$$

If it holds that $Z = e(g_1, g_2)^{\alpha\beta\gamma} = e(g_1^\alpha, B_1)^\gamma$, then c^* clearly is a correctly distributed ciphertext for message m_b and randomnesses $t = \gamma$ and $\kappa_0 = \kappa' - \alpha$. Thus, $\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_1^\alpha, g_1^\gamma, g_2, g_2^\alpha, g_2^\beta, e(g_1, g_2)^{\alpha\gamma\beta}\right)\right] =$

$1/2 + \epsilon(\lambda)$. On the other hand, if $Z = e(g_1, g_2)^\zeta$ for an independent random ζ, then $c_0 = Z \cdot m_b$ information theoretically hides m_b and $\mathcal{A}$ can therefore only guess. Thus, $\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_1^\alpha, g_1^\gamma, g_2, g_2^\alpha, g_2^\beta, e(g_1, g_2)^\zeta\right)\right] = 1/2$. Combining the two cases, we get

$$\left| \begin{array}{l} \mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_1^\alpha, g_1^\gamma, g_2, g_2^\alpha, g_2^\beta, e(g_1, g_2)^{\alpha\gamma\beta}\right)\right] \\ - \quad \mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_1^\alpha, g_1^\gamma, g_2, g_2^\alpha, g_2^\beta, e(g_1, g_2)^\zeta\right)\right] \end{array} \right| = \epsilon(\lambda)$$

as claimed. Since DBDH holds in $(\mathbb{G}_1, \mathbb{G}_2)$, $\epsilon(\lambda)$ must therefore be negligible and ZIPE is thus selectively payload-hiding. □

Theorem 6. *If the XDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the zero inner product encryption scheme* ZIPE *shown in Figure 9 is statically attribute-hiding.*

Proof. Let $\mathcal{A}$ be an algorithm that breaks the static attribute-hiding property of ZIPE with probability $1/2 + \epsilon(\lambda)$. We then construct an algorithm $\mathcal{B}$ distinguishing the two cases of the XDH problem with an advantage of $\epsilon(\lambda)$ as depicted in Figure 11. Note that, for simplicity, we consider a modified version of the game where the challenger always encrypts m_0. The simulation is however indistinguishable from the original experiment since the scheme is selectively payload-hiding (see proof above).

$\mathcal{B}(g_1, g_2, A, B, C)$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\boldsymbol{x^0}, \boldsymbol{x^1}, \boldsymbol{y}) \leftarrow \mathcal{A}(1^\lambda);\ \boldsymbol{\omega} \leftarrow \mathbb{Z}_p^n;\ (a, r) \leftarrow\!\!\$\ \mathbb{Z}_p^2$
if $\langle \boldsymbol{x^1}, \boldsymbol{y} \rangle = 0$ **then** $u := 1$
else $u := \dfrac{\langle \boldsymbol{x^0}, \boldsymbol{y} \rangle}{\langle \boldsymbol{x^1}, \boldsymbol{y} \rangle}$
$\mathsf{ek} := \left(g_1, g_2, g_1^a, A^{x_1^0 - u \cdot x_1^1} \cdot g_1^{\omega_1}, \ldots, A^{x_{0,n} - u \cdot x_{1,n}} \cdot g_1^{\omega_n}\right)$
$\mathsf{dk} := \left(g_2^{a + r\langle \boldsymbol{y}, \boldsymbol{\omega} \rangle}, g_2^r\right)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ek}, \mathsf{dk});\ \kappa' \leftarrow\!\!\$\ \mathbb{Z}_p$
$c^* := \left(e(B, g_2^a) \cdot m_0, B, C^{x_1^b \kappa' + x_1^0 - u x_1^1} B^{\omega_1}, \ldots, C^{x_n^b \kappa' + x_n^0 - u x_n^1} B^{\omega_n}\right)$
$b' \leftarrow \mathcal{A}(c^*)$
if $b' = b$ **then return** 1
else return 0

Fig. 11: The reduction $\mathcal{B}$ from the static attribute-hiding of ZIPE to the XDH assumption.

It is easy to see, that the public encryption key used by $\mathcal{B}$ is correctly distributed. g_1, g_2 and g_1^a are computed exactly as specified in the scheme and the $h_i = A^{x_i^0 - u \cdot x_i^1} \cdot g_1^{\omega_i} = g_1^{\alpha(x_i^0 - u \cdot x_i^1) + \omega_i}$ are correctly distributed values with $\kappa_i = \alpha(x_i^0 - u \cdot x_i^1) + \omega_i$, because the ω_i are chosen independently and uniformly at random. Further, the decryption key is correctly distributed. $d_1 = g_2^r$ is computed exactly as specified by the scheme and it is easily verified that due to the choice of the κ_i the following holds

$$a_2 \cdot \left(\prod_{i=1}^n g_2^{\kappa_i y_i}\right)^r = g_2^a \cdot \left(g_2^{\sum_{i=1}^n \kappa_i y_i}\right)^r = g_2^a \cdot \left(g_2^{\sum_{i=1}^n \left(\alpha(x_i^0 - u \cdot x_i^1) + \omega_i\right) y_i}\right)^r$$

$$= g_2^a \cdot \left(g_2^{\alpha \sum_{i=1}^n (x_i^0 y_i - u \cdot x_i^1 y_i) + \sum_{i=1}^n \omega_i y_i}\right)^r = g_2^a \cdot \left(g_2^{\alpha \underbrace{\left(\langle \boldsymbol{x^0}, \boldsymbol{y} \rangle - u \langle \boldsymbol{x^1}, \boldsymbol{y} \rangle\right)}_{=0 \text{ by choice of } u} + \langle \boldsymbol{y}, \boldsymbol{\omega} \rangle}\right)^r$$

$$= g_2^{a+r\langle \boldsymbol{y},\boldsymbol{\omega}\rangle}$$

and dk is therefore a correctly distributed decryption key for $\boldsymbol{y}$. Finally, for the challenge ciphertext we have $c_0 = e(B, g_2^a) \cdot m_b = e(a_2, g_2)^\beta \cdot m_b$, $c0' = B = g^\beta$. In the case that $C = g_1^{\alpha\beta}$, we further have

$$\begin{aligned} c_i &= C^{x_i^b\kappa' + x_i^0 - ux_i^1} B^{\omega_i} = g_1^{\alpha\beta(x_i^b\kappa' + x_i^0 - ux_i^1) + \beta\omega_i} = (g_1^{\alpha x_i^b \kappa' + \alpha(x_i^0 - ux_i^1) + \omega_i})^\beta \\ &= (g_1^{\alpha x_i^b \kappa' + \kappa_i})^\beta = ((g_1^{\alpha\kappa'})^{x_i^b} h_i)^\beta. \end{aligned}$$

I.e., the challenge ciphertext is a correctly distributed ciphertext for m_b under x^b with randomness $t = \beta$ and $\kappa_0 = \alpha\kappa'$, both of which are uniformly distributed. By assumption we have that

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^{\alpha\beta}\right)\right] = 1/2 + \epsilon(\lambda).$$

On the other hand, let us now consider the case where $C = g^\gamma$ is a uniformly sampled element of $\mathbb{G}_1$. Then, for all i, we have that

$$\begin{aligned} c_i &= C^{x_i^b\kappa' + x_i^0 - ux_i^1} B^{\omega_i} = g_1^{\gamma(x_i^b\kappa' + x_i^0 - ux_i^1)} B^{\omega_i} = g_1^{x_i^b\kappa'\gamma} g_1^{\gamma(x_i^0 - ux_i^1)} B^{\omega_i} \\ &= g_1^{x_i^b\kappa'\gamma + \gamma(x_i^0 - ux_i^1)} B^{\omega_i}. \end{aligned}$$

If we fix $b = 0$:

$$c_i = g_1^{x_i^0\kappa'\gamma + \gamma(x_i^0 - ux_i^1)} B^{\omega_i} = g_1^{x_i^0\gamma(\kappa'+1) - \gamma u x_i^1} B^{\omega_i} = g_1^{x_i^0\gamma' - \frac{\gamma'}{(\kappa'+1)} u x_i^1} B^{\omega_i}.$$

Where the last equality is obtained by defining $\gamma' = \gamma(\kappa' + 1)$ and rearranging the equation. Note that, since γ is uniformly distributed in $\mathbb{Z}_p$, then so is γ'. We now rewrite $\frac{1}{(\kappa'+1)}$ as $(\kappa'' + 1)$. Again it is easy to see that κ'' is a uniformly distributed element of $\mathbb{Z}_p$. Therefore we have

$$c_i = g_1^{x_i^0\gamma' - \gamma'(\kappa''+1)ux_i^1} B^{\omega_i},$$

which implies that, for the case $b = 0$ all of the c_i are distributed identically to the case $b = 1$. This means that the choice of b is information theoretically hidden to the eyes of $\mathcal{A}$. Thus it holds that

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^\gamma\right)\right] = 1/2.$$

Combining the two cases, we get

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^{\alpha\beta}\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^\gamma\right)\right]\right| = \epsilon(\lambda)$$

as claimed. Since XDH holds in $(\mathbb{G}_1, \mathbb{G}_2)$, $\epsilon(\lambda)$ must therefore be negligible and ZIPE is statically attribute-hiding. □

6 Bootstrapping to P/poly

We discuss several bootstrapping techniques to extend the domain of a garbling scheme for inner-products. As previously discussed, the main difference between arithmetic and standard garbling is that the former does not have access to the binary representation of the input x. Instead, the encoding algorithm must operate directly on the algebraic representation of x as a field element. Here we consider two variants of such a model:

1. The encoding algorithm has access to the algebraic representation of the first n powers of the input $(x, x^2, \ldots, x^n) \in \mathbb{F}^n$. This relaxed model is motivated by the scenario when the input is encrypted under a multiplicatively-homomorphic encryption scheme (such as ElGamal [21]).
2. In the (standard) more restrictive version, the encoding algorithm operates only on $x \in \mathbb{F}$.

Depending on which model we consider, we describe a different bootstrapping technique to compile an arithmetic garbling scheme for inner-product predicates to a fully-fledged arithmetic garbling scheme for P/poly. For the latter case, the transformation is limited to circuits with small (poly-size) input domain.

Loose Arithmetic Representation. As already observed by Katz, Sahai, and Waters [32], inner-product predicates are sufficient to encode the evaluation of bounded-degree polynomials: Garbling the evaluation of a polynomial $p(\mathfrak{x}) = c_0 + c_1\mathfrak{x} + \ldots, c_n\mathfrak{x}^n$ is done by evaluating $\mathsf{GC.Garble}(1^\lambda, \boldsymbol{c})$, where $\boldsymbol{c} := (c_0, c_1, \ldots, c_n)$. Then, the encoding of the input x is obtained by running the encoding algorithm on the vector $\boldsymbol{x} := (1, x, x^2, \ldots, x^n)$. Then, the resulting garbled circuit securely evaluates the predicate $\langle \boldsymbol{c}, \boldsymbol{x} \rangle = p(x) = 0$, which corresponds to a polynomial predicate of degree-n.

It is a well known fact that any NC1 circuit can be represented by a polynomial with polynomial degree [26], which immediately implies that our garbling scheme supports log-depth circuits. Since NC1 circuits have a boolean output space, then polynomial predicates suffice. Given this garbling gadget for NC1 circuits, the final step of the transformation consists in applying the Chinese Remainder Theorem-based compiler of Applebaum, Ishai, and, Kushilevitz [6], which yields an arithmetic garbling scheme for P/poly.

Restrictive Arithmetic Representation. The main ingredient of the transformation is a classical projective garbling scheme. Let $\{1, \ldots, n\}$, for some polynomially-bounded n, be the input space. To garble a function f one first runs a classical garbling scheme on the following (boolean) circuit

$\underline{\varGamma(x)}$

parse $x = x_1 \| \ldots \| x_n$
for $i \in \{1 \ldots, n\}$
  if $x_i = 1$ **then** $y := i$
return $f(y)$

and obtains the vector of labels $(X_1^0, X_1^1, \ldots, X_n^0, X_n^1)$ together with the encoded circuit $\tilde{\varGamma}$. Then for each $i \in \{1, \ldots, n\}$ run the arithmetic garbling scheme for inner products for the following predicate

$$\varUpsilon_i(x) := \ x - i = 0$$

using X_i^0 and X_i^1 as the random coins (i.e., the values of r^0 and r^1 in Figure 2). The final garbled circuit consists of the elements $(\tilde{\varGamma}, \tilde{\varUpsilon}_1, \ldots, \tilde{\varUpsilon}_n)$. Inputs are then encoded using the corresponding algorithm $\tilde{X} \leftarrow \mathsf{GC.Enc}(e, x)$, as defined in Figure 2. Given $\tilde{X}$, one can evaluate all the circuits $(\tilde{\varUpsilon}_1, \ldots, \tilde{\varUpsilon}_n)$, which return the set of labels $(X_1^0, \ldots, X_{x-1}^0, X_x^1, X_{x+1}^0, \ldots, X_n^0)$. That is, the evaluator recovers the 0 label on all bits except for the position corresponding to the value x. Such a set of labels constitutes a valid encoding for $\tilde{\varGamma}$ so the evaluator can recover the result of the computation $f(x)$.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (Mar / Apr 2015)
2. Agrawal, S., Bhattacherjee, S., Phan, D.H., Stehlé, D., Yamada, S.: Efficient public trace and revoke from standard assumptions: Extended abstract. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17. pp. 2277–2293. ACM Press (Oct / Nov 2017)
3. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (Aug 2016)
4. Applebaum, B., Avron, J., Brzuska, C.: Arithmetic cryptography: Extended abstract. In: Roughgarden, T. (ed.) ITCS 2015. pp. 143–151. ACM (Jan 2015)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. In: 45th FOCS. pp. 166–175. IEEE Computer Society Press (Oct 2004)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 120–129. IEEE Computer Society Press (Oct 2011)
7. Applebaum, B., Ishai, Y., Kushilevitz, E., Waters, B.: Encoding functions with constant online rate or how to compress garbled circuits keys. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 166–184. Springer, Heidelberg (Aug 2013)

8. Attrapadung, N., Libert, B.: Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 384–402. Springer, Heidelberg (May 2010)
9. Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for boolean and arithmetic circuits. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16. pp. 565–577. ACM Press (Oct 2016)
10. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Heidelberg (Aug 2017)
11. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 12. pp. 784–796. ACM Press (Oct 2012)
12. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 470–491. Springer, Heidelberg (Nov / Dec 2015)
13. Boneh, D.: The decision Diffie-Hellman problem. In: Third Algorithmic Number Theory Symposium (ANTS). LNCS, vol. 1423. Springer, Heidelberg (1998), invited paper
14. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001)
15. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (May 2014)
16. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012)
17. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 97–106. IEEE Computer Society Press (Oct 2011)
18. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-round secure computation and secure autonomous mobile agents. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 512–523. Springer, Heidelberg (Jul 2000)
19. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (Apr 2015)
20. Chen, J., Libert, B., Ramanna, S.C.: Non-zero inner product encryption with short ciphertexts and private keys. In: Zikas, V., De Prisco, R. (eds.) SCN 16. LNCS, vol. 9841, pp. 23–41. Springer, Heidelberg (Aug / Sep 2016)
21. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (Aug 1984)
22. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 469–472 (1985)
23. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013)
24. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (Aug 2010)
25. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (Aug 2013)
26. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (Aug 2012)
27. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (Aug 2015)
28. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., Vimercati, S. (eds.) ACM CCS 06. pp. 89–98. ACM Press (Oct / Nov 2006), available as Cryptology ePrint Archive Report 2006/309
29. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS. pp. 294–304. IEEE Computer Society Press (Nov 2000)
30. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13. pp. 955–966. ACM Press (Nov 2013)
31. Katsumata, S., Yamada, S.: Non-zero inner product encryption schemes from various assumptions: LWE, DDH and DCR. In: PKC 2019, Part II. LNCS, Springer, Heidelberg (2019)

32. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (Apr 2008)
33. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (Jul 2008)
34. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: 31st ACM STOC. pp. 245–254. ACM Press (May 1999)
35. Okamoto, T., Takashima, K.: Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 11. LNCS, vol. 7092, pp. 138–159. Springer, Heidelberg (Dec 2011)
36. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM CCS 07. pp. 195–203. ACM Press (Oct 2007)
37. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (Mar 2012)
38. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005)
39. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 10. pp. 463–472. ACM Press (Oct 2010)
40. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (May 2005)
41. Valiant, L.G.: Universal circuits (preliminary report). In: Proceedings of the eighth annual ACM symposium on Theory of computing. pp. 196–203. ACM (1976)
42. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)

A Obliviousness

We define obliviousness for a garbling scheme in the following.

$\mathsf{ExpObl}^{\mathcal{A}}_{\mathsf{GC},\Phi}(1^\lambda)$

$(C, x) \leftarrow \mathcal{A}(1^\lambda)$
if $x \notin \{0,1\}^n$ **return** $\bot$
$b \leftarrow\!\!\$\ \{0,1\}$
if $b = 0 : (\tilde{C}, e, d) \leftarrow \mathsf{GC.Garble}(1^\lambda, C)$
$\qquad \tilde{X} \leftarrow \mathsf{GC.Enc}(e, x)$
if $b = 1 : (\tilde{C}, \tilde{X}) \leftarrow \mathcal{S}(1^\lambda, \Phi(C))$
$b' \leftarrow \mathcal{A}(\tilde{C}, \tilde{X})$
return $b = b'$

Fig. 12: Obliviousness game.

Definition 13 (Φ-Obliviousness). *Let* $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Enc}, \mathsf{GC.Dec}, \mathsf{GC.Eval}, \mathsf{GC.ev})$ *be a garbling scheme and* Φ *a side information function.* GC *achieves* Φ*-obliviousness if there exists a* PPT *simulator* $\mathcal{S}$ *such that for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$2 \cdot \Pr[\mathsf{ExpObl}^{\mathcal{A}}_{\mathsf{GC},\Phi}(1^\lambda) = 1] - 1 \leq \mathsf{negl}(\lambda),$$

where $\mathsf{ExpObl}^{\mathcal{A}}_{\mathsf{GC},\Phi}(1^\lambda)$ *is defined in Figure 12.*

To argue about obliviousness of our garbling scheme, we must define an additional property for a predicate encryption scheme. In a nutshell, such a property demands that an encryption of a random message is computationally indistinguishable from a failed decryption. We refer to this notion as *oblivious decryption* and we formally define it in the following.

Definition 14 (Oblivious Decryption). *A PE scheme has* oblivious decryption *with respect to $\mathcal{F}$ and Σ if for all* PPT *adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter λ.*

1. *On input 1^λ, $\mathcal{A}$ outputs a pair of attributes $(\mathsf{A}_0, \mathsf{A}_1) \in \Sigma^2$ and a predicate $f \in \mathcal{F}$.*
2. $\mathsf{Setup}(1^\lambda)$ *is run to generate* ek *and* msk *and the adversary is given* ek *and* $\mathsf{KeyGen}(\mathsf{msk}, f)$.
3. *A random b is chosen and a random message $r \leftarrow \mathbb{F}$ is sampled, then $\mathcal{A}$ is given the ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{A}_b, r)$.*
4. *$\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$.*

We can show that the property as defined above suffices to construct an oblivious garbling scheme.

Theorem 7. *Let Φ be the function that takes as input an inner-product predicate and returns the length of its vector. Let* NIPE *be a NIPE scheme with oblivious decryption and let* ZIPE *be a ZIPE scheme scheme with oblivious decryption, then the garbling scheme* GC *as described in Figure 2 achieves Φ-obliviousness.*

Proof. The simulator is identical to the one described in Figure 3 except that the string $\boldsymbol{s}$ is sampled uniformly at random from $\mathbb{F}^{n+1}$ and the decoding information d is set to be the empty string. The indistinguishability argument follows along the same lines, except that the distance between neighbouring experiments is bounded with reductions against the oblivious decryption of ZIPE and NIPE, respectively. □

A.1 Obliviousness of ZIPE

Here we show that our ZIPE scheme has oblivious decryption.

Theorem 8. *If the XDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the zero inner product encryption scheme* ZIPE *shown in Figure 9 has oblivious decryption.*

Proof. Let $\mathcal{A}$ be an algorithm that breaks the oblivious decryption of ZIPE with probability $1/2 + \epsilon(\lambda)$. We then construct an algorithm $\mathcal{B}$ distinguishing the two cases of the XDH problem with an advantage of $\epsilon(\lambda)$ as depicted in Figure 13.

It is easy to show that the public encryption key used by $\mathcal{B}$ is correctly distributed: g_1, g_2 and g_1^a are computed exactly as specified in the scheme and the $h_i = A^{\delta_i} \cdot g_1^{\omega_i}$ are correctly distributed values since ω_i are chosen independently and uniformly at random. Further, we argue that the decryption key is functional Further, the decryption key is correctly distributed. $d_1 = g_2^r$ is computed exactly as specified by the scheme and it is easily verified that the following holds

$$
\begin{aligned}
a_2 \cdot \left(\prod_{i=1}^{n} g_2^{\kappa_i y_i} \right)^r &= g_2^a \cdot \left(g_2^{\sum_{i=1}^n \kappa_i y_i} \right)^r \\
&= g_2^a \cdot \left(g_2^{\sum_{i=1}^{n-1} (\alpha\delta_i + \omega_i) y_i + \left(\alpha\delta' + \omega_n\right) y_n} \right)^r \\
&= g_2^a \cdot \left(g_2^{\sum_{i=1}^{n} \omega_i y_i + \alpha \underbrace{\left(\sum_{i=1}^{n-1} \delta_i y_i + \delta' y_n\right)}_{=0 \text{ by choice of } \delta'}} \right)^r \\
&= g_2^{a + r\langle \boldsymbol{y}, \boldsymbol{\omega} \rangle}
\end{aligned}
$$

and dk_w is therefore a correctly distributed decryption key for w. In the case that (g_1, g_2, A, B, C) is a well-formed DDH tuple (i.e., $A = g_1^\alpha, B = g_1^\beta, C = g_1^{\alpha \cdot \beta}$) then we argue that c^* is a well-formed ciphertext for a

$\mathcal{B}(g_1, g_2, A, B, C)$

$b \leftarrow\!\!\$\ \{0,1\}$

$(\boldsymbol{x}^{\mathbf{0}}, \boldsymbol{x}^{\mathbf{1}}, \boldsymbol{y}) \leftarrow \mathcal{A}(1^\lambda);\ \boldsymbol{\delta} \leftarrow\!\!\$\ \mathbb{Z}_p^{n-1};\ \boldsymbol{\omega} \leftarrow\!\!\$\ \mathbb{Z}_p^n;\ (a, r, s) \leftarrow\!\!\$\ \mathbb{Z}_p^3;\ T \leftarrow\!\!\$\ \mathbb{G}_T$

$\delta' := \dfrac{-\langle \boldsymbol{\delta}, (y_1, \ldots, y_{n-1}) \rangle}{y_n}$

$\mathsf{ek} := \left(g_1, g_2, g_1^a, A^{\delta_1} \cdot g_1^{\omega_1}, \ldots, A^{\delta_{n-1}} \cdot g_1^{\omega_{n-1}}, A^{\delta'} \cdot g_1^{\omega_n}\right)$

$\mathsf{dk}_w := (g_2^{a + r\langle \boldsymbol{y}, \boldsymbol{\omega} \rangle}, g_2^r)$

$c^* := \left(T, B, C^{\delta_1} B^{\omega_1 + s x_1^b}, \ldots, C^{\delta_{n-1}} B^{\omega_{n-1} + s x_{n-1}^b}, C^{\delta'} B^{\omega_n + s x_n^b}\right)$

$b' \leftarrow \mathcal{A}(\mathsf{ek}, \mathsf{dk}_w, c^*)$

if $b' = b$ **then return** 1

else return 0

Fig. 13: The reduction $\mathcal{B}$ from the oblivious decryption of ZIPE to the XDH assumption.

random message m: Observe that $T = e(B, g_2^a) \cdot m = e(a_2, g_2)^\beta \cdot m$, for a certain m. Since T is randomly sampled form $\mathbb{G}_T$, then so is m. Clearly $B = g_1^\beta$, and for all $i \in \{1, \ldots, n-1\}$ we have that

$$c_i = C^{\delta_i} B^{\omega_i + s x_i^b} = g_1^{\alpha\beta\delta_i} g_1^{\beta\left(\omega_i + s x_i^b\right)} = g_1^{\beta(\alpha\delta_i + \omega_i) + \beta s x_i^b} = \left(g_1^{\alpha\delta_i + \omega_i}\right)^\beta \left(g_1^{s x_i^b}\right)^\beta$$
$$= \left(h_i h_0^{x_i^b}\right)^\beta.$$

where $h_0 = g_1^s$, for some random $s \in \mathbb{Z}_p$. A similar argument can be used to show that $c_n = C^{\delta'} B^{\omega_n + s x_n^b}$ is correctly distributed. It follows that the challenge ciphertext is well formed for a random m, under $\boldsymbol{x}^b$ with randomness $t = \beta$ and $\kappa_0 = s$, which are uniformly sampled. Therefore by initial assumption we have that $\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^{\alpha\beta}\right)\right] = 1/2 + \epsilon(\lambda)$. We now turn to the case where $C = g_1^\gamma$ is a uniformly sampled element of $\mathbb{G}_1$. In this case the observation is that c^* is exclusively composed by random group elements: Clearly T and B are two random elements of $\mathbb{G}_T$ and $\mathbb{G}_1$, respectively. Further, for all $i \in \{1, \ldots, n-1\}$, the value of $c_i = C^{\delta_i} B^{\omega_i + s x_i^b}$ is uniformly distributed in $\mathbb{G}_1$ since δ_i is randomly sampled from $\mathbb{Z}_p$. Finally $c_n = C^{\delta'} B^{\omega_n + s x_n^b}$ is also uniformly distributed in $\mathbb{G}_1$ since C is a random element of $\mathbb{G}_1$. Therefore the value of b is information theoretically hidden to the eyes of the adversary. Thus, we have that $\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^\gamma\right)\right] = 1/2$. Combining the two cases, we get

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^{\alpha\beta}\right)\right] - \mathsf{Pr}\left[1 \leftarrow \mathcal{B}\left(g_1, g_2, g_1^\alpha, g_1^\beta, g_1^\gamma\right)\right]\right| = \epsilon(\lambda)$$

as claimed. Since XDH holds in $(\mathbb{G}_1, \mathbb{G}_2)$, $\epsilon(\lambda)$ must be negligible and therefore ZIPE has oblivious decryption. □

A.2 Obliviousness of NIPE

The last barrier towards efficiently instantiating the garbling scheme in Figure 2 with our NIPE construction is the fact that the scheme, as described in Figure 6, does not have oblivious decryption. Consider a cipher $c = (c_1, c_2) = (\mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, mr \cdot \boldsymbol{x}), \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{ek}, r \cdot \boldsymbol{x}))$ of our scheme, for some attribute $\boldsymbol{x}$. Whenever key $\mathsf{dk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}, \boldsymbol{y})$ corresponding the predicate $\boldsymbol{y}$ cannot decrypt c, then we have that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. By the correctness of the IPFE it follows that $(\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{dk}_{\boldsymbol{y}}, c_1), \mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{dk}_{\boldsymbol{y}}, c_2)) = (\langle mr \cdot \boldsymbol{x}, \boldsymbol{y} \rangle, \langle r \cdot \boldsymbol{x}, \boldsymbol{y} \rangle) = (mr \cdot \langle \boldsymbol{x}, \boldsymbol{y} \rangle, r \cdot \langle \boldsymbol{x}, \boldsymbol{y} \rangle) = (0, 0)$, which is efficiently distinguishable from the successful decryption of a random

message. It follows that a blackbox instantiation of our garbling scheme (Figure 2) would achieve privacy and authenticity but not obliviousness.

In Figure 14 we show how to modify the NIPE proposed in Section 4 to achieve oblivious decryption. The resulting scheme will no longer be correct, however it can still be used to implement the garbling scheme (Figure 2) with a minor modification of the decoding procedure.

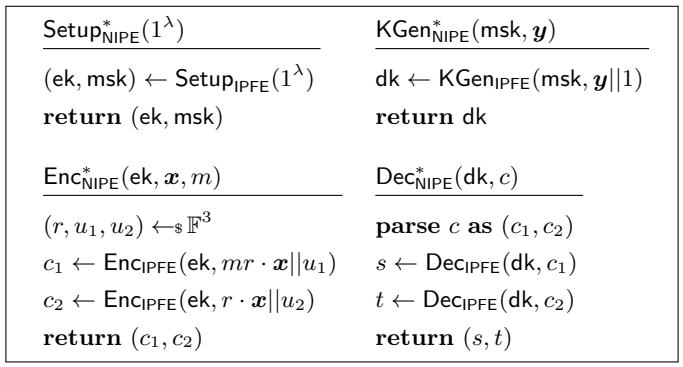

Fig. 14: Modified non-zero inner product encryption.

The garbling scheme in Figure 2 is then modified as follows: The extra randomness (u_1, u_2) sampled in the encryption algorithm $\mathsf{Enc}^*_{\mathsf{NIPE}}$ is also included in the decoding information d, which now consists of the tuples (r_0, r_1) and (u_1, u_2). Upon receiving $\tilde{Y} = (\tilde{r}_0, \tilde{r}_1)$, the decoding procedure checks whether $\tilde{r}_0 = r_0$ and returns 0 if this is the case. Else it parses $\tilde{r}_1$ as (s, t) and returns 1 if $r_1 = \frac{s-u_1}{t-u_2}$ holds. In case none of the checks verifies, the decoding algorithm returns $\perp$.

It is an easy exercise to show that the garbling scheme still achieves privacy, authenticity, and obliviousness under the assumption that the ZIPE scheme ZIPE and the modified NIPE scheme NIPE^* achieve static attribute-hiding and oblivious decryption. The only non-trivial part of the argument is to show that the scheme in Figure 14 has oblivious decryption.

Theorem 9. *Let* IPFE *be a semantically secure inner product functional encryption scheme, then the non-zero inner product encryption scheme* NIPE^* *shown in Figure 14 has oblivious decryption.*

Proof. Let $\mathcal{A}$ be an algorithm that breaks the oblivious decryption of NIPE with probability $1/2 + \epsilon(\lambda)$. We then construct an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ breaking the semantic security of IPFE with the same probability as depicted in Figure 15. Note, that we use a notion of semantic security where $\mathcal{B}$ can output pairs of messages.

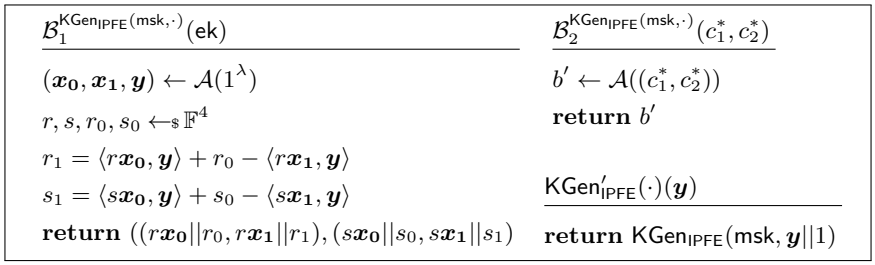

Fig. 15: The reduction $\mathcal{B}$ from the oblivious decryption of NIPE to the semantic security of IPFE.

This notion follows from standard semantic security with a simple hybrid argument.

Clearly, the public key and the decryption key passed to $\mathcal{A}$ are distributed exactly as specified by the scheme. Further, the attacker can only ask for a single decryption key and must do so before it outputs the challenge. Note that we can rewrite $r = m \cdot s$, for some $m \in \mathbb{F}$, since s and r are uniformly distributed over $\mathbb{F}$, then so is m. It follows that, for both choices of b, the challenge ciphertexts are correctly distributed to the eyes of the adversary.

What is left to be shown is that the advantage of $\mathcal{A}$ carries over to the distinguisher, in particular we need to show that the challenges of $\mathcal{B}$ are legit, i.e., $\langle r\boldsymbol{x_0}||r_0, \boldsymbol{y}||1\rangle = \langle r\boldsymbol{x_1}||r_1, \boldsymbol{y}||1\rangle$ and $\langle s\boldsymbol{x_0}||s_0, \boldsymbol{y}||1\rangle = \langle s\boldsymbol{x_1}||s_1, \boldsymbol{y}||1\rangle$. This easily follows from the fact that

$$\begin{aligned}\langle r\boldsymbol{x_1}||r_1, \boldsymbol{y}||1\rangle &= \langle r\boldsymbol{x_1}, \boldsymbol{y}\rangle + r_1 \\ &= \langle r\boldsymbol{x_1}, \boldsymbol{y}\rangle + \langle r\boldsymbol{x_0}, \boldsymbol{y}\rangle + r_0 - \langle r\boldsymbol{x_1}, \boldsymbol{y}\rangle \\ &= \langle r\boldsymbol{x_0}, \boldsymbol{y}\rangle + r_0.\end{aligned}$$

A similar argument holds for the second ciphertext. It follows that whenever $\mathcal{A}$ is successful, so is $\mathcal{B}$. Therefore $\mathcal{B}$ is successful with probability $1/2 + \epsilon(\lambda)$. Since IPFE is semantically secure, $\epsilon(\lambda)$ must be negligible and NIPE has oblivious decryption. □