Witness Encryption from Instance Independent Assumptions

Craig Gentry
IBM Research, T.J. Watson
cbgentry@us.ibm.com

Allison Bishop Lewko
Columbia University
alewko@cs.columbia.edu

Brent Waters
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

Witness encryption was proposed by Garg, Gentry, Sahai, and Waters as a means to encrypt to an instance, x, of an NP language and produce a ciphertext. In such a system, any decryptor that knows of a witness w that x is in the language can decrypt the ciphertext and learn the message. In addition to proposing the concept, their work provided a candidate for a witness encryption scheme built using multilinear encodings. However, one significant limitation of the work is that the candidate had no proof of security (other than essentially assuming the scheme secure).

In this work we provide a proof framework for proving witness encryption schemes secure under instance independent assumptions. At the highest level we introduce the abstraction of *positional witness encryption* which allows a proof reduction of a witness encryption scheme via a sequence of 2^n hybrid experiments where n is the witness length of the NP-statement. Each hybrid step proceeds by looking at *a single witness candidate* and using the fact that it does not satisfy the NP-relation to move the proof forward. We show that this "isolation strategy" enables one to create a witness encryption system that is provably secure from assumptions that are (maximally) independent of any particular encryption instance. We demonstrate the viability of our approach by implementing this strategy using level n-linear encodings where n is the witness length. Our complexity assumption has $\approx n$ group elements, but does not otherwise depend on the NP-instance x.

1 Introduction

Witness encryption, as introduced by Garg, Gentry, Sahai, and Waters [15], is a primitive that allows one to encrypt to an instance of an **NP** language L. An encryptor will take in an instance x along with a message m and run the encryption algorithm to produce a ciphertext CT. Later a user will be able to decrypt the ciphertext and recover m if they know of a witness w showing that x is in the language L according to some witness relation $R(\cdot,\cdot)$. The security of witness encryption states that, for any ciphertext created for an instance x that *is not in the language* L, it must be hard to distinguish whether the ciphertext encrypts m_0 or m_1.

The primitive of encrypting to an instance is intriguing in its own right, and Garg et. al. show that it has many compelling applications, including public key encryption with very fast key generation, identity-based encryption [27, 3, 8], attribute-based encryption [25] (ABE) for arbitrary circuits, and ABE for Turing Machines [16]. The work of [16] goes on to develop even further applications, such as reusable garbling schemes for Turing machines.

These powerful applications motivate the quest for constructions of witness encryption with strong provable security guarantees. In [15], they gave a witness encryption construction for the **NP**-complete Exact Cover problem [18] using multilinear encodings (first suggested in [5] and first constructed by Garg, Gentry, and Halevi [12], with an alternative construction later provided by Coron, Lepoint and Tibouchi [9]).

While the GGSW construction candidate demonstrates the plausibility of realizing secure witness encryption, they were unable to reduce the security of their system to anything simpler than directly assuming the security of their construction. Instead they applied what we will call an *instance dependent* family of assumptions that they called the "Decision Graded Encoding No-Exact-Cover Problem." The assumption is that for each instance x not in the language, no PPT attacker can distinguish between two particular distributions of multilinear encodings. The distributions directly embed the Exact Cover instance x are almost identical to the structure of the ciphertexts from the construction.

The Importance and Difficulty of Using Instance-Independent Assumptions While a generic group argument might give some confidence that it will be difficult to find an attack on a scheme, a reduction to an assumption simpler than the scheme itself is much more desirable. First, such a reduction will often provide critical insight and understanding into why the scheme is secure. Second, the ideas behind proof reductions often transcend their original settings and will be of use elsewhere. Having a single, concrete assumption also provides a clearer focus for cryptanalysis efforts to stress-test a candidate scheme.

Prior to this work, no known schemes could be reduced to instance-independent assumptions. This is also the case for all known indistinguishability obfuscation schemes. For example, [13] explicitly reduces to a instance-dependent family of assumption, while [24] implicitly does this through a meta-assumption.

Our goal is to create techniques for building witness encryption systems that are provably secure under instance independent assumptions. To achieve this, we must first confront an intuitive barrier that is formalized as an impossibility result in [15] (with some restrictions). The idea is that any black-box security reduction to an instance-independent assumption for a witness encryption scheme must (in some sense) verify that a statement is false. Otherwise, we could use the reduction to break the assumption by "fooling it" on a true statement for which we know a witness, and hence can simulate an attack. Since the best known methods for solving **NP-hard** problems take exponential time, this implies an instance independent reduction will have an exponential security loss.

Our Strategy To address the barrier above, we devise a proof technique that employs a reduction which gradually "learns" that the instance x is not in the language. Consider a instance $x \notin L$ with witness candidates of n bits. Our strategy is to allow a reduction to build a hybrid argument by isolating and examining each witness candidate, w, in sequence and utilizing the fact that $R(x, w) = 0$ (i.e. the witness is not valid) to progress the hybrid to the next step. (Since there are 2^n witness candidates, the proof strategy will inherently use complexity leveraging, as will any reduction strategy that falls within the confines of the [15] impossibility result.) In this way, we obtain a "true reduction" that represents a new understanding of the security of witness encryption.

To implement this hybrid approach, we will need a technique that somehow allows a proof to compactly "save" its work for all of the witnesses it has examined. Our starting point will be a broadcast encryption (BE) [10] system proposed by Boneh and Waters [6] in 2006, which was the first collusion resistant system to be proved adaptively secure. Instead of proving security all at once, they employed a method of altering the challenge ciphertext over a sequence of N hybrid experiments for a BE system of N users. At the center of their approach was a new abstraction that they called augmented broadcast encryption. An augmented BE system has an encryption algorithm $\mathit{Encrypt}_{\mathrm{AugBE}}(\mathrm{PK}, S, t, m)$ that takes as input a public key PK, a set of user indices $S \subseteq [0, N-1]$, an index $t \in [0, N]$, and a message m. This produces a ciphertext CT. The semantics of the system are that a user with index $u \in [0, N-1]$ [1] can decrypt the ciphertext and learn the message only if $u \in S$ *and* $u \geq t$. These are like the semantics of standard broadcast encryption, but with the added constraint of the index t. Augmented broadcast encryption has two security properties. The first is that no poly-time attack can distinguish between $\mathit{Encrypt}_{\mathrm{AugBE}}(\mathrm{PK}, S, t, m)$ and $\mathit{Encrypt}_{\mathrm{AugBE}}(\mathrm{PK}, S, t+1, m)$ if the attacker does not have the key for index t or if $t \notin S$. The second property is that the scheme is semantically secure if we encrypt to index $t = N$, thus cutting off all the user keys whether or not they are in S.

It is straightforward to make a standard broadcast encryption using an augmented one, as we can create a broadcast ciphertext to the set S by simply calling $\mathit{Encrypt}_{\mathrm{AugBE}}(\mathrm{PK}, S, t = 0, m)$. By setting $t = 0$, the range condition is never invoked. The advantage of using this condition comes into the proof where we want to prove that no attack algorithm can distinguish an encryption to an adaptively chosen set S^* (meaning it is chosen after seeing the public key) if it is only given keys for $u \notin S^*$. The proof proceeds by a sequence of indistinguishable hybrid experiments where at the i-th hybrid the challenge ciphertext is generated for index $t = i$. Finally, we move to $t = N$ and the second property then implies security of the scheme. Even though there were N indices, the abstraction and hybrid sequence allowed for a proof to isolate one user at a time. The BW construction melded a broadcast system with the Boneh-Sahai-Waters [4] traitor tracing [7] system to enforce the range condition.

Positional Witness Encryption With these concepts in mind, we can turn back to the problem of devising a proof strategy for witness encryption for an **NP**-complete language L. The first step we take is the introduction of a primitive that we call positional witness encryption. A positional

[1] The Boneh-Waters paper uses indices $1, \ldots, N$ for the users. We shift this to $0, \ldots, N-1$ to better match our exposition.

witness encryption system has an encryption algorithm $\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t, m)$ that takes as input a security parameter 1^λ, astring x, a position index $t \in [0, 2^n]$, and a message m and outputs a ciphertext CT. Here we let n be the witness length of x and let $N = 2^n$. One can decrypt a ciphertext by producing a witness w such that $R(x, w) = 1$ *and* $w \geq t$ where w is interpreted as an integer. Essentially, this has the same correctness semantics as standard witness encryption, but with the range condition added. The security properties are as follows:

- Positional Indistinguishability: If $R(x, t) = 0$ then no poly-time attacker can distinguish between $\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t, m)$ and $\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t + 1, m)$.

- Message Indistinguishability: No poly-time attacker can distinguish between $\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t = 2^n, m_0)$ and $\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t = 2^n, m_1)$ for all equal length messages m_0, m_1.

We point out that the security definition of positional witness encryption is not explicitly constrained to $x \notin L$ in any place. However, if some $x \notin L$, then for all witnesses $w \in [0, 2^n - 1]$ (interpreting the bitstring w as an integer) we have that $R(x, w) = 0$. This leads to a natural construction and proof strategy for witness encryption. To witness encrypt a message m to an instance x, we call $\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t = 0, m)$. To prove security, we design a sequence of indistinguishable hybrids where we increase the value of t at each step until we get to $t = N = 2^n$ where we can invoke message indistinguishability. Each step can be made since $x \notin L$ implies $R(x, w) = 0$. The hybrids cause a 2^n loss of security relative to the security of the positional witness encryption and this should be compensated for in setting the security parameter. [2]

The potential advantage of positional witness encryption is that it offers a hybrid strategy where the core security property is focused on whether a *single* witness satisfies a relation. However, there is still a very large gap between imagining this primitive and realizing it. First, we need a data structure that can both securely hide t and compactly store it (e.g. ciphertexts cannot grow proportional to the number of witnesses $N = 2^n$). Next, we need to be able to somehow embed an instance x of an **NP**-complete problem. This must be done in such a way that the security proof can isolate a property that depends on whether $R(x, w) = 0$ for each witness candidate w and use this to increment the positional data in a manner oblivious to an attacker.

Tribes Schemes and Their Uses We begin our realization by introducing a data structure that we call a tribes matrix, which will be flexible enough to encode both a position and a CNF formula. A tribes matrix will induce a boolean function from n-bit inputs to a single output bit. We then introduce a cryptographic primitive called a tribes scheme that will hide some properties of the matrix while still enabling evaluation of the corresponding boolean function. The benefit of this middle layer of abstraction is that it portions the work into a manageable hybrid security proof at the abstract level and creates a rather slim and concise target for lower level instantiations. This naturally increases the potential for instantiating our framework with a variety of different assumptions.

The name "tribes" was chosen because of the structural similarities between the induced boolean function and the tribes function commonly considered in boolean function analysis (e.g. [2]). In

[2]We note that complexity leveraging is used elsewhere in "computing on encrypted data". For example, current solutions of Attribute-Based Encryption for circuits [14, 17] are naturally selectively secure and require complexity leveraging to achieve adaptive security.

the tribes function, n inputs are thought of as people that are partitioned into ℓ tribes, and the function outputs 1 if and only if at least one tribe takes value 1 unanimously. In our case, we define 3-dimensional $n \times \ell \times 2$ matrix, where we think of it as having n rows, ℓ columns, and 2 "slots" for each row and column pair. The slots take values from a 2-symbol alphabet, notated as $\{U, B\}$ and are $\{0, 1\}$-indexed. The B stands for "blocked" and the U stands for "unblocked." To evaluate the boolean function on a n bit input $x_1, \ldots, x_n$, we consider each of the ℓ columns as a tribe, but in each row i we take the value in the slot indexed by x_i (this means that the input bits specify the composition of the tribe from a pre-existing set of values, rather than providing the values themselves). If some tribe is unanimously "blocked," the function outputs 1, otherwise it outputs 0. For a tribes matrix denoted by M, we will denote the associated boolean function by f_M.

When we embed a tribes matrix into a tribes scheme, we seek to allow access to evaluating the function without revealing full information about the matrix entries. Of course, some properties of the matrix entries can be inferred from black-box access to the function, and this is fine; we only seek to hide a very specific kind of structure that does not affect the function evaluation on any input. For this, we define the notions of "inter-column" and "intra-column" security, and the combination essentially requires that the tribes schemes for two matrices that differ in a single slot value are indistinguishable if there a simple reason why this slot value does not affect the boolean function.

More precisely, suppose we wish to hide the value of a slot in row i^*, column k. If there is a column j such that the corresponding slot has value B, and furthermore in all rows $i \neq i^*$, occurrences of B in column k are always matched by occurrences of B in column j, then we can change the value of this slot in row i^* without affecting the boolean function. To see this, observe that regardless of the value at this slot in column k, for any input where tribe k is unanimously blocked, tribe j is also unanimously blocked. Inter-column security requires that we can furthermore hide this change in the sense of computational indistinguishability. The second property of intra-column security ensures that if both slots of a particular row in a single column have the value U, then any other slot in the column can be changed without an attacker noticing.

We next consider how one might encode positions and CNF formulas into a tribes matrix. Our approach is to encode these two objects separately, and then simply concatenate the matrices. To encode a position t, we wish to produce a tribes matrix M where the boolean function f_M will output 1 for every witness $y < t$, and will output 0 otherwise. The key observation is that every potential witness $y < t$ will have some bit j where it first departs from t (starting from the most significant bit), and in this bit y will be 0 and t will be 1. We leverage this by designing the j^{th} column of M to be blocked precisely for such y.

To encode a CNF formula in a tribes matrix, we build a column corresponding to each clause, where the rows are indexed by the variables, $x_1, \ldots, x_n$. To fill in the slots of row i in column j, we see if the literal x_i or its negation $\overline{x_i}$ appear in the j^{th} clause. If x_i appears, we put a U in the 1-indexed slot. If $\overline{x_i}$ appears, we put a U in the 0-indexed slot. For any remaining slots, we put B. This yields a column that is blocked precisely for inputs that do not satisfy the clause. We therefore get a matrix whose associated boolean function outputs 1 if and only if the CNF formula is unsatisfied.

From this, we can construct a positional witness encryption scheme. To encrypt a message bit to a particular position and formula, the encryptor forms tribes matrices as above, concatenates them, and concatenates one extra column to encode the message (it will contain all U's if the bit is 0 and all B's if the bit is one). It finally embeds this matrix in a tribes scheme, which serves as the ciphertext. A decryptor can then evaluate the boolean function to recover the message. (If $R(x, w) = 1$ and $w \geq t$, then the output of the tribes evaluation will reflect the message; otherwise, it outputs 1 regardless of the message.)

To prove positional indistinguishability, we proceed through a hybrid argument that relies upon the inter-column security of the cryptographic tribes scheme to incrementally change the matrix entries to an encoding of the next position. During this process, we will need to leverage the fact that the current position represents a witness that *does not satisfy* the CNF formula. The key observation here is that there will be at least one clause that is not satisfied, and the column for that clause can be used to make changes to entries in another column through the inter-column security game.

Instantiating a Tribes Scheme Finally, what is left is to instantiate a tribes scheme and reduce the inter-column and intra-column security requirements to a computational assumption. We give three related constructions each from multilinear algebraic groups with an n linear map, which required for a tribes instance of n rows.

Our first instantiation (in Section 5) uses *composite* order symmetric multilinear groups. The group's order is a product $n + \ell$ primes for an n by ℓ tribes matrix. Our next instantiation (in Section 6) utilizes asymmetric groups to reduce the order of the group to the product of ℓ primes. In Section 7, we modify the instantiation to be in prime order using a translation based on eigenvectors. Each instance is based on a pair of multilinear map assumptions that depend on n (or n and ℓ), but are independent of the contents of the tribes matrix. The assumptions we use in the composite order symmetric context for example, are given in Section 5, and we call them the multilinear subgroup decision and multilinear subgroup elimination assumptions, as they are rather natural variants of subgroup decision assumptions typically used in bilinear groups. In Appendix C, we justify the prime order variants of our assumptions in the multilinear generic group model. In Appendix B, we show how to translate from algebraic groups into the multilinear encodings of Coron, Lepoint and Tibouchi (CLT) [9].

2 Positional Witness Encryption

We will first give our definition of a positional witness encryption system and then show how it implies standard witness encryption by a hybrid argument.

We define a *positional witness encryption* scheme for an **NP** language L. Let $R(\cdot, \cdot)$ be the corresponding witness relation and let $n = n(|x|)$ be the witness length for a particular witness x. The system consists of two algorithms:

> ***Encryption.*** The algorithm $Encrypt_{\mathrm{PWE}}(1^\lambda, x, t, m)$ takes as input a security parameter 1^λ, an unbounded-length string x, a position index $t \in [0, 2^n]$ (we let $n = n(x)$) and a message $m \in \mathcal{M}$ for some (fixed and finite) message space $\mathcal{M}$, and outputs a ciphertext CT.
>
> ***Decryption.*** The algorithm $Decrypt_{\mathrm{PWE}}(\mathrm{CT}, w)$ takes as input a ciphertext CT and a length n string w, and outputs a message m or the symbol $\perp$. (We assume the ciphertext specifies the instance x and therefore $n = n(|x|)$ is known.)

Given a string $w \in \{0, 1\}^n$ we will sometimes slightly abuse notation and also refer to w as an integer in $[0, 2^n - 1]$ where the most significant bit is the leftmost bit. In other words, we consider the integer $\Sigma_{i=1,\cdots,n} w_i \cdot 2^{n-i}$, where w_i is the i-th bit of the string w.

Definition 2.1 ((Perfect) Correctness of Positional Witness Encryption)**.** *For any security parameter λ, for any $m \in \mathcal{M}$, and for any $x \in L$ such that $R(x, w)$ holds* for *$w \geq t$, we have that*

$$\mathit{Decrypt}_{\mathrm{PWE}}\big(\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t, m), w\big) = m.$$

2.1 Security of Positional Witness Encryption

Message Indistinguishability The security of a positional witness encryption for language L is given as two security properties. The first is message indistinguishability, which is parameterized by an instance x and two equal length messages m_0, m_1. Intuitively, the security property states that if one encrypts to the "final" position $t = 2^n$ (where n is the witness length of x) then no attacker can distinguish whether a ciphertext is an encryption of m_0 or m_1. *We emphasize that this security property is entirely independent of whether $x \in L$.* We define the (parameterized) advantage of an attacker as

$$\mathsf{MsgPWE\,Adv}_{\mathcal{A},x,m_0,m_1}(\lambda) =$$

$$\Big|\Pr[\mathcal{A}(\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t = 2^n, m_1)) = 1] - \Pr[\mathcal{A}(\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t = 2^n, m_0)) = 1]\Big|.$$

Definition 2.2 (Message Indistinguishability Security of Positional Witness Encryption)**.** *We say that a positional witness encryption scheme for a language L with witness relation $R(\cdot,\cdot)$ is Message Indistinguishability secure if for any probabilistic poly-time attack algorithm $\mathcal{A}$ there exists a negligible function in the security parameter $negl(\cdot)$ such that for all instances x and equal length messages m_0, m_1 we have $\mathsf{MsgPWE\,Adv}_{\mathcal{A},x,m_0,m_1}(\lambda) \leq negl(\lambda)$.*

We let $\mathsf{MsgPWE\,Adv}_{\mathcal{A},x}(\lambda)$ be the maximum value of $\mathsf{MsgPWE\,Adv}_{\mathcal{A},x,m_0,m_1}(\lambda)$ over the pairs $m_0, m_1 \in \mathcal{M}$ for each λ.

Position Indistinguishability The second security game is positional indistinguishability. Informally, this security game states that it is hard to distinguish between an encryption to position t from an encryption to $t+1$ when t is not a valid witness – that is, $R(x, t) = 0$. (Here we slightly abuse notation in the other direction by interpreting the integer t as a bit string.) Positional indistinguishability security is parameterized by an instance x, a message m, and a position $t \in [0, 2^n - 1]$ where n is the witness length of x. We define the (parameterized) advantage of an attacker as

$$\mathsf{PosPWE\,Adv}_{\mathcal{A},x,m,t}(\lambda) = \Big|\Pr[\mathcal{A}(\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t+1, m)) = 1] - \Pr[\mathcal{A}(\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, x, t, m)) = 1]\Big|.$$

Definition 2.3 (Position Indistinguishability Security of Positional Witness Encryption)**.** *We say that a positional witness encryption scheme for a language L with witness relation $R(\cdot,\cdot)$ is Position Indistinguishability secure if for any probabilistic poly-time attack algorithm $\mathcal{A}$ there exists a negligible function in the security parameter $negl(\cdot)$ such that for all instances x, all message m, and any $t \in [0, 2^n - 1]$ where $R(x, t) = 0$ we have $\mathsf{PosPWE\,Adv}_{\mathcal{A},x,m,t}(\lambda) \leq negl(\lambda)$.*

We let $\mathsf{PosPWE\,Adv}_{\mathcal{A},x}(\lambda)$ be the maximum value of $\mathsf{PosPWE\,Adv}_{\mathcal{A},x,m,t}(\lambda)$ over $m \in \mathcal{M}$ and $t \in [0, 2^n]$ where $R(x, t) = 0$ for each λ.

We further require that both the message length and the problem statement length must be bounded by some polynomial of the security parameter.

A quick note on the witness encryption definition. We provide the definition of witness encryption in Appendix A. The definition of our appendix follows the original of Garg, Gentry, Sahai, and Waters [15], but with two modifications. First, we restrict ourselves to perfect correctness for simplicity. Second, in defining soundness security we use a notation that the scheme is secure if for all PPT attackers there exists a negligible function $\text{negl}(\cdot)$ such that for any $x \notin L$ the attacker must only be able to distinguish encryption with probability at most $\text{negl}(\lambda)$. The GGSW definition had a different ordering of quantifiers which allowed the bounding negligible function for a particular attacker to depend on the instance x. Bellare and Tung Hoang [1] showed that this formulation was problematic for multiple applications of witness encryption. Our positional witness encryption definition follows a similar corrected ordering of quantifiers.

2.2 Building Witness Encryption from Positional Witness Encryption

We now show how to build a witness encryption scheme from a positional witness encryption scheme. To witness encrypt to an instance x, simply do a positional encryption to position $t = 0$, which allows for decryption using any valid witness.

The benefit of building WE from positional witness encryption comes from a hybrid security argument. Suppose $x \notin L$, then it will be the case that for all $t \in [0, 2^n - 1]$, $R(x,t) = 0$ and thus no attacker can distinguish an encryption to position t to one of $t+1$. This argument can be applied repeatedly to "move" the encryption position from 0 to 2^n. The cost of performing this hybrid is a security loss factor of 2^n and thus it innately requires complexity leveraging. The benefit is that while the security game of witness encryption needs to simultaneously consider *all* possible witnesses, the positional security game is only concerned about one particular witness and thus is potentially much more amenable to a reduction under a simple assumption.

We now formally describe the construction of witness encryption from positional witness encryption followed by a security proof.

$\boldsymbol{Encrypt}_{\text{WE}}(1^\lambda, x, m)$ calls $\mathit{Encrypt}_{\text{PWE}}(1^\lambda, x, t = 0, m)$.

$\boldsymbol{Decrypt}_{\text{WE}}(\text{CT}, w)$ calls $\mathit{Decrypt}_{\text{PWE}}(\text{CT}, w)$.

The correctness conditions of positional witness encryption are equivalent to those of standard witness encryption when $t = 0$ since $0 \le w$ for all $w \in [0, 2^n - 1]$. Therefore, the correctness of the witness encryption follows immediately.

We now state and prove our security theorem.

Theorem 2.4. *Consider the constructed witness encryption scheme for a language L with witness relation $R(\cdot,\cdot)$ and the security soundness property for any $x \notin L$, and two equal length messages m_0, m_1. We have that for any polynomial time attacker $\mathcal{A}$*

$$\mathsf{WE\,Adv}_{\mathcal{A},x,m_0,m_1}(\lambda) \le 2^n \cdot \ \mathsf{PosPWE\,Adv}_{\mathcal{A},x}(\lambda) + \mathsf{MsgPWE\,Adv}_{\mathcal{A},x}(\lambda).$$

Proof. For $j \in [0, 2^n]$ let $\mathsf{Hyb_j}$ be a hybrid experiment that is identical to the original except that the challenge ciphertext is generated as a call to $\mathit{Encrypt}_{\text{PWE}}(1^\lambda, x, t = j, m_b)$. Note that $\mathsf{Hyb_0}$ corresponds to the actual witness encryption security game. Let $\mathsf{AdvHyb_j}$ denote the advantage of the attacker $\mathcal{A}$ in experiment $\mathsf{Hyb_j}$.

By a direct reduction to the security of the position indistinguishability property, it holds that the advantage of any attacker in $\mathsf{AdvHyb_{j+1}} \le \mathsf{AdvHyb_j} + \mathsf{PosPWE\,Adv}_{\mathcal{A},x}(\lambda)$.

Applying induction implies that $\mathsf{AdvHyb}_0 \leq 2^n \cdot \mathsf{PosPWE\,Adv}_{\mathcal{A},x}(\lambda) + \mathsf{AdvHyb}_{2^n}$. Now by definition $\mathsf{AdvHyb}_{2^n} = \mathsf{MsgPWE\,Adv}_{\mathcal{A},x}(\lambda)$. Plugging this in results in our theorem.

□

Required Security and Complexity Leveraging If all of the terms in our reduction were polynominal in n (and thus λ) then the only requirement we would have is that any poly-time algorithm have negligible advantage in each of our security games. However, there is an exponential term of 2^n attached to the positional game. Therefore we will need to use complexity leveraging and for all poly-time algorithms $\mathcal{A}$ demand that:

$$\mathsf{PosPWE\,Adv}_{\mathcal{A},x}(\lambda) = \mathrm{negl}(\lambda) \cdot 2^{-n}$$

where $\mathrm{negl}(\lambda)$ is some negligible function. This requirement will be passed down to our next level of abstraction and eventually to our multi-linear encoding instantiation. At the instantiation level the security parameter will be increased to match this condition.

3 Tribes Schemes

To build positional witness encryption schemes, we will employ an intermediary object which we call a tribes scheme. Before we can define this object and demonstrate its usefulness, we must first define tribes matrices.

3.1 Tribes Matrices

A tribes matrix M is an $n \times \ell \times 2$ 3-dimensional matrix, with entries belonging to the two symbol alphabet $\{B, U\}$, which stand for "blocked" and "unblocked". We consider $[n] = \{1, 2, \ldots, n\}$ as indexing the rows, $[\ell]$ as indexing the columns, and $\{0, 1\}$ as indexing the "slots" (i.e. we think of M as an $n \times \ell$ matrix whose entries are pairs of slots, each containing a symbol from $\{B, U\}$).

Such a matrix M defines a boolean function f_M from $\{0, 1\}^n$ to $\{0, 1\}$ as follows. Given an input $x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$, we examine each column of M. Suppose, for example, that we are considering column j. We cycle through the n rows of this column, and while considering row i, we take the value of the slot whose index matches x_i. If the column contains *at least one* value U in these slots, then we define the value of the column to be 0. Otherwise, we define it to be 1. Finally, if there exists a column with value 1, we define the output of the function to be 1, otherwise it is 0.

More formally, we define f_M as:

$$f_M(x) := \begin{cases} 1, & \exists j \text{ s.t. } M_{i,j,x_i} = B\ \forall i \in [n]; \\ 0, & \text{otherwise.} \end{cases}$$

The name for these matrices is inspired by the tribes function, an interesting object in boolean function analysis. In that domain, one considers the input boolean vector as specifying the "votes" of a population that is organized into tribes, and the output is 1 if and only if there exists a tribe that unanimously voted 1. This is not a perfect analogy to our setting, since we view the input not as specifying these votes directly but rather as selecting each vote from a predetermined set of two possible values. Nonetheless, we adopt the "tribes" terminology as a helpful device for reinforcing the key feature here that the output of our function is 1 if and only some column takes on unanimous values of B when the input is used for indexing the slots.

3.2 Tribes Schemes

We next use the notion of tribes matrices to define a cryptographic primitive that we will call a tribes scheme. A tribes scheme will have two algorithms. The first algorithm, Create, will take in tribes matrices and generate objects that we will call cryptographic tribes. This algorithm is randomized. The second algorithm, Eval, will take in a cryptographic tribe and an input and compute the boolean function described above for the tribes matrix that is incorporated in the cryptographic tribe. This algorithm is deterministic.

Create$(\lambda, M) \rightarrow T$ The creation algorithm takes in a security parameter λ and a tribes matrix M and outputs a cryptographic tribe T.

Eval$(T, x \in \{0,1\}^n) \rightarrow \{0,1\}$ The evaluation algorithm takes in a cryptographic tribe T and a boolean vector x and outputs a value $\{0,1\}$.

Correctness We require perfect correctness, meaning that for every tribes matrix $M \in \{B, U\}^{n \times \ell \times 2}$, for any security parameter λ, and for any input vector $x \in \{0,1\}^n$, we have that

$$Eval(Create(\lambda, M), x) = f_M(x).$$

3.3 Tribes Security Properties

We will define two security properties for a tribes scheme. Both will be defined as typical distinguishing games between a challenger and an attacker. We call the first of these the intra-column game, as it only relies on a condition within a single column of the underlying tribes matrix. We call the other the inter-column game, as it involves a relationship between two columns that allows us to change a "U" symbol to a "B".

Intra-column Game This game is parameterized by a security parameter λ, a tribes matrix M, an index j of a column in M such that there is some row i^* where both slots take the value U[3], and an alternate column $C \in \{B, U\}^{n \times 2}$ such that the row i^* also has both slots equal to U. All of these parameters are given both to the challenger and to the attacker.

The challenger samples a uniformly random bit $b \in \{0,1\}$. If $b = 0$, it runs $Create(\lambda, M)$ to produce a cryptographic tribe T. If $b = 1$, it forms M' by replacing the j^{th} column of M with C, and then runs $Create(\lambda, M')$ to produce T. It gives T to the attacker, who must then guess the value of the bit b.

Definition 3.1. *We say a tribes scheme has* intra-column security *if for every polynomial time attacker $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that the attacker's advantage in the Intra-column Game is $\leq negl(\lambda)$, for any valid settings of M, j, C. Note that the negligible function depends only on $\mathcal{A}$ and λ, and is independent of the dimensions of M, for example.*

[3]Of course, for an arbitrary tribes matrix, such a column may not exist. This is an extra condition we are imposing on M, and this property is only defined for such M.

Inter-column Game This game is parameterized by a security parameter λ, a tribes matrix M, two indices j and k of columns of M, an index i^* of a row of M, and a slot index β such that $M_{i^*,j,\beta} = B$. We require the following condition on the j^{th} and k^{th} columns of M. For every row i and slot γ *except for* $i = i^*$ *and* $\gamma = 1 - \beta$, if $M_{i,k,\gamma} = B$, then $M_{i,j,\gamma} = B$ as well (i.e. when there is only one U among these values, it is always in column k)[4]. All of these parameters are given both to the challenger and to the attacker.

The challenger samples a uniformly random bit $b \in \{0, 1\}$. If $b = 0$, it runs $\text{Create}(\lambda, M)$ to produce a cryptographic tribe T. If $b = 1$, it forms M' by copying M except for flipping just one entry: $M'_{i^*,k,\beta} = B$ if $M_{i^*,k,\beta} = U$, and $M'_{i^*,k,\beta} = U$ if $M_{i^*,k,\beta} = B$. It then runs $Create(\lambda, M')$ to produce T. The challenger gives T to the attacker, who finally must guess the value of the bit b.

Definition 3.2. *We say a tribes scheme has* inter-column security *if for every polynomial attacker* $\mathcal{A}$*, there exists a negligible function* $negl(\lambda)$ *such that the attacker's advantage in the Inter-Column Game is* $\leq negl(\lambda)$*, for any valid settings of* M, j, k, i, β*. Note that the negligible function depends only on* $\mathcal{A}$ *and* λ*, and is independent of the dimensions of* M*, for example.*

Required Security To be useful for ultimately building witness encryption, the required security of all of our security games is that they must be $\text{negl}(\lambda) \cdot 2^{-n}$ where $\text{negl}(\lambda)$ is some negligible function. The demand for the 2^{-n} term is passed down from the positional hybrid of the previous Section 2. In the next section we show how to build positional WE from a Tribes scheme. Since that reduction involves only a polynomial number of hybrids in n (and thus λ) these are absorbed in the negligible function.

3.4 Tribes-lite

We introduce a variant of the tribes primitive that we call tribes-lite. A tribes-lite scheme is the same as a tribes scheme except that we do not require it to have intra-column security. We then show that it is possible to construct a (full) tribes scheme by a simple transformation from a tribes-lite scheme.

Suppose that we want to construct a tribes scheme for a matrix M of dimension $n \times \ell \times 2$. We achieve this by creating a tribes-lite scheme with matrix M' of dimension $n \times (\ell + n) \times 2$. The matrix M' will contain M plus append n "helper columns", $d_1, \ldots, d_n$. The column d_i has the property that in row i it contains a pair of unblocked entries (U, U) and in every other row $k \neq i$ it has a pair of blocked entries (B, B). Since each of these additional columns has a row with two unblocked entries, these will have no impact on the evaluation semantics and correctness follows. Inter-column security of the construction is immediately implied by the inter-column security of the underlying tribes scheme. Intra-column security can be derived (via a hybrid argument) from inter-column security of the underlying tribes-lite scheme. The proof of security (given below) relies on the presence of column d_{i^*} when invoking the intra-column security rule for row i^*.

The impact of this transformation is that it is possible to build a tribes scheme by just constructing a tribes-lite scheme and then applying our transformation. In creating our instantiations in Sections 5, 6, and 7, we choose to construct a full tribes schemes directly. To do so, we apply two multilinear map assumptions (in each instance): one to prove inter-column security and a second for intra-column security. However, we observe that if we instead construct a tribes-lite scheme

[4]Again, these are extra conditions we are imposing on M, j, k, β in order for this game to be applicable.

and then apply the transformation described, it is then sufficient to only use one assumption for each instantiation.

We give our formal definition and theorem below.

Definition 3.3. *A tribes-lite scheme has all the properties of a tribes scheme, except that it is not required to have intra-column security.*

Theorem 3.4. *Given a tribes-lite scheme with algorithms $Create_{lite}$, $Eval_{lite}$, we can construct a tribes scheme with algorithms Create, Eval such that running Create on a $n \times \ell \times 2$ tribes matrix requires running $Create_{lite}$ on an $n \times (n+\ell) \times 2$ tribes matrix.*

Proof. We define Create, Eval as follows:

Create$(\lambda, M) \to T$ The creation algorithm takes in a security parameter λ and a tribes matrix M of dimensions $n \times \ell \times 2$. It forms a $n \times (n+\ell) \times 2$ matrix M' by appending n columns to M. The i^{th} appended column (which has column index $\ell + i$ in M') has U, U in the slots of row i and B, B in the slots of all other rows. The output of $\text{Create}_{lite}(\lambda, M')$ is then given as the output of $\text{Create}(\lambda, M)$.

Eval$(T, x \in \{0,1\}^n) \to \{0,1\}$ The evaluation algorithm takes in a cryptographic tribe T and a boolean vector x and outputs the result of $\text{Eval}_{lite}(T, x)$.

Correctness Assuming perfect correctness of the tribes-lite scheme, we see that for every tribes matrix $M \in \{B, U\}^{n \times \ell \times 2}$, for any security parameter λ, and for any input vector $x \in \{0,1\}^n$, we have that

$$Eval(Create(\lambda, M), x) = Eval_{lite}(Create_{lite}(\lambda, M'), x) = f_{M'}(x) = f_M(x),$$

by definition of $f_M, f_{M'}$, since the appended columns do not effect the computation of f.

Security Since the columns of M are a subset of columns of M', inter-column security for the tribes-lite scheme immediately implies inter-column security for our tribes scheme. Hence it remains to show intra-column security. We will establish this via a hybrid argument, where each step in the hybrid will be an application of the inter-column security property for the tribes-lite scheme.

We consider an instance of the intra-column game parameterized by a row index i^* and a column index j such that both slots of row i^* in column j of M are equal to U, as well as an alternate column C. We can gradually replace column j of M with C by invoking the inter-column security game on M' once for each entry of this column in M that differs from C. In every case, we can use the i^* added column, which has all B's except for row i^*. Note that this column will always satisfy our condition for the inter-column game, as its B entries are a superset of the B entries in the j^{th} column of M and C. Hence intra-column security for our tribes schemes follows from the inter-column security of the tribes-lite scheme. □

4 Constructing a Positional Witness Encryption Scheme from a Tribes Scheme

We will now show how to build a positional witness encryption scheme from a tribes scheme.

4.1 Encoding a CNF in a Tribal Matrix

Suppose we have a CNF formula ϕ with n variables and ℓ clauses. In other words, we can write $\phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_\ell$, where each ϕ_i is a clause over the variables $X_1, \ldots, X_n$ and their negations, denoted $\overline{X_1}, \ldots, \overline{X_n}$. We will define an $n \times \ell \times 2$ tribes matrix M^ϕ.

In order to set the entries of the j^{th} column of M^ϕ, we consider the j^{th} clause ϕ_j. For each row i, we do the following:

- If X_i appears in ϕ_j, we set $M^\phi_{i,j,1} = U$.
- If $\overline{X_i}$ appears in ϕ_j, we set $M^\phi_{i,j,0} = U$.
- For any entries $M^\phi_{i,j,\beta}$ not yet defined, set $M^\phi_{i,j,\beta} = B$.

We note the following property of M^ϕ:

Lemma 4.1. *If we consider a boolean string $x \in \{0,1\}^n$ as an assignment of truth values to the variables $X_1, \ldots, X_n$ of ϕ, then if clause ϕ_j is* unsatisfied *by x, column j of M^ϕ will evaluate to value 1, and hence $f_{M^\phi}(x) = 1$. If x satisfies ϕ, then $f_{M^\phi}(x) = 0$.*

Proof. Suppose clause ϕ_j is unsatisfied by the assignment x. For each $i \in [n]$, we consider M^ϕ_{i,j,x_i}. If $x_i = 0$, then ϕ_j unsatisfied implies that $\overline{X_i}$ does not appear in ϕ_j, and so $M^\phi_{i,j,0} = B$. Similarly, if $x_i = 1$, then ϕ_j unsatisfied implies X_i does not appear in ϕ_j, so $M^\phi_{i,j,1} = B$. Thus, $f_{M^\phi}(x) = 1$. Conversely, if x satisfies ϕ, then for each column j, there exists some row i such that either $x_i = 0$ and $\overline{X_i}$ appears in ϕ_j or $x_i = 1$ and X_i appears in ϕ_j. Either way, $M^\phi_{i,j,x_i} = U$. Hence, $f_{M^\phi}(x) = 0$. □

4.2 Encoding a Position in a Tribal Matrix

Suppose we have a position t considered as a binary string $t = (t_1, t_2, \ldots, t_n) \in \{0,1\}^n$. We will define an $n \times n \times 2$ tribes matrix M^t. We describe M^t by specifying how to fill in the j^{th} column of M^t:

To Set Column j:

- For $i < j$,
  $$M^t_{i,j,0} = B,$$
  $$M^t_{i,j,1} = \begin{cases} U, & \text{if } t_i = 0; \\ B, & \text{if } t_i = 1. \end{cases}$$
- For $i = j$,
  $$M^t_{i,j,0} = \begin{cases} U, & \text{if } t_i = 0; \\ B, & \text{if } t_i = 1. \end{cases}$$
  $$M^t_{i,j,1} = U$$
- For $i > j$,
  $$M^t_{i,j,0} = B = M^t_{i,j,1}.$$

We now establish some relevant properties of M^t. First, we observe that the associated boolean function f_{M^t} evaluates to 1 for every boolean string $y < t$ and evaluates to 0 for every $y \geq t$. Here, we use "$<$" and "$\geq$" to denote the order induced by the usual ordering of integers, when we think of t, y as binary expansions with t_1, y_1 being the most significant bits.

Lemma 4.2. *If $y < t$, then $f_{M^t}(y) = 1$.*

Proof. Since $y < t$, there must be some index $k \in [n]$ such that $t_i = y_i$ for all $i < k$ and $t_k = 1$ while $y_k = 0$. We consider the k^{th} column of M^t. We claim that for all i, $M^t_{i,k,y_i} = B$. To see this, we can consult our description of the k^{th} column of M^t above, noting that for $i < k$, whenever $y_i = 1$, then $t_i = 1$ as well (by definition of k). Thus, $f_{M^t}(y) = 1$. $\square$

Lemma 4.3. *If $y \geq t$, then $f_{M^t}(y) = 0$.*

Proof. We let $k \in [n]$ denote an index such that $y_i = t_i$ for all $i \leq k$, and $y_{k+1} = 1$, $t_{k+1} = 0$, if $k + 1 \leq n$. For a column j where $j \leq k$, we observe that $M_{j,j,y_j} = U$, since $y_j = t_j$. For any column j where $j > k$, we observe that $M_{k+1,j,y_{k+1}} = U$. This is because $t_{k+1} = 0$ and $y_{k+1} = 1$. Hence, $f_{M^t}(y) = 0$. $\square$

This defines an effective encoding of positions t from 0 to $2^n - 1$ (considering t as an integer). It will also be useful to have an encoding of 2^n. This will be used in the last step of our hybrid proof of position-hiding security for our positional witness encryption scheme. For simplicity of that step, we define the encoding of 2^n to be a small change from the encoding of $2^n - 1$ that will ensure the corresponding f will also evaluate to 1 on the position $2^n - 1$. Note that for $t = 2^n - 1$, only the diagonal entries of M^t are not completely filled with B slots. So we define M^{2^n} to be the same as M^{2^n-1}, except that the first diagonal entry has both slots equal to B. We observe that $f_{M^{2^n}}(y) = 1$ for all n-bit values y, since the first column is all filled with B values.

4.3 Our Positional Witness Encryption Scheme

We let our message space be $\{0, 1\}$.

Encrypt$_{\mathrm{PWE}}(1^\lambda, \phi, t, m)$ The encryptor constructs M^ϕ and M^t as above. For $m \in \{0, 1\}$, it constructs an additional column C^m (which is $n \times 2$) as follows. If $m = 1$, $C^m_{i,0} = C^m_{i,1} = B$ for all i, and if $m = 0$, $C^m_{i,0} = C^m_{i,1} = U$ for all i. It also constructs a completely unblocked column S, meaning that $S_{i,0} = S_{i,1} = U$ for all i. Note that appending such a column to a tribes matrix will not affect the evaluation function. (This "scratch column" S will be useful in the proof of security.)

It then forms an $n \times (\ell + n + 2) \times 2$ tribes matrix M as $M := M^\phi || M^t || C^m || S$, meaning that the first ℓ columns are taken to be M^ϕ, the next n columns are taken to be M^t, and the final two columns are taken to be C^m and S. The encryptor then calls $Create(\lambda, M)$ to produce a tribes scheme T, and sets $\mathrm{CT} := T$.

Decrypt$_{\mathrm{PWE}}(\mathrm{CT}, w)$ The decryptor runs $Eval(\mathrm{CT}, w)$ and outputs the result.

4.4 Security of our Positional Witness Encryption Scheme

We now prove security of the positional witness encryption based on the two tribes properties on inter-column and intra-column security. The most complex part is the proof of position hiding, which is given over a sequence of hybrid steps. At a very high level the proof (for indistinguishability of position t from $t+1$) proceeds in two stages. In the first stage the reduction algorithm identifies a clause, j, in the CNF formula that the witness candidate $w = t$ does not satisfy. Such a clause must exist for this to be a valid instance of the positional game. The proof then uses the j-th column in the CNF portion of the matrix to (undetectably) change the scratch column S from having U's in each of its $2 \cdot n$ slots to having a B in row i slot t_i for each i. (The other n slots remain U.) The security properties are used to argue that such a change is indistinguishable to an attacker. This copy action into the scratch column will cause the column to evaluate as "blocked" on input t and remain evaluating to unblocked on all other inputs. Intuitively, this will have no impact on the overall evaluation since the j-th column caused a block on input t anyway — providing a conceptual sanity check for our claim. Intuitively, this stage reflects the fact that t is not a valid witness and represents this fact in the scratch column.

The next stage of our proof solely involves the scratch column and position matrix. A series of hybrid steps will use the scratch column to update the positional part of the matrix from position t to position $t+1$ by "assimilating" the scratch column from the previous stage. At the end of these steps that scratch column will again become unblocked in all slots and thus matching the end goal of our argument.

We now prove:

Theorem 4.4. *The positional witness encryption scheme Encrypt*$_{\mathrm{PWE}}$*, Decrypt*$_{\mathrm{PWE}}$ *defined in Section 4.3 is message hiding secure and position hiding secure if the underlying tribes scheme has intra-column and inter-column security.*

We first prove position hiding security.

Theorem 4.5. *The positional witness encryption scheme Encrypt*$_{\mathrm{PWE}}$*, Decrypt*$_{\mathrm{PWE}}$ *defined in Section 4.3 is position hiding secure if the underlying tribes scheme has intra-column and inter-column security.*

To prove this lemma, we will use a hybrid argument over a sequence of games. These games will involve changes to the portions of the tribes matrix M that correspond to M^t (which encodes the position) and S (which will be used as "scratch space" for the proof). Note that we will use M^ϕ in the proof but will not need to modify it. C^m is rather irrelevant for this lemma, as there is just a single message.

For an arbitrary position t, we observe that the corresponding encoding matrices M^t and M^{t+1} may differ in several entries. Our hybrid argument will transform M^t into M^{t+1} gradually, changing only a single entry at a time with applications of the inter-column security game and changing a single column at a time with applications of the intra-column security game.

We assume throughout our hybrid argument that we have a fixed t which *does not* satisfy the encoded CNF formula ϕ. To define the sequence of game transitions, we also consider the position t as a binary vector, $t = (t_1, \ldots, t_n)$, with t_1 being the most significant bit. For such an t, we define a column $A^t \in \{U, B\}^{n \times 2}$ as follows. For each i from 1 to n, we set $A^t_{i,t_i} = B$, $A^t_{i,1-t_i} = U$. Note that this is a column that evaluates to 1 exactly on t and otherwise evaluates to 0. The first portion of our hybrid argument will be devoted to transitioning from a completely unblocked scratch column S to having A^t in that column. We define Game$_0$ to be a variant of the original game for position hiding security where the attacker is given either a tribes scheme created from $M_0 := M^\phi||M^t||C^m||S$ or a tribes scheme created from $M_1 := M^\phi||M^t||C^m||A^t$.

Lemma 4.6. *If the underlying tribes scheme has intra-column and inter-column security, then any PPT attacker has only a negligible advantage in Game$_0$.*

Proof. First, by intra-column security, we can transition from $M := M^\phi||M^t||C^m||S$ to $M := M^\phi||M^t||C^m||\tilde{A}^t$, where $\tilde{A}^t$ is identical to A^t except that it has U in both slots in row 1. We let j denote the index of a clause ϕ_j that is unsatisfied by t. We consider column j of M^ϕ, denoted M^ϕ_j. We will use this column to justify our switch of $\tilde{A}^t_{1,t_i}$ from a U to a B (this will result in A^t) by invoking the inter-column security game with k equal to $\ell + n + 2$ (the index of the last column), and $i^* = 1$, $\beta = t_1$. To see that this applies, note that for every i, $M^\phi_{i,j,t_i} = B$, as this is implied by the fact that the assignment t does not satisfy clause ϕ_j (see Lemma 4.1). □

Remark 1. *An alternative way to prove the above lemma is to only apply the inter-column security game. One would apply this n times from the identified column j to the last "scratch" column (using one invocation for each blocked entry of A^t. Using either method the result is that A^t is "copied" into the scratch column and this is possible by the fact that t is not satisfied by at least one clause j.*

We let r be index of the last 0 in t, assuming for now that $t \neq (1, 1, \ldots, 1)$. In other words, $r = n$ if $t_n = 0$, $r = n - 1$ if $t_n \neq 0, t_{n-1} = 0$, and so on. For every z from 1 to $n - r$, we define $M_{1,z}$ to be the same as M_1, except the *last* z rows of the final column have all slots $= B$, instead of being A^t. For each such z, we define Game$_{1,z}$ to be a variant of the security game where the attacker is given a tribes scheme created either from $M_{1,z-1}$ or $M_{1,z}$ (where we interpret $M_{1,0}$ as M_1).

Lemma 4.7. *For each z, if the underlying tribes scheme has inter-column security, then any PPT attacker has only a negligible advantage in Game$_{1,z}$.*

Proof. We will invoke the inter-column security game with $k = \ell + n + 2$, $i^* = n - (z - 1)$, $\beta = 1 - t_{n-(z-1)} = 0$, and $j = \ell + n - (z - 1)$. To see that this is a valid use of inter-column security, first note that $M_{i^*,j,0} = B$, since this is a diagonal position inside M^t and $t_{n-(z-1)} = 1$. (Referring to our positional encoding rules of Section 4.2.) Next note that for $i > n - (z - 1)$, $M_{i,j,0} = M_{i,j,1} = B$, since these entries are below the diagonal in M^t. For all $i < n - (z - 1)$, $M_{i,j,t_i} = B$ holds, by definition of M^t. Note that these are the only B entries in A^t that appear for these rows. Lastly, note that the single slot $M_{n-(z-1),j,1}$ is exempted from the condition that column j have a B wherever column k does in the definition of our inter-column security game. □

This brings us to the matrix $M_2 := M_{1,n-r}$. Now for z from 0 to $r - 1$, we define $M_{2,r-1-z}$ to be the same as M_2, except the first $(r - 1) - z$ rows of the final column match A^t, while the next z rows match column r inside M^t. For each z from 1 to $r - 1$, we define Game$_{2,z}$ to be a variant of the security game where the attacker is given a tribes scheme created either from $M_{2,r-2-z}$ or from $M_{2,r-1-z}$. Note that in this part of the proof we are progressing forward by starting with $z = r - 1$ and "counting down" to $z = 0$.

Lemma 4.8. *For each z from 1 to $r - 1$, if the underlying tribes scheme has inter-column security, then any PPT attacker has only a negligible advantage in Game$_{2,z}$.*

Proof. First we consider what the $r-1-z$ row of A^t and the r^{th} column of M^t have in their slots. If $t_{r-1-z} = 0$, then A^t and the r^{th} column of M^t both have a B value in the 0 slot and a U value in the 1 slot on this row. Thus, there is no difference here, and the games are in fact identical.

It thus suffices to consider the case where $t_{r-1-z} = 1$. In this case, A^t has a U in the 0 slot and a B in the 1 slot, while the r^{th} column of M^t has B's in both slots on this row. So we will invoke the inter-column security game with $k = \ell + n + 2$, $i^* = r - 1 - z$, $\beta = 0$, and $j = \ell + r - 1 - z$.

To see that this applies, we first observe that $M_{r-1-z,j,0} = B$, since this is a diagonal position inside M^t and $t_{r-1-z} = 1$. We next need to check that for every B that currently appears in the k^{th} column, we have a B in the same position in the j^{th} column. We first consider the rows $i > r - 1 - z$. For these rows, the j^{th} column is full of B's in all slots, as these are below diagonal rows of a column inside M^t. For rows $i < r - 1 - z$, the k^{th} column currently matches A^t, and so only has B in the slots corresponding to the t_i values. Note that above diagonal entries of a column inside M^t will also have B's in these slots, so we can indeed apply inter-column security to change the indicated slot in column k from a U to a B. $\square$

This brings us to the matrix $M_3 := M_{2,0}$, where the final column of M (the scratch column) is now exactly what we would like the $\ell + r$ column inside M to become in order to switch from M^t to M^{t+1}. We let M_4 denote the matrix that is formed by adjusting M_3 so that the $\ell + r$ column matches the r^{th} column in the definition of M^{t+1}. We define Game$_3$ to be a variant of the security game where the attacker is given a tribes scheme created either from M_3 or from M_4.

Lemma 4.9. *If the underlying tribes scheme has inter-column security, then any PPT attacker has only a negligible advantage in Game$_3$.*

Proof. In fact, the $\ell + r$ column inside M_3 only differs from this column inside M_4 in one position: namely a U that appears in the 0 slot in row r that we would like to change to a B. Clearly, since the scratch column has a B in this position, we can use the inter-column security game with $j = \ell + n + 2$, $k = \ell + r$, $i^* = r$, and $\beta = 0$ to make this transition. $\square$

We now have the $\ell + r$ column of M_4 matching what it should be in M^{t+1}. We now must change columns $\ell + r + z$ as z ranges from 1 to $n - r$. For each such z, we let $M_{4,z}$ denote the matrix which is similar to M_4, except that the columns $\ell + r$ up to $\ell + r + z$ match M^{t+1}. Note that $M_{4,0} = M_4$. For z from 1 to $n - r$, we define Game$_{4,z}$ to be a variant of the security game where the attacker is given a tribes scheme created either from $M_{4,z-1}$ or from $M_{4,z}$.

Lemma 4.10. *For each z, if the underlying tribes scheme has intra-column and inter-column security, then any PPT attacker has only a negligible advantage in Game$_{4,z}$.*

Proof. To transition from $M_{4,z-1}$ to $M_{4,z}$, we consider the $r + z$ column of M^t. We consider its diagonal slots, i.e. the slots on row $r + z$. We wish to change the value in the 0 slot here from B to U. To do this, we invoke inter-column security with $i^* = r + z$, $\beta = 0$, $k = \ell + r + z$, and $j = \ell + r$. To see that this applies, note that column j of M_4 is equal to the r^{th} column of M^{t+1}, and this has a B in the relevant slot because it is a below diagonal entry. Furthermore, these columns j and k agree in all of their entries above row i^*. Now, once we have we this row i^* with both slots having U values, we can invoke intra-column security to change the rest of the $\ell + r + z$ column to the value it should take in $M_{4,z}$. $\square$

We let $M_5 := M_{4,n-r}$. We note that this is almost the same as the desired $M_6 := M^{\phi}||M^{t+1}||C^m||S$, except that the last column, S, has not been reset to all U values. For this last transition, we define Game$_5$ as a variant of the security game where the attacker is given a tribes scheme created either from M_5 or M_6.

Lemma 4.11. *If the underlying tribes scheme has intra-column and inter-column security, then any PPT attacker has only a negligible advantage in Game$_5$.*

Proof. In M_5, the final column is still an exact copy of column $\ell + r$, which is where we left it after using it to perform our switch from M^t to M^{t+1}. Thus, by an easy application of inter-column security with $k = \ell + n + 2$ and $j + \ell + r$, we can create a row of the final column that has U's in both slots. Then, finally invoking intra-column security, we can set it back to all U values in all rows. □

4.4.1 The Last Position

In the above argument, we assumed that $t < 2^n - 1$. Finally we must argue that we can transition from $M^{\phi}||M^{2^n-1}||C^m||S$ to $M^{\phi}||M^{2^n}||C^m||S$. This will be similar to the argument applied above and will take multiple steps. The first steps will be identical to the argument above, as the assumption that $t < 2^n - 1$ only became relevant midway through the argument. For notational convenience, we now set $t := 2^n - 1$ and (re-)define $M_0 := M^{\phi}||M^{2^n-1}||C^m||S$ as our starting point and $M_1 := M^{\phi}||M^{2^n-1}||C^m||A^t$ as our first transition target. Here, A^t is a column that has a B in all 1 slots and a U in all 0 slots. We define Game$'_0$ to be a variant of the security game where the attacker is given a tribes scheme created either from M_0 or M_1.

Lemma 4.12. *If the underlying tribes scheme has intra-column and inter-column security, then any PPT attacker has only a negligible advantage in Game$'_0$.*

Proof. This follows identically to the proof of Lemma 4.6. □

For every z from 1 to n, we define $M_{1,z}$ to be the same as M_1, except the last z rows of the final column, S, have all slots $= B$, instead of being A^t. For each such z, we define Game$'_{1,z}$ to be a variant of the security game where the attacker is given a tribes scheme created either from $M_{1,z-1}$ or $M_{1,z}$ (where we interpret $M_{1,0}$ as M_1).

Lemma 4.13. *For each z, if the underlying tribes scheme has inter-column security, then any PPT attacker has only a negligible advantage in Game$'_{1,z}$.*

Proof. This follows identically to the proof of Lemma 4.7. □

This brings us to the matrix $M_2 := M_{1,n}$, which has a final column, S, of all slots $= B$. We define M_3 to be the same as M_2, except that its $\ell + 1$ column also has all slots $= B$. We define Game$'_2$ to be a variant of the security game where the attacker is given a tribes scheme created either from M_2 or from M_3.

Lemma 4.14. *If the underlying tribes scheme has inter-column security, then any PPT attacker has only a negligible advantage in Game$'_2$.*

Proof. Here we invoke the inter-column security game with $j = \ell + n + 2$, $k = \ell + 1$, $\beta = 1$, and $i^* = 1$. It is easy to confirm that this applies, as the column j (the final column of M_2) contains all B values. □

We have now transitioned to M_3, which is almost equal to the desired $M_4 := M^{\phi}||M^{2^n}||C^m||S$, except we want to reset the last column of M_3 to be all U values instead of being all B values. We will do this with a final application of the inter-column and then intra-column security games. We let Game'_3 be a variant of the security game where the attacker is given a tribes scheme created either from M_3 or from M_4.

Lemma 4.15. *If the underlying tribes scheme has inter-column and intra-column security, then any PPT attacker has only a negligible advantage in Game'_3.*

Proof. We invoke the inter-column security game twice with $j = \ell + 1$, $k = \ell + n + 2$, $i^* = 1$, for $\beta = 0$ and $\beta = 1$. We note that column j contains all B's, so these invocations allow us to change the first row of the final column to have U's in both slots. Then, by a final application of the intra-column security game, we can change the remaining rows of the final column to U's as well. □

This completes our proof of Theorem 4.5.

4.4.2 Message Hiding Security

We next prove message hiding security:

Theorem 4.16. *The positional witness encryption scheme $\mathit{Encrypt}_{\text{PWE}}$, $\mathit{Decrypt}_{\text{PWE}}$ defined in Section 4.3 is message hiding secure if the underlying tribes scheme has intra-column and inter-column security.*

We let Expt_0 denote the real security game for message hiding security with $m_b = 0$. We let Expt_1 denote the real security game for message hiding security with $m_b = 1$. We additionally define an intermediary experiment, $\text{Expt}_{0.5}$, where the message column C^m is equal to C^1 except for U values in both slots in the first row. We note that if no PPT attacker can distinguish between Expt_0 and Expt_1, then message hiding security holds.

Lemma 4.17. *If the underlying tribes scheme has intra-column security, then any PPT attacker has only a negligible advantage in distinguishing between Expt_0 and $\mathit{Expt}_{0.5}$.*

Proof. To transition between Expt_0 and $\text{Expt}_{0.5}$, we observe that this is a direct application of intra-column security for the column C^0, since this already has U's in both slots in the first row. □

Lemma 4.18. *If the underlying tribes scheme has inter-column security, then any PPT attacker has only a negligible advantage in distinguishing between $\mathit{Expt}_{0.5}$ and Expt_1.*

Proof. To transition from $\text{Expt}_{0.5}$ to Expt_1, we use two applications of the inter-column security with $j = \ell + 1$, $k = \ell + n + 1$ (the message column), $i^* = 1$, for $\beta = 0$ and $\beta = 1$. Note that column j is filled with all B values, so the requirements of the inter-column game are satisfied. □

Taken together, Lemmas 4.17 and 4.18 imply Theorem 4.16. Finally, Theorems 4.16 and Theorem 4.5 imply Theorem 4.4.

5 An Instantiation in a Symmetric Model of Composite Order Multilinear Groups

We move to presenting three related constructions each from multilinear algebraic groups with an n linear map required for a tribes instance of n rows. Our first instantiation (presented in this section) uses *composite* order symmetric multilinear groups. The group's order is a product $n+\ell$ primes for an n by ℓ tribes matrix. Our next instantiation (in Section 6) utilizes asymmetric groups to reduce the order of the group to the product of ℓ primes. Finally, in Section 7 we modify the instantiation to be in prime order groups using a translation based on eigenvectors. Each instance is based on a pair of multilinear map assumptions that depend on n (or n and ℓ), but are independent of the contents of the tribes matrix. In Appendix B, we show how to translate from algebraic groups into the multilinear encodings of Coron, Lepoint and Tibouchi (CLT) [9].

We now present our first instance of a tribes schemes by instantiating it in symmetric composite order multilinear groups.

5.1 An Abstract Model of Composite Order Multilinear Groups

We let G denote a (cyclic) group of order $N = p_1 p_2 \cdots p_r$, where $p_1, \ldots, p_r$ are distinct primes. We let G_T also denote a cyclic group of order N. We suppose that we have a k-linear map $E : G^k \to G_T$. We assume this is non-degenerate, meaning that if g generates G, then $E(g, g, \ldots, g)$ generates G_T. We write the group operations multiplicatively, and we let $1_G, 1_{G_T}$ denote the identity elements in G and G_T respectively.

For each prime p_i dividing the group order of N, we have a subgroup G_{p_i} of order p_i inside G. We let g_{p_i} denote a generator for G_{p_i}. We let $G_{p_1 p_2}$, for example, denote the subgroup of order $p_1 p_2$ that is generated by $g_{p_1} g_{p_2}$.

These subgroups are "orthogonal" under G, meaning (for example) that if $h \in G_{p_1 p_2 \cdots p_{i-1} p_{i+1} p_r}$, then for any $g_2, \ldots, g_{k-1} \in G$,

$$E(h, g_2, \ldots, g_{k-1}, g_{p_i}) = 1_{G_T}.$$

More generally, each element of G can be expressed as $g_{p_1}^{\alpha_1} g_{p_2}^{\alpha_2} \cdots g_{p_r}^{\alpha_r}$. Thus if we have k elements of G that are input to E, we can write them as $g_{p_1}^{\alpha_{1,1}} g_{p_2}^{\alpha_{2,1}} \cdots g_{p_r}^{\alpha_{r,1}}$, …, $g_{p_1}^{\alpha_{1,k}} g_{p_2}^{\alpha_{2,k}} \cdots g_{p_r}^{\alpha_{r,k}}$, and by multi-linearity of E and orthogonality we then have:

$$E(g_{p_1}^{\alpha_{1,1}} g_{p_2}^{\alpha_{2,1}} \cdots g_{p_r}^{\alpha_{r,1}}, \ldots, g_{p_1}^{\alpha_{1,k}} g_{p_2}^{\alpha_{2,k}} \cdots g_{p_r}^{\alpha_{r,k}}) = E(g_{p_1}^{\alpha_{1,1}}, \ldots, g_{p_1}^{\alpha_{1,k}}) \cdots E(g_{p_r}^{\alpha_{r,1}}, \ldots, g_{p_r}^{\alpha_{r,k}}). \quad (1)$$

We let $\mathcal{G}(\lambda, r, k)$ denote a group generation algorithm that takes in a security parameter λ, a desired number of prime factors r, and a desired level of multilinearity k and outputs a description of a group G as above. We assume the description includes a generator $g \in G$, the group order N, the primes $p_1, \ldots, p_r$ comprising N, and efficient algorithms for the group operation in G, the group operation in G_T, and the multilinear map E. Note that with a generator g for the full group plus knowledge of the prime factors, one can efficiently produce a generator for any subgroup of order dividing N.

Computational Assumption 1_S Our first computational assumption in the symmetric setting will be parameterized by positive integers n and ν. It will concern a group of order $N = a_1 \ldots a_n b_1 \ldots b_\nu c$, where $a_1, \ldots, a_n, b_1, \ldots, b_\nu, c$ are $n + \nu + 1$ distinct primes. We give out generators $g_{a_1}, \ldots, g_{a_n}, g_{b_1}, \ldots, g_{b_\nu}$ for each prime order subgroup *except for the subgroup of order* c. For each $i \in [n]$, we also give out a group element h_i sampled uniformly at random from the subgroup of order $ca_1 \cdots a_{i-1}a_{i+1} \cdots a_n$. The challenge term is a group element $T \in G$ that is either sampled uniformly at random from the subgroup or order $ca_1 \cdots a_n$ or uniformly at random from the subgroup of order $a_1 \cdots a_n$. The task is to distinguish between these two distributions of T.

We name this assumption the (n, ν)-*multilinear subgroup elimination assumption.* Though we continue to refer to it as assumption 1_S for conciseness below.

Computational Assumption 2_S Our second computational assumption will be parameterized by positive integers n and ν. It will again concern a group of order $N = a_1 \dots a_n b_1 \dots b_\nu c$, where $a_1, \dots, a_n, b_1, \dots, b_\nu, c$ are $n + \nu + 1$ distinct primes. As in Assumption 1, we give out generators $g_{a_1}, \dots, g_{a_n}, g_{b_1}, \dots, g_{b_\nu}$ for each prime order subgroup *except for the subgroup of order c.* The challenge term is a group element T that is either sampled uniformly at random the subgroup of order ca_n or the subgroup of order a_n. The task is to distinguish between these two distributions of T.

We name this assumption the (n, ν)-*multilinear subgroup decision assumption.* Though we continue to refer to it assumption 2_S for conciseness below.

Intuition for generic security of these assumptions To see why the assumption 1_S holds in the multi-linear (symmetric) generic group model, note that one can argue similarly to the bilinear case (see [19]) that if we assume that finding a nontrivial factor of the group order is hard, the only way to break this assumption in the generic group model would be to find an algebraic relationship that holds for one distribution on the challenge and fails for the other. Since the challenge distribution only differs in the presence or absence of the subgroup of order c, one would need to use the challenge as an input to the multilinear map along with other terms that have non-zero subgroup c components. The only choices are the h_i's, but each one of these can only cancel one of the masking a_i subgroups, so when you can only combine the challenge with $n - 1$ of these, you simply cannot get rid of some a_i space this way, and hence will get something random in this subgroup.

The intuition for generic security of assumption 2_S is that combining the challenge with any of the given terms in the multilinear map will zero the contribution from the c component (if present), hence preventing the adversary from detecting its presence. We note that formal proofs of generic security for the more complex analogs of these assumptions that we employ in the asymmetric, non-composite-order setting can be found in Section C.

5.2 Instantiating a Tribes Scheme

Suppose we wish to build a tribes scheme for $n \times \ell \times 2$ tribes matrices, and we have a generation algorithm $\mathcal{G}$ for producing composite order multilinear groups. We construct a tribes scheme as follows:

Create(λ, M)**:** The creation algorithm takes in a security parameter λ and an $n \times \ell \times 2$ tribes matrix M (entries in $\{U, B\}$). It then calls $\mathcal{G}(\lambda, r = n + \ell, n)$ to produce a group G of order $N = p_1 \cdots p_n q_1 \cdots q_\ell$ equipped with an n-linear map E. It will produce $2n$ group elements, each indexed by a row $i \in [n]$ and a slot $\beta \in \{0, 1\}$. We let $g_{i,\beta}$ be sampled as follows. First, for each $i' \neq i$, a uniformly random element $s_{i'}$ of the subgroup of order $p_{i'}$ is sampled. Next, for each column index $j \in [\ell]$, if $M_{i,j,\beta} = B$, then z_j is sampled as a uniformly random element of the subgroup of order q_j. If $M_{i,j,\beta} = U$, then $z_j := 1_G$. (All of these values are freshly resampled for each i, β.) We set:

$$g_{i,\beta} := \prod_{i' \neq i} s_{i'} \prod_{j=1}^{\ell} z_j.$$

The tribes scheme T consists of these $2n$ elements $\{g_{i,\beta}\}$ (we assume this implicitly includes a description of G that enables efficient computation of the group operation and E, and the full group order N, but *not* the individual primes comprising N).

Eval(T, x)**:** The evaluation algorithm takes in a tribes scheme T and a boolean vector $x = (x_1, \ldots, x_n) \in \{0,1\}^n$. It computes $E(g_{1,x_1}, g_{2,x_2}, \ldots, g_{n,x_n})$ and checks whether this is equal to 1_{G_T} or not. If is it the identity, it outputs 0. Otherwise, it outputs 1.

Correctness We first establish that $Eval(Create(\lambda, M), x) = f_M(x)$. We first observe that (by orthogonality of distinct prime order subgroups) $E(g_{1,x_1}, g_{2,x_2}, \ldots, g_{n,x_n})$ can be considered as a product of contributions within each prime order subgroup. Consider a prime p_i. No component in the subgroup of order p_i appears in g_{i,x_i} (regardless of the value of x_i), so this contribution is trivial (just the identity element). For analyzing the contribution of the q_j primes, we consider two cases. Suppose $\exists$ a column j such that $M_{i,j,x_i} = B$ for all $i \in [n]$. This is equivalent to supposing that $f_M(x) = 1$. In this case, there is a random component in the subgroup of order q_j incorporated in *every* g_{i,x_i}, so the contribution will be (with high probably) a non-identity element in the q_j order subgroup of G_T. This cannot be "canceled out" by a contribution in any other prime order subgroup, so the result will be $\neq 1_{G_T}$ in this case, resulting in an output that matches f_M. In the other case, no such column j exists. This means that for every column j, there is some g_{i,x_i} which lacks a component in the order q_j subgroup, hence causing a result of 1_{G_T}, and the output again matches f_M.

We now show that inter-column security for this tribes scheme is implied by our first computational assumption described above.

Remark 2. *For security parameter λ, we ultimately want polynomial time attackers to have advantage at most $2^{-\lambda}$ against our witness encryption scheme. However, there is a 2^n security loss factor from PWE to WE, and there are further* poly(n, ℓ) *losses from our computational assumption to PWE. Thus, in our assumptions, the "real" security parameter is $\lambda' = \lambda + \tilde{O}(n + \ell)$. Furthermore, when our computational assumptions are instantiated with CLT encodings, the size of the encodings will need to be polynomial (certainly super-linear) in λ' to achieve $2^{-\lambda'}$ security. This will be true for all of our instantiations.*

Lemma 5.1. *Assumption 1_S on $\mathcal{G}$ implies inter-column security for this tribes scheme.*

Proof. For notational convenience, we consider the game with $i^* = n$, $\beta = 0$, $j = 2$, $k = 1$ and any $n \times \ell \times 2$ tribes matrix M satisfying the game requirements for these indices. We will define $\nu := \ell - 1$. We suppose we are given $g_{a_1}, \ldots, g_{a_n}, g_{b_1}, \ldots, g_{b_\nu}$ that generate each prime order subgroup except for the subgroup of order c. For each $i \in [n]$, we are also given a group element h_i sampled uniformly at random from the subgroup of order $c a_1 \cdots a_{i-1} a_{i+1} \cdots a_n$. We are lastly given the challenge T, and our task is to guess the distribution T was sampled from.

We will set $p_i := a_i$ for i from 1 to $n-1$, $p_n := b_\nu$, $q_1 := c$, $q_2 := a_n$, $q_i := b_{i-2}$ for $i = 3$ to ℓ. We form the $2n$ group elements to give to $\mathcal{A}$ as follows. First, for each row $i \neq i^*$ and slot σ, we check the values of $M_{i,j',\sigma}$ in each column $j' \neq 1, 2$. For indices j' where these are B, we sample $z_{j'}$ to be a random element of the subgroup of order $q_{j'}$ (note that we can do this by raising $g_{b_{j'-2}}$ to a random exponent modulo N). For values where these are U, we set $z_{j'} := 1_G$. There are a few cases for the values of $M_{i,1,\sigma}$ and $M_{i,2,\sigma}$. In all cases, we choose a random exponent α modulo N. If both are B, we set $z_1 z_2 \prod_{i' \neq i} s_{i'} := h_i^\alpha g_{b_\nu}^\alpha$. If both are U, we set

$$z_1 z_2 \prod_{i' \neq i} s_{i'} = g_{b_\nu}^\alpha \prod_{i' \neq i, i' < n} g_{a_{i'}}^\alpha$$

when $i \neq n$, and when $i = n$ we do the same except we leave off the $g_{b_\nu}^\alpha$ term. In other words, for $i' \neq n$, we have $s_{i'} := g_{a_{i'}}^\alpha$, and for $i' = n$ we have $s_n := g_{b_\nu}^\alpha$. If $M_{i,2,b} = B$ and $M_{i,1,b} = U$, we set $z_1 z_2 \prod_{i' \neq i} s_{i'} := g_{a_n}^\alpha \prod_{i' \neq i} s_{i'}$, where $s_{i'} := g_{a_{i'}}^\alpha$ for $i' \neq n$ and $s_n := g_{b_\nu}^\alpha$. We note that the case of $M_{i,1,b} = B$ and $M_{i,2,b} = U$ is disallowed by the game requirements. This allows us to compute:

$$g_{i,\sigma} := z_1 z_2 z_3 \cdots z_\ell \prod_{i' \neq i} s_{i'},$$

which is properly distributed.

Now, to form $g_{n,1}$, we can sample (fresh) values $z_3, \dots, z_\ell$ as above. We note that any combination of the values of $M_{n,1,1}$ and $M_{n,2,1}$ are possible. If both are B's, we set $z_1 z_2 \prod_{i'<n} s_{i'} := h_n g_{a_n}$. If both are U, we set $z_1 z_2 \prod_{i'<n} s_{i'} := \prod_{i'<n} g_{a_{i'}}$. If $M_{n,2,1} = B$ and $M_{n,1,1} = U$, we set $z_1 z_2 \prod_{i'<n} s_{i'} := \prod_{i' \leq n} g_{a_{i'}}$. If $M_{n,2,1} = U$ and $M_{n,1,1} = B$, we set $z_1 z_2 \prod_{i'<n} s_{i'} := h_n$. We can then compute:

$$g_{n,1} := z_1 \cdots z_\ell \prod_{i'<n} s_{i'},$$

which is properly distributed.

Finally, we must form $g_{n,0}$, whose distribution will depend on the nature of the challenge term T. We can still sample $z_3, \dots, z_\ell$ as above. We then set:

$$g_{n,0} := T z_3 \dots z_\ell.$$

If T was sampled as a random element of the subgroup of order $c a_1 \cdots a_n$, this is properly distributed for the case $M_{n,1,0} = M_{n,2,0} = B$. If T was sampled as a random element of the subgroup of order $a_1 \cdots a_n$, this is properly distributed for the case $M_{n,1,0} = U$ and $M_{n,2,0} = B$. Thus, we can leverage $\mathcal{A}$'s non-negligible advantage in the inter-column security game to obtain a non-negligible advantage against our Assumption 1_S. $\square$

We last show that intra-column security for this tribes scheme is implied by our second computational assumption.

Lemma 5.2. *Assumption 2_S on $\mathcal{G}$ implies intra-column security for this tribes scheme.*

Proof. For notational convenience, we consider the game with $i^* = n$, $j = 1$, and any $n \times \ell \times 2$ tribes matrix M and alternate column C satisfying the game requirements for these indices. We will define $\nu := \ell - 1$. We will break the game into two stages: in the first stage, we will transition to the first column of M being all U entries. In the second stage, we will transition to this column being equal to C.

We describe the reduction to Assumption 2_S for the first stage only, as the second stage is analogous. We suppose we are given $g_{a_1}, \ldots, g_{a_n}, g_{b_1}, \ldots, g_{b_\nu}$ that generate each prime order subgroup except for the subgroup of order c. We are lastly given the challenge T, and our task is to guess the distribution T was sampled from.

We will set $p_i := a_i$ for i from 1 to n, $q_1 := c$, $q_i := b_{i-1}$ for $i = 2$ to ℓ. We form the $2n$ group elements to give to an attacker $\mathcal{A}$ as follows. For each row $i < n$ and slot σ, we check the values of $M_{i,j',\sigma}$ in each column $j' \neq 1$. For indices j' where these are B, we sample $z_{j'}$ to be a random element of the subgroup of order $q_{j'}$ (note that we can do this by raising $g_{b_{j'-1}}$ to a random exponent modulo N). For values where these are U, we set $z_{j'} := 1_G$. For each $i' \neq i, n$, we sample $s_{i'}$ uniformly from the subgroup of order $p_{i'}$ by raising $g_{a_{i'}}$ to a uniformly random exponent modulo N. If $M_{i,1,\sigma} = B$, we choose another random exponent $\alpha \mod N$ and set

$$z_1 s_n := T^\alpha.$$

If $M_{i,1,\sigma} = U$, we set $z_1 s_n := g_{a_n}^\alpha$. This allows us to compute

$$g_{i,\sigma} := z_1 \cdots z_\ell \prod_{i' \neq i} s_{i'},$$

which is properly distributed for the original matrix M if T is uniform in the group of order ca_n, and otherwise is properly distributed for a matrix that is equal to M except for the first column being overwritten by U entries.

For row n and slot σ, we choose a random exponent $\alpha \mod N$ and set $s_{i'} := g_{a_{i'}}^\alpha$ for $i' < n$. We sample $z_2, \ldots, z_\ell$ as above and compute

$$g_{n,\sigma} := z_2 \cdots z_\ell \prod_{i' < n} s_{i'},$$

which is properly distributed for either case, as the last row of M has all U values regardless. Thus, we can leverage an attacker's non-negligible advantage in the intra-column security game to obtain a non-negligible advantage against our Assumption 2_S.

□

6 An Instantiation in an Asymmetric Model of Composite Order Multilinear Groups

In this section, we give a construction of tribes schemes in for asymmetric composite order multilinear groups.

6.1 An Abstract Model of Asymmetric Composite Order Multilinear Groups

We let G_1, G_2, …, G_k denote cyclic groups of order $N = p_1 p_2 \cdots p_r$, where p_1, …, p_r are distinct primes. We let G_T also denote a cyclic group of order N. We suppose that we have a k-linear map $E : G_1 \times G_2 \times \cdots \times G_k \to G_T$. We assume this is non-degenerate, meaning that if g_i generates

G_i for each i, then $E(g_1, g_2, \ldots, g_k)$ generates G_T. We write the group operations multiplicatively, and we let $1_{G_i}, 1_{G_T}$ denote the identity elements in each G_i and G_T respectively.

For each prime p_j dividing the group order of N, we have a subgroup G_{i,p_j} of order p_j inside each G_i. We let g_{i,p_j} denote a generator for G_{i,p_j}. We let G_{i,p_1p_2}, for example, denote the subgroup of order p_1p_2 inside G_i that is generated by $g_{i,p_1}g_{i,p_2}$.

These subgroups are "orthogonal" under E, meaning (for example) that if $h_1 \in G_{1,p_1p_2\cdots p_{i-1}p_{i+1}p_r}$, then for any $h_2 \in G_2, \ldots, h_{k-1} \in G_{k-1}$,

$$E(h_1, h_2, \ldots, h_{k-1}, g_{k,p_i}) = 1_{G_T}.$$

Each element of G_i can be expressed as $g_{1,p_1}^{\alpha_1} g_{1,p_2}^{\alpha_2} \cdots g_{1,p_r}^{\alpha_r}$. Thus if we have elements of $G_1, \ldots, G_k$ that are input to E, we can write them as $g_{1,p_1}^{\alpha_{1,1}} g_{1,p_2}^{\alpha_{2,1}} \cdots g_{1,p_r}^{\alpha_{r,1}}$, $\ldots$, $g_{k,p_1}^{\alpha_{1,k}} g_{k,p_2}^{\alpha_{2,k}} \cdots g_{k,p_r}^{\alpha_{r,k}}$, and by multilinearity of E and orthogonality we then have:

$$E(g_{1,p_1}^{\alpha_{1,1}} g_{1,p_2}^{\alpha_{2,1}} \cdots g_{1,p_r}^{\alpha_{r,1}}, \ldots, g_{k,p_1}^{\alpha_{1,k}} g_{k,p_2}^{\alpha_{2,k}} \cdots g_{k,p_r}^{\alpha_{r,k}}) = E(g_{1,p_1}^{\alpha_{1,1}}, \ldots, g_{k,p_1}^{\alpha_{1,k}}) \cdots E(g_{1,p_r}^{\alpha_{r,1}}, \ldots, g_{k,p_r}^{\alpha_{r,k}}). \quad (2)$$

We let $\mathcal{G}(\lambda, r, k)$ denote a group generation algorithm that takes in a security parameter λ, a desired number of prime factors r, and a desired level of multilinearity k and outputs a description of groups $G_1, \ldots, G_k, G_T$ as above. We assume the description includes a generator $g_i \in G_i$ for each i, the group order N, the primes $p_1, \ldots, p_r$ comprising N, and efficient algorithms for the group operations in each G_i, the group operation in G_T, and the multilinear map E. Note that with generator g_i for each full group plus knowledge of the prime factors, one can efficiently produce a generator for any subgroup of order dividing N inside any G_i.

Computational Assumption 1_{AC} Our first computational assumption in the asymmetric, composite-order setting is parameterized by the linearity, k, as well as r, the number of prime factors for each group order. Thus our groups $G_1, \ldots, G_k$ will each have order $N = p_1 \cdots p_r$, where $p_1, \ldots, p_r$ are distinct primes. For each $i > 1$, we will give out random generators g_{i,p_j} for each $j > 1$, as well as a random generator g_{i,p_1p_2} for the subgroup of order p_1p_2. For $i = 1$, we will give out random generators $g_{1,p_1}, \ldots, g_{1,p_r}$ for all prime order subgroups. The challenge term is a group element $T \in G_1$ that is either sampled uniformly at random from the order p_2 subgroup, or uniformly at random from the order p_1p_2 subgroup.

Computational Assumption 2_{AC} Our second computational assumption is parameterized by the linearity, k, as well as r, the number of prime factors for each group order. Thus our groups $G_1, \ldots, G_k$ will each have order $N = p_1 \cdots p_r$, where $p_1, \ldots, p_r$ are distinct primes. For each $i > 1$, we give out random generators g_{i,p_j} for each j. For G_1, we only give out generators $g_{1,p_2}, \ldots, g_{1,p_\ell}$. The challenge term is a group element $T \in G_2$ that is either sampled uniformly at random from the order p_1 subgroup of G_2 or is (an encoding) of 1_{G_2}.

Remark 3. *Since the groups $G_1, \ldots, G_k$ are distinct, we technically use families of these assumptions, where the special roles of groups 1 and 2 in our assumption statements are rotated around to all other pairs of groups.*

Intuition for generic security To see why assumption 1_{AC} holds in the asymmetric composite-order generic group model if it is hard to find a non-trivial factor of the group order, note that any attack in this setting must produce an algebraic relation that holds for one distribution of the challenge and fails for the other. To do this, one naturally needs to feed the challenge term as input in the multi-linear map, along with some other term that has a p_1 component but not a p_2 one. However, the only such term given is g_{1,p_1}, which cannot be combined with the challenge due to the asymmetry. Similarly, a violation of assumption 2_{AC} would require feeding the challenge into the multilinear map along with terms that all have p_1 components. But no such term is given in G_1.

6.2 Instantiating a Tribes Scheme

Suppose we wish to build a tribes scheme for $n \times \ell \times 2$ tribes matrices, and we have a generation algorithm $\mathcal{G}$ for producing asymmetric composite order multilinear groups. We construct a tribes scheme as follows:

Create(λ, M)**:** The creation algorithm takes in a security parameter λ and an $n \times \ell \times 2$ tribes matrix M (entries in $\{U, B\}$). It then calls $\mathcal{G}(\lambda, r = \ell, n)$ to produce a sequence of groups $G_1, \ldots, G_n$ of order $N = p_1 \cdots p_\ell$ equipped with an n-linear map E. It will produce $2n$ group elements, each indexed by a row $i \in [n]$ and a slot $\beta \in \{0, 1\}$. We let $h_{i,\beta}$ be sampled as follows. For each column index $j \in [\ell]$, if $M_{i,j,\beta} = B$, then an element z_j is sampled uniformly at random from the subgroup of order p_j inside G_i. If $M_{i,j,\beta} = U$, then $z_j := 1_{G_i}$. (All of these values are freshly resampled for each i, β.) We set:

$$h_{i,\beta} := \prod_{j=1}^{\ell} z_j.$$

The tribes scheme T consists of these $2n$ elements $\{h_{i,\beta}\}$ (we assume this implicitly includes a description of G that enables efficient computation of the group operation and E, and the full group order N, but *not* the individual primes comprising N).

Eval(T, x)**:** The evaluation algorithm takes in a tribes scheme T and a boolean vector $x = (x_1, \ldots, x_n) \in \{0, 1\}^n$. It computes $E(h_{1,x_1}, h_{2,x_2}, \ldots, h_{n,x_n})$ and checks whether this is equal to 1_{G_T} or not. If is it the identity, it outputs 0. Otherwise, it outputs 1.

Correctness We first establish that $Eval(Create(\lambda, M), x) = f_M(x)$. We first observe that (by orthogonality of distinct prime order subgroups) $E(h_{1,x_1}, h_{2,x_2}, \ldots, h_{n,x_n})$ can be considered as a product of the contributions within each prime order subgroup. For analyzing the contribution of a prime p_j, we consider two cases. Suppose that column j satisfied $M_{i,j,x_i} = B$ for all $i \in [n]$. This implies that $f_M(x) = 1$. In this case, there is a random component in the subgroup of order p_j incorporated in *every* h_{i,x_i}, so the contribution in G_T will be (with high probability) a non-identity element in the p_j order subgroup. This cannot be "canceled out" by a contribution in other prime order subgroup, so the final result will be $\neq 1_{G_T}$ in this case, matching f_M. In the other case, there is some i such that h_{i,x_i} *does not* have a component in the subgroup of order p_j inside G_i, and hence the contribution of this prime will be trivial in G_T. If this occurs for every j, the final output will be 1_{G_T}, matching f_M. Thus we have established correctness.

We now show that inter-column security for this tribes scheme is implied by our first computational assumption described above.

Lemma 6.1. *Assumption 1_{AC} implies inter-column security for this tribes scheme.*

Proof. For notational convenience, we consider the game with $i^* = 1$, $\beta = 0$, $j = 2$, $k = 1$ and any $n \times \ell \times 2$ tribes matrix M satisfying the game requirements for these indices. We assume we have an attacker $\mathcal{A}$ who obtains a non-negligible advantage in this game. We will employ the assumption with linearity n and $r = \ell$ prime factors. We suppose we are given g_{i,p_j} for each $i, j > 1$. We are also given g_{i,p_1p_2} for each $i > 1$ and $g_{1,p_1}, \dots, g_{1,p_\ell}$. Our challenge term is T, and we must guess if it is sampled from G_{1,p_2} or from G_{1,p_1p_2}.

We form the $2n$ group elements to give to $\mathcal{A}$ as follows. For each row $i \neq i^*$ and slot σ, we check the values of $M_{i,j',\sigma}$ in each column $j' \neq 1, 2$. For indices j' where these are B, we sample $z_{j'}$ to be a random element of the subgroup of order $p_{j'}$ inside G_i (note that we can do this by raising the appropriate generator to a random exponent modulo N). For values where these are U, we set $z_{j'} := 1_{G_i}$. There are a few cases for the values of $M_{i,1,\sigma}$ and $M_{i,2,\sigma}$. In all cases, we choose a random exponent α modulo N. If both are B, we set $z_1 z_2 := g^{\alpha}_{i,p_1p_2}$. If both are U, we set $z_1 z_2 := 1_{G_i}$. If If $M_{i,2,b} = B$ and $M_{i,1,b} = U$, we set $z_1 z_2 := g^{\alpha}_{i,p_2}$. We note that the case of $M_{i,1,b} = B$ and $M_{i,2,b} = U$ is disallowed by the game requirements. This allows us to compute:

$$h_{i,\sigma} := z_1 z_2 z_3 \cdots z_\ell,$$

which is properly distributed.

To form $g_{1,1}$, we can sample each z_j as a random element of $G_{1,p_{j'}}$ if $M_{1,j',1} = B$, and as 1_{G_1} otherwise. We can then compute: $h_{1,1} := z_1 z_2 \cdots z_\ell$, which is properly distributed. To form $g_{1,0}$, we sample each $z_3, \dots, z_\ell$ as before, as then set $h_{1,0} := T z_3 \cdots z_\ell$. If T was sampled as a random element of the subgroup of order p_1p_2 inside G_1, this is properly distributed for the case $M_{n,1,0} = M_{n,2,0} = B$. If T was sampled as a random element of the subgroup of order p_2, this is properly distributed for the case $M_{n,1,0} = U$ and $M_{n,2,0} = B$. Thus, we can leverage $\mathcal{A}$'s non-negligible advantage in the inter-column security game to obtain a non-negligible advantage against our Assumption 1_{AC}. $\square$

We last show that intra-column security for this tribes scheme is implied by our second computational assumption.

Lemma 6.2. *Assumption 2_{AC} on $\mathcal{G}$ implies intra-column security for this tribes scheme.*

Proof. We will prove this for a version of the intra-column security game where we have an $n \times \ell \times 2$ tribes matrix M and we are considering the first column, with both slots in the first row taking the value U. It will be our goal to change the value of a single slot in the second row (say slot 0). Observe that this implies the seemingly more general notion of intra-column security via a standard hybrid argument.

We will employ our Assumption 2_{AC} with linearity n and $r = \ell$ prime factors. We suppose we are given random generators g_{i,p_j} for each j for each $i > 1$. For G_1, we are also given generators $g_{1,p_2}, \dots, g_{1,p_\ell}$. The challenge term $T \in G_2$ is either sampled uniformly at random from the order p_1 subgroup of G_2 or is (an encoding) of 1_{G_2}, and it our task to guess which.

We form the $2n$ group elements to give to $\mathcal{A}$ as follows. For each row $i > 3$ and slot σ, we check the values of $M_{i,j',\sigma}$ in each column j'. For indices j' where these are B, we sample $z_{j'}$ to be a random element of the subgroup of order $p_{j'}$ inside G_i (note that we can do this by raising the appropriate generator to a random exponent modulo N). This allows us to compute $h_{i,\sigma} = z_1 z_2 \cdots z_\ell$. For $i = 1$ and slot σ, we similarly check the values of $M_{i,j',\sigma}$ for each column

$j' > 1$, and whenever we have a B value, we sample $z_{j'}$ to a random element of the subgroup of order $p_{j'}$ in G_1, and define $z_{j'} = 1_{G_1}$ otherwise. We then set $h_{1,\sigma} = z_2 \cdots z_\ell$.

Now we consider the group elements for row 2. For the slot 1 element, we sample $z_1, \ldots, z_\ell$ according to the entries of M as above, and we form $h_{2,1} = z_1 z_2 \ldots z_\ell$. For the changing slot 0, we sample $z_1, z_3, \ldots, z_\ell$ as before, but define $z_2 := T$ and form $h_{2,0} = z_1 z_2 z_3 \ldots z_\ell$. If T was sampled from the order p_1 subgroup, this will be distributed as if $M_{2,1,0} = B$. Otherwise, it will be distributed as if $M_{2,1,0} = U$. □

7 Instantiating Our Scheme in Asymmetric Prime-Order Multilinear Groups: An Eigenvector-Based Approach

In the setting of bilinear maps, it is often the case that schemes initially described over composite-order bilinear groups can be "converted" to prime-order groups (e.g. [11, 21, 23, 20, 22, 26] and many more). This conversion typically increases the conceptual and algorithmic complexity of the scheme, but allows the use of more "natural" prime-order groups, and often simplifies the underlying computational assumptions.

Here, we show how to convert the asymmetric composite-order construction from Section 6 to prime-order asymmetric multilinear groups (or, more generally, groups of arbitrary order). We base its security on assumptions that (unlike the assumptions for the composite-order schemes) remain plausible even if factoring is easy. Moreover, the assumptions depend on n (the number of variables in the 3CNF), but not the number of clauses ℓ, which often will be a big improvement.

At the moment, this conversion is mainly of theoretical interest, since our assumption here is false for GGH encodings – indeed all decisional "source-group assumptions" are false for GGH encodings due to their susceptibility to "weak discrete log attacks" – leaving CLT encodings as currently the only construction of multilinear maps that we can use here, and CLT encodings *already* inherently provide a composite-order encoding space, and their security relies on the hardness of factoring. However, we believe it is quite likely that, in the future, alternative multilinear maps will be constructed that support source-group assumptions and do not rely on the hardness of factoring; in that event, our construction here may become useful.

Eigensystems over Asymmetric Prime Order Multilinear Groups We let $G_1, \ldots, G_k$ and G_T denote (cyclic) groups of prime order p. We suppose that we have a k-linear map $E : G_1 \times \ldots \times G_k \to G_T$. We assume this is non-degenerate, meaning that if g_i generates G_i for each i, then $g_T \leftarrow E(g_1, g_{,2} \ldots, g_k)$ generates G_T. We write the group operations multiplicatively, and we let $1_{G_i}, 1_{G_T}$ denote the identity elements in each G_i and G_T respectively.

In the prime-order setting, group elements will be replaced with matrices of group elements. For a matrix M, we use g_i^M to denote encodings of the individual entries of M in a group G_i, and we apply the multilinear map to matrices of encodings in the natural way, when we take one encoding from each G_i. (The multilinear map then computes the product of the matrices "in the exponent").

In the composite-order setting, a particular prime order subgroup of some G_i was either "blocked" or "unblocked", meaning that the component in that subgroup was either randomly distributed or set to the identity. The notion of these subgroups is replaced by subspaces that are either blocked or unblocked, meaning that the rank of our matrices will change as we apply our computational assumptions. As we used ℓ subgroups inside each G_i in the composite-order setting, we will now have ℓ subspaces that can be present or absent in the matrices corresponding to each G_i. When each subspace is blocked in at least one of G_i, the product of the resulting matrices will be all 0s, otherwise not. We will use more than ℓ dimensions in order to create space for extra randomization.

The multilinear map can be used to "zeroize" subspaces in a way that is analogous to the composite-order setting. For example, consider $d \times d$ matrices over $\mathbb{Z}_p$ (for some positive integer d) of the following form. Let $D_1, D_2, \ldots, D_k$ be diagonal matrices sampled from some joint distribution that guarantees that for each $j \in [d]$, at least one of D_i will have a 0 in its j^{th} diagonal entry. Let $R_0, R_1, \ldots, R_k$ be uniformly random $d \times d$ matrices over $\mathbb{Z}_p$. Then:

$$E(g_1^{R_0 D_1 R_1^{-1}}, g_2^{R_1 D_2 R_2^{-1}}, \ldots, g_k^{R_{k-1}^{-1} D_k R_k}) = g_T^{0^{d\times d}} . \tag{3}$$

This follows from the fact that

$$R_0 D_1 R_1^{-1} R_1 D_2 R_2^{-1} R_{k-1}^{-1} D_k R_k = R_0 D_1 D_2 \cdots D_k R_k = R_0 (0^{d\times d}) R_k.$$

We let $\mathcal{G}(\lambda, k)$ denote a group generation algorithm that takes in a security parameter λ and a desired level of multilinearity k and outputs a description of groups $G_1, \ldots, G_k, G_T$ as above. We assume the description includes a generator $g_i \in G_i$ for each i, the common group order p, and efficient algorithms for the group operation in each G_i, the group operation in G_T, and the multilinear map E.

7.1 Computational Assumptions

We describe computational assumptions that will serve as the analogs of our assumptions for asymmetric composite-order groups. We discuss the plausibility of these assumptions in detail in Section C.

Like the assumptions for composite-order groups, our assumptions here are parameterized by a positive integer n indicating the level of multilinearity. They concern a multilinear group description output by $\mathcal{G}(\lambda, n)$ for security parameter λ and multilinearity n. We let p be the order of the group.

Assumption 1_{AP}**:** Set $d = 4$. Generate random matrices $R_0, \ldots, R_n$ in $\mathbb{Z}_p^{d\times d}$. Sample diagonal matrices $C_1^1, D_1^1, C_1^2, D_1^2$ as follows. When all sub/super-scripts are matching, C's and D's will always denote two independent samples from the same distribution (these will be used to supply sufficient randomness to give out the group elements for both slots when we instantiate our tribes scheme). For C_1^1, D_1^1, the first entries are random, while the remaining entries are 0. For C_1^2, D_1^2, the second entries are random and the remaining entries are 0.

We additionally sample diagonal matrices $C_i^0, D_i^0, C_i^2, D_i^2, C_i^{1,2}, D_i^{1,2}$ for each $i \geq 2$ as follows. The last two entries are always randomly distributed. The first two entries are both 0 when the superscript is 0. When the superscript is 2, the second entry is random and the first entry remains 0. When the superscript is 1,2, both of these are random.

Two final matrices $T^2, T^{1,2}$ are also sampled, where the last two entries of each are 0 and the second entry of each is random, but the first entry is 0 in T^2 and is random in $T^{1,2}$.

An instance of the assumption includes a description of the multilinear group and:

$g_i^{R_{i-1} C_i^0 R_i^{-1}}, g_i^{R_{i-1} D_i^0 R_i^{-1}}, g_i^{R_{i-1} C_i^2 R_i^{-1}}, g_i^{R_{i-1} D_i^2 R_i^{-1}} \; g_i^{R_{i-1} C_i^{1,2} R_i^{-1}}, g_i^{R_{i-1} D_i^{1,2} R_i^{-1}}$ for $i = 2, \ldots, n$

$g_1^{R_0 C_1^0 R_1^{-1}}, g_1^{R_0 D_1^0 R_1^{-1}}, g_1^{R_0 C_1^1 R_1^{-1}}, g_1^{R_0 D_1^1 R_1^{-1}}, g_1^{R_0 C_1^2 R_1^{-1}}, g_1^{R_0 D_1^2 R_1^{-1}}$

Either $g_1^{R_0 T^2 R_1^{-1}}$ or $g_1^{R_0 T^{1,2} R_1^{-1}}$

The task is to distinguish which distribution the final encoded term comes from.

Assumption 2_{AP}: Set $d = 5$. Generate random matrices $R_0, \ldots, R_n$ in $\mathbb{Z}_p^{d\times d}$. Sample diagonal matrices $C_1, D_1, C_2^U, D_2^U, C_2^B, D_2^B, \ldots, C_n^U, D_n^U, C_n^B, D_n^B$ in $\mathbb{Z}_p^{d\times d}$ as follows. The C, D matrices with the same subscripts and superscripts will always represent two independent samples from the same distribution (these will be used to supply sufficient randomness to give out the group elements for both slots when we instantiate our tribes scheme).

The first entries of C_1, D_1 will be set to 0. For the rest, the first entry of each C_i^U, D_i^U will be distributed as 0 and the first entry of each C_i^B, D_i^B will be distributed randomly in $\mathbb{Z}_p$. The remaining 4 entries are sampled as follows. For C_1, D_1, the 1st of these 4 is set randomly, while the remaining three are fixed to 0. For each C_i, D_i with $i > 1$, the first of these four entries will be fixed to 0, while the remaining 3 will be set randomly. (note it is the same for all of C_i^U, D_i^U and C_i^B, D_i^B). Two more matrices T_2^U, T_2^B are also sampled, distributed the same as D_2^U, D_2^B (though sampled independently).

An instance of the assumption includes a description of the multilinear group and:

$$g_i^{R_{i-1}C_i^U R_i^{-1}}, g_i^{R_{i-1}D_i^U R_i^{-1}}, g_i^{R_{i-1}C_i^B R_i^{-1}}, g_i^{R_{i-1}D_i^B R_i^{-1}} \text{ for } i = 2, \ldots, n$$
$$g_1^{R_0 C_1 R_1^{-1}}, g_1^{R_0 D_1 R_1^{-1}}$$
$$\text{Either } g_2^{R_1 T_2^U R_2^{-1}} \text{ or } g_2^{R_1 T_2^B R_2^{-1}}$$

The task is to distinguish which distribution the final encoded term comes from.

Remark 4. *As was the case in Section 6, since the groups $G_1, \ldots, G_k$ are distinct, we technically use families of these assumptions, where the special roles of groups 1 and 2 in our assumption statements are rotated around to all other pairs of groups.*

We now define expanded versions of these assumptions with additional dimensions that allow us to directly embed these into the security proof for our scheme. We will then show a simple reduction that establishes the equivalence of the expanded assumptions. Intuitively, carrying around extra dimensions whose distributions are not changing is easy to do, as you can pick a random embedding of the 4 or 5 challenge dimensions into the larger space and use this knowledge to form proper distributions in the dimensions not relevant to the challenge.

To simplify the description of our expanded assumptions, we define a distribution $Diag(i)$ over $\mathbb{Z}_p^{4n}$ for each $i \in n$ as follows. The $4n$ coordinates are divided into segments of length 4. For the i^{th} segment, the first entry is randomly distributed in $\mathbb{Z}_p$, and the rest are 0. For all other segments, the first coordinate is set to 0, and the remaining three coordinates are distributed randomly in $\mathbb{Z}_p$.

These assumptions are additionally parametrized by a positive integer ℓ.

Assumption $1'_{AP}$: Set $d = \ell(1 + 4n)$. Generate random matrices $R_0, \ldots, R_n$ in $\mathbb{Z}_p^{d\times d}$. Sample diagonal matrices $C_1^0, D_1^0, C_1^1, D_1^1, C_1^2, D_1^2$ and $C_i^0, D_i^0, C_i^2, D_i^2, C_i^{1,2}, D_i^{1,2}$ for each $i \geq 2$ as follows. Divide the d diagonal entries into ℓ blocks. In the first block, the first entry is distributed randomly whenever 1 appears as a superscript. The remaining entries are distributed according to $Diag(i)$. In the second block, the first entry is distributed randomly whenever 2 appears as a superscript. The remaining entries are distributed according to $Diag(i)$. In the remaining blocks, all entries are set to 0. Sample T^2 and $T^{1,2}$ analogously.

For each $j > 2$, sample additional diagonal matrices $W^U_{1,j}, V^U_{1,j}, W^B_{1,j}, V^B_{1,j}, \dots, W^U_{n,j}, V^U_{n,j}, W^B_{n,j}, V^B_{n,j}$ in $\mathbb{Z}_p^{d\times d}$ as follows. Each V will be independently sampled from the same distribution as the corresponding W. In the j^{th} block, each $W^U_{i,j}$ has the first entry set to 0. Each $W^B_{i,j}$ has this entry sampled uniformly at random from $\mathbb{Z}_p$. The remaining $4n$ entries are sampled from $Diag(i)$. In the other blocks (not equal to j), all diagonal entries are set to 0.

An instance of the assumption includes a description of the multilinear group and:

$$
\begin{array}{l}
g_i^{R_{i-1}C_i^0R_i^{-1}}, g_i^{R_{i-1}D_i^0R_i^{-1}}, g_i^{R_{i-1}C_i^2R_i^{-1}}, g_i^{R_{i-1}D_i^2R_i^{-1}} g_i^{R_{i-1}C_i^{1,2}R_i^{-1}}, g_i^{R_{i-1}D_i^{1,2}R_i^{-1}} \text{ for } i = 2, \dots, n \\
g_1^{R_0C_1^0R_1^{-1}}, g_1^{R_0D_1^0R_1^{-1}}, g_1^{R_0C_1^1R_1^{-1}}, g_1^{R_0D_1^1R_1^{-1}}, g_1^{R_0C_1^2R_1^{-1}}, g_1^{R_0D_1^2R_1^{-1}} \\
g_i^{R_{i-1}W^U_{i,j}R_i^{-1}}, g_i^{R_{i-1}V^U_{i,j}R_i^{-1}}, g_i^{R_{i-1}W^B_{i,j}R_i^{-1}}, g_i^{R_{i-1}V^B_{i,j}R_i^{-1}} \text{ for } i = 1, \dots, n, j = 3, \dots, \ell \\
\text{Either } g_1^{R_0T^2R_1^{-1}} \text{ or } g_1^{R_0T^{1,2}R_1^{-1}}
\end{array}
$$

The task is to distinguish which distribution the final encoded term comes from.

Assumption $2'_{AP}$: Set $d = \ell(1 + 4n)$. Generate random matrices $R_0, \dots, R_n$ in $\mathbb{Z}_p^{d\times d}$. Sample diagonal matrices $C_1, D_1, C^U_2, D^U_2, C^B_2, D^B_2, \dots, C^U_n, D^U_n, C^B_n, D^B_n$ in $\mathbb{Z}_p^{d\times d}$ as follows. We divide the $\ell(1 + 4n)$ diagonal coordinates into ℓ blocks each of length $1 + 4n$. In the first block, the first five entries will be distributed identically to Assumption 2. The remaining entries will be distributed according to $Diag(i)$. In the remaining blocks, all entries are fixed to 0. Matrices T^U_2, T^B_2 are then (independently) sampled from the same distributions used for C^U_2, C^B_2, respectively.

For each $j > 1$, sample additional diagonal matrices $W^U_{1,j}, V^U_{1,j}, W^B_{1,j}, V^B_{1,j}, \dots, W^U_{n,j}, V^U_{n,j}, W^B_{n,j}, V^B_{n,j}$ in $\mathbb{Z}_p^{d\times d}$ as follows. Each V will be independently sampled distribution as the corresponding W. In the j^{th} block, each $W^U_{i,j}$ has the first entry set to 0. Each $W^B_{i,j}$ has this entry sampled uniformly at random from $\mathbb{Z}_p$. The remaining $4n$ entries are sampled from $Diag(i)$. In the other blocks (not equal to j), all diagonal entries are set to 0.

An instance of the assumption includes a description of the multilinear group and:

$$
\begin{array}{l}
g_i^{R_{i-1}C^U_iR_i^{-1}}, g_i^{R_{i-1}D^U_iR_i^{-1}}, g_i^{R_{i-1}C^B_iR_i^{-1}}, g_i^{R_{i-1}D^B_iR_i^{-1}} \text{ for } i = 2, \dots, n \\
g_1^{R_0C_1R_1^{-1}}, g_1^{R_0D_1R_1^{-1}} \\
g_i^{R_{i-1}W^U_{i,j}R_i^{-1}}, g_i^{R_{i-1}V^U_{i,j}R_i^{-1}}, g_i^{R_{i-1}W^B_{i,j}R_i^{-1}}, g_i^{R_{i-1}V^B_{i,j}R_i^{-1}} \text{ for } i = 1, \dots, n, j = 2, \dots, \ell \\
\text{Either } g_2^{R_1T^U_2R_2^{-1}} \text{ or } g_2^{R_1T^B_2R_2^{-1}}
\end{array}
$$

The task is to distinguish which space the final encoded term comes from.

Lemma 7.1. *Assumption 1_{AP} implies Assumption $1'_{AP}$ and Assumption 2_{AP} implies Assumption $2'_{AP}$.*

Proof. We argue this for assumptions 2_{AP} and $2'_{AP}$, as the notation is a bit cleaner in this case. We show that an algorithm that can break the assumption $2'_{AP}$ can be used to break the simpler assumption 2_{AP}. Let $r = 5$ and $r' = \ell(1 + 4n)$. Let $R_0, R_1, \dots, R_n \in \mathbb{Z}_p^{r\times r}$ be as in the simpler assumption. Let $R^*_i \in \mathbb{Z}_p^{r'\times r'}$ for each i be a matrix with R_i in the upper left, the identity matrix $I_{r'-r} \in \mathbb{Z}_p^{r'-r\times r'-r}$ in the lower-right, and zeros elsewhere. Let $Z_i \in \mathbb{Z}_p^{r'\times r'}$ be a random invertible matrix for each i, and let $R'_i = Z_iR^*_i$. Now from an instance of the simpler assumption, it is clear that we can use our knowledge of the Z_i's and encodings of 0 and 1 in each subgroup G_i to generate a well-distributed instance of the expanded assumption. The distinguisher's answer for the initial assumption can be used directly as the answer for the second assumption.

The argument for assumptions 1_{AP} and $1'_{AP}$ is analogous, just with a slightly more complex shifting of the diagonal coordinates. Note that the 4 challenge dimensions that we must embed into the $r' = \ell(1 + 4n)$ dimensional space to simulate assumption $1'_{AP}$ can be mapped as follows. The first diagonal position of the challenge will play the role of the the first coordinate in r', the second will play the role of the first coordinate in the second block of length $1+4n$, and the remaining two challenge coordinates can be taken e.g. as the 2 and 3rd positions on the length r' diagonals. $\square$

7.2 Instantiating a Tribes Scheme

Suppose we wish to build a tribes scheme for $n \times \ell \times 2$ tribes matrices (not to be confused with the encoded matrices that we will use in the cryptosystem), and we have a generation algorithm $\mathcal{G}$ for producing asymmetric multilinear groups. We construct a tribes scheme as follows:

$Create(\lambda, M)$**:** The creation algorithm takes in a security parameter λ and an $n \times \ell \times 2$ tribes matrix M (entries in $\{U, B\}$). It then calls $\mathcal{G}(\lambda, n)$ to produce groups $G_1, \ldots, G_n, G_T$ of prime order p, equipped with an n-linear map E. We let $d = \ell(1 + 4n)$. It will produce $2n$ encoded matrices of size $d \times d$, each indexed by a row $i \in [n]$ and a slot $\beta \in \{0, 1\}$.

It begins by sampling random matrices $R_0, \ldots, R_n \in Z_p^{d\times d}$. (These will be used for both β values.) For each slot β, it then samples diagonal matrices $D_1^\beta, \ldots, D_n^\beta$ as follows. The d diagonal entries of each D_i^β are considered as ℓ blocks of length $1+4n$. The distribution of block j inside D_i^β will depend on the entry $M_{i,j,\beta}$. If $M_{i,j,\beta} = B$, then the first coordinate of the block is sampled as a random element of $\mathbb{Z}_p$. Otherwise, it is set to 0. The remaining $4n$ coordinates are distributed as $Diag(i)$. (Recall this means they are further divided into segments of length 4. For the i^{th} segment, the first entry is randomly distributed in $\mathbb{Z}_p$, and the rest are 0. For all other segments, the first coordinate is set to 0, and the remaining three coordinates are distributed randomly in $\mathbb{Z}_p$.) Note that all these segments are distributed merely as a function of i, and only the first of the $1 + 4n$ diagonal entries of block j inside D_i^β actually depend on the relevant entry of M. These extra $4n$ dimensions are useful randomization for proving that our computational assumptions hold in the (asymmetric) generic group model.

We then define $A_{i,\beta} = R_{i-1} D_i^\beta R_i^{-1}$ for each i, β, and the tribes scheme T consists of the $2n$ encoded matrices $\{h_{i,\beta} := g_i^{A_{i,\beta}}\}$. (We assume this implicitly includes a description of $G_1, \ldots, G_n$ that enables efficient computation of the group operation and E, and the group order p.)

$Eval(T, x)$**:** The evaluation algorithm takes in a tribes scheme T and a boolean vector $x = (x_1, \ldots, x_n) \in \{0, 1\}^n$. It computes $E(g_1^{A_{1,x_1}}, g_2^{A_{2,x_2}}, \ldots, g_n^{A_{n,x_n}})$ and checks whether this is equal to $g_T^{0^{d\times d}}$ or not. If so, it outputs 0. Otherwise, it outputs 1.

Remark 5. *We emphasize that, in the construction, after the initial parameters are set, the parties (encrypter and decrypter) do not need to generate encodings of* particular *values, but rather only* random *values. Generating encodings of random values is possible with GGH and CLT encodings.*

Correctness We establish that $Eval(Create(\lambda, M), x) = f_M(x)$. We first observe that, by Equation 3, the value $E(g_1^{A_{1,x_1}}, g_2^{A_{2,x_2}}, \ldots, g_n^{A_{n,x_n}})$ will be equal to $g_T^{0^{d\times d}}$ if and (whp) only if for every $v \in [\ell]$, some $D_i^{x_i}$ is set to 0 in that entry. We consider the indices v in blocks of length $1 + 4n$. Note that in the j^{th} block, this property is satisfied for the last $4n$ positions just by construction, regardless of M, x. However, to get the first coordinate to be 0 in D_i, we must have $M_{i,j,x_i} = U$. Therefore the output matches f_M.

Security We now show that inter-column and intra-column security for this tribes scheme are implied by our computational assumptions described above.

Lemma 7.2. *Our computational assumption $1'_{AP}$ on $\mathcal{G}$ implies inter-column security for this tribes scheme.*

Proof. For notational convenience, we consider the game with $i^* = 1$, $\beta = 0$, $j = 2$, $k = 1$ and any $n \times \ell \times 2$ tribes matrix M satisfying the game requirements for these indices. We assume we have an attacker $\mathcal{A}$ who obtains a non-negligible advantage in this game. We will employ the assumption with linearity n. Observe that our design of the matrix distributions in our construction is symmetric with respect to the shifting roles of groups and blocks, so our assumption as stated can be equally applied to other parameterizations of the inter-column game by simply relabeling these.

We suppose we are given $g_i^{R_{i-1}C_i^0R_i^{-1}}, g_i^{R_{i-1}D_i^0R_i^{-1}}, g_i^{R_{i-1}C_i^2R_i^{-1}}, g_i^{R_{i-1}D_i^2R_i^{-1}}, g_i^{R_{i-1}C_i^{1,2}R_i^{-1}}, g_i^{R_{i-1}D_i^{1,2}R_i^{-1}}$ for $i = 2, \ldots, n$, $g_1^{R_0C_1^0R_1^{-1}}, g_1^{R_0D_1^0R_1^{-1}}, g_1^{R_0C_1^1R_1^{-1}}, g_1^{R_0D_1^1R_1^{-1}}, g_1^{R_0C_1^2R_1^{-1}}, g_1^{R_0D_1^2R_1^{-1}}, g_i^{R_{i-1}W_{i,j}^UR_i^{-1}}$, $g_i^{R_{i-1}V_{i,j}^UR_i^{-1}}, g_i^{R_{i-1}W_{i,j}^BR_i^{-1}}, g_i^{R_{i-1}V_{i,j}^BR_i^{-1}}$ for $i = 1, \ldots, n$, $j = 3, \ldots, \ell$, and a challenge term $g_1^{R_0TR_1^{-1}}$ where T is either distributed as T^2 or $T^{1,2}$, and it is our task to determine which.

We form the $2n$ group elements to give to $\mathcal{A}$ as follows. For each row $i \neq 1$ and slot $\sigma = 0$, we check the values of $M_{i,j',\sigma}$ in each column $j' \neq 1, 2$. For indices j' where these are B, we set $z_{j'}$ equal to $g_i^{R_{i-1}W_{i,j'}^BR_i^{-1}}$. For indices j' where these are U, we set $z_{j'}$ equal to $g_i^{R_{i-1}W_{i,j}^UR_i^{-1}}$.

There are a few cases for the values of $M_{i,1,\sigma}$ and $M_{i,2,\sigma}$. If these are both B, we set $z_1z_2 = g_i^{R_{i-1}C_i^{1,2}R_i^{-1}}$. If these are both U, we set $z_1z_2 = g_i^{R_{i-1}C_i^0R_i^{-1}}$. If $M_{i,1,\sigma} = U$ and $M_{i,2,\sigma} = B$, we set $z_1z_2 = g_i^{R_{i-1}C_i^2R_i^{-1}}$. We note that the case of $M_{i,1,\sigma} = B$ and $M_{i,2,\sigma} = U$ is disallowed by the game requirements. This allows us to compute:

$$h_{i,\sigma} := z_1z_2z_3\cdots z_\ell,$$

which is properly distributed. For the $\sigma = 1$ slots, we perform the same procedure, just using V's and D's in placed of W's and C's to obtain independent samples.

To form $h_{1,1}$, we can sample each $z_{j'}$ for $j' \geq 3$ as above. If $M_{1,1,1} = B$, we sample z_1 as $g_1^{R_0C_1^1R_1^{-1}}$. Otherwise, we sample it as $g_1^{R_0C_1^0R_1^{-1}}$. If $M_{1,2,1} = B$, we sample z_2 as $g_1^{R_0C_1^2R_1^{-1}}$, otherwise we sample it as $g_1^{R_0D_1^0R_1^{-1}}$. We can then compute: $h_{1,1} := z_1z_2\cdots z_\ell$, which is properly distributed.

To form $h_{1,0}$, we sample each $z_3, \ldots, z_\ell$ as before, and then set $h_{1,0} := g_1^{R_0TR_1^{-1}} z_3\cdots z_\ell$. If T was sampled as $T^{1,2}$, this is properly distributed for the case $M_{1,1,0} = M_{1,2,0} = B$. If T was sampled as T^2, this is properly distributed for the case $M_{1,1,0} = U$ and $M_{1,2,0} = B$. Thus, we can leverage $\mathcal{A}$'s non-negligible advantage in the inter-column security game to obtain a non-negligible advantage against our Assumption $1'_{AP}$.

□

Lemma 7.3. *Our computational assumption $2'_{AP}$ on $\mathcal{G}$ implies intra-column security for this tribes scheme.*

Proof. We will prove this for a version of the intra-column security game where we have an $n \times \ell \times 2$ tribes matrix M and we are considering the first column, with both slots in the first row taking the value U. It will be our goal to change the value of a single slot in the second row (say slot 0). Observe that this implies the seemingly more general notion of intra-column security via a standard hybrid argument. Again observe that our design of the matrix distributions in our construction is symmetric with respect to the shifting roles of groups and blocks, so our assumption as stated can be equally applied to other parameterizations of the intra-column game by simply relabeling these.

We suppose we are given $g_i^{R_{i-1}C_i^U R_i^{-1}}$, $g_i^{R_{i-1}D_i^U R_i^{-1}}$, $g_i^{R_{i-1}C_i^B R_i^{-1}}$, $g_i^{R_{i-1}D_i^B R_i^{-1}}$ for $i = 2, \dots, n$, $g_1^{R_0 C_1 R_1^{-1}}$, $g_1^{R_0 D_1 R_1^{-1}}$, $g_i^{R_{i-1}W_{i,j}^U R_i^{-1}}$, $g_i^{R_{i-1}V_{i,j}^U R_i^{-1}}$, $g_i^{R_{i-1}W_{i,j}^B R_i^{-1}}$, $g_i^{R_{i-1}V_{i,j}^B R_i^{-1}}$ for $i = 1, \dots, n$, $j = 2, \dots, \ell$ and a challenge term $g_2^{R_1 T_2 R_2^{-1}}$, and our goal is to distinguish between $T_2 = T_2^U$ and $T_2 = T_2^B$.

We form the $2n$ group elements to give to $\mathcal{A}$ as follows. For each row $i \neq 2$ and slot $\sigma = 0$, we check the values of $M_{i,j,\sigma}$ in each column j. For indices j where these are B, we set z_j to be $g_i^{R_{i-1}W_{i,j}^B R_i^{-1}}$. For indices j where these are U, we set z_j to be $g_i^{R_{i-1}W_{i,j}^U R_i^{-1}}$. This applies in all cases except for when $i = 1$ and $j = 1$. In this case, note we must have $M_{1,1,\sigma} = U$ for either value of σ, so we can set $z_1 = g_1^{R_0 C_1 R_1^{-1}}$. We can then compute

$$h_{i,\beta} = \prod_{j=1}^{\ell} z_j.$$

For rows $i \neq 2$ and slots $\sigma = 1$, we do the same procedure, except using the analogous terms with the D, V matrices instead of the C, W's. Then ensures that all of these are properly distributed without unwanted correlations between the slots.

Now we consider the group elements for row 2. For the slot 1 element, we sample $z_1, \dots, z_\ell$ according to the entries of M as above, and we form $h_{2,1} = z_1 z_2 \dots z_\ell$. For the changing slot 0, we sample $z_1, z_3, \dots, z_\ell$ as before, but define $z_2 = g_2^{R_1 T_2 R_2^{-1}}$ and form $h_{2,0} = z_1 z_2 z_3 \dots z_\ell$. If T_2 was sampled as T_2^B, this will be distributed as if $M_{2,1,0} = B$. Otherwise, it will be distributed as if $M_{2,1,0} = U$.

□

Acknowledgements

We thank Mihir Bellare and Amit Sahai for helpful discussions and comments.

References

[1] Mihir Bellare and Viet Tung Hoang. Adaptive witness encryption and asymmetric password-based cryptography. Cryptology ePrint Archive, Report 2013/704, 2013. `http://eprint.iacr.org/`.

[2] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *FOCS*, pages 408–416, 1985.

[3] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. extended abstract in Crypto 2001.

[4] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, 2006.

[5] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[6] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM Conference on Computer and Communications Security*, pages 211–220, 2006.

[7] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, pages 257–270, 1994.

[8] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[9] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[10] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1993.

[11] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013.

[14] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128, 2013. `http://eprint.iacr.org/`.

[15] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

[16] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

[17] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits. In *STOC*, 2013.

[18] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[19] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[20] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.

[21] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[22] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, pages 180–198, 2012.

[23] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[24] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. Cryptology ePrint Archive, Report 2013/781, 2013. `http://eprint.iacr.org/`.

[25] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[26] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In *TCC*, pages 133–150, 2012.

[27] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

A Witness Encryption

We give the definition of witness encryption. paper. The definition here follows the original of Garg, Gentry, Sahai, and Waters [15], but with two modifications. First, we restrict ourselves to perfect correctness for simplicity. Second, in defining soundness security we use a notation that the scheme is secure if for all PPT attackers there exists a negligible function $\text{negl}(\cdot)$ such that for any $x \notin L$ the attacker must only be able to distinguish encryption with probability at most $\text{negl}(\lambda)$. The GGSW definition had a different ordering of quantifiers which allowed the bounding negligible function for a particular attacker to depend on the instance x. Bellare and Tung Hoang [1] showed that this formulation was problematic for multiple applications of witness encryption. We further require that both the message length and the problem statement length must be bounded by some polynomial of the security parameter.

A *witness encryption* scheme for an ***NP*** language L (with corresponding witness relation R) consists of the following two polynomial-time algorithms:

> ***Encryption.*** The algorithm $\mathit{Encrypt}_{\mathrm{WE}}(1^\lambda, x, m)$ takes as input a security parameter 1^λ, an unbounded-length string x, and a message $m \in \mathcal{M}$ for some message space $\mathcal{M}$, and outputs a ciphertext CT.
>
> ***Decryption.*** The algorithm $\mathit{Decrypt}_{\mathrm{WE}}(\mathrm{CT}, w)$ takes as input a ciphertext CT and an unbounded-length string w, and outputs a message m or the symbol $\perp$.

These algorithms must satisfy the following correctness condition:

Definition A.1 (Correctness of Witness Encryption)**.** *For any security parameter λ, for any $m \in \mathcal{M}$, and for any $x \in L$ such that $R(x, w)$ holds, we have that*

$$\Pr\left[\mathit{Decrypt}_{\mathrm{WE}}\big(\mathit{Encrypt}_{\mathrm{WE}}(1^\lambda, x, m), w\big) = m\right] = 1$$

Soundness Security We model soundness security for a witness encryption scheme for a language L that is parameterized as for an instance x and two equal length messages m_0, m_1. We define the (parameterized) advantage of an attacker as

$$\mathsf{WE\,Adv}_{\mathcal{A},x,m_0,m_1}(\lambda) = \Pr[\mathcal{A}(\mathit{Encrypt}_{\mathrm{WE}}(1^\lambda, x, m_1)) = 1] - \Pr[\mathcal{A}(\mathit{Encrypt}_{\mathrm{WE}}(1^\lambda, x, m_0)) = 1]$$

Definition A.2 (Soundness Security of Witness Encryption)**.** *We say that a witness encryption scheme for a language L with witness relation $R(\cdot,\cdot)$ is secure if for any probabilistic poly-time attack algorithm $\mathcal{A}$ there exists a negligible function in the security parameter $negl(\cdot)$ such that for all $x \notin L$ and equal length messages m_0, m_1 we have $\mathsf{WE\,Adv}_{\mathcal{A},x,m_0,m_1}(\lambda) \leq negl(\lambda)$.*

B Implementing Our Schemes with Current Multilinear Maps

We have described witness encryption schemes that can be built from multilinear maps. Now, we describe how to adapt the schemes to graded encoding systems – in particular, the graded encoding system of Coron, Lepoint and Tibouchi (CLT) [9]. We describe how to handle the "noise" issues that arise, and also discuss how our use of subrings of the encoding space impacts security and requires (minor) modifications to CLT.

We begin abstractly, describing generic graded encoding procedures.

B.1 Graded Encoding Procedures

We recall the definition of a graded encoding system and describe some procedures for graded encodings, mostly following [12, 9]. The definition of "graded encoded system" and some of the procedures have been simplified or removed for our setting. We also add some procedures.

Definition B.1 (Graded Encoding System)**.** *A κ-graded encoding system for a ring R is a system of sets $\mathcal{E} = \{\mathcal{E}_i^{(m)} \in \{0,1\}^* : i \in \{0, 1, \ldots, \kappa, T\}, m \in R\}$ with the following properties:*

1. *For every i, the sets $\{\mathcal{E}_i^m : m \in R\}$ are disjoint.*
2. *There are binary operations $+$ and $-$ (on $\{0,1\}^*$) such that for every $m_1, m_2 \in R$, every i, and every $u_1 \in \mathcal{E}_i^{(m_1)}$ and $u_2 \in \mathcal{E}_i^{(m_2)}$, it holds that $u_1 + u_2 \in \mathcal{E}_i^{(m_1+m_2)}$ and $u_1 - u_2 \in \mathcal{E}_i^{(m_1-m_2)}$ where $m_1 + m_2$ and $m_1 - m_2$ are addition and subtraction in R.*
3. *There is a κ-ary operation $\times$ (on $\{0,1\}^*$) such that for every $m_1, \ldots, m_\kappa \in R$ and every $u_1 \in \mathcal{E}_1^{(m_1)}, \ldots, u_\kappa \in \mathcal{E}_\kappa^{(m_\kappa)}$, it holds that $u_1 \times \cdots \times u_\kappa \in \mathcal{E}_T^{(m_1 \cdots m_\kappa)}$ where $m_1 \cdots m_\kappa$ is multiplication in R.*
4. *There is a binary operation $\star$ (on $\{0,1\}^*$) such that for every $m_0, m_i \in R$ and every $u_0 \in \mathcal{E}_0^{(m_0)}$ and $u_i \in \mathcal{E}_i^{(m_i)}$, it holds that $u_0 \star u_i \in \mathcal{E}_i^{(m_0 \cdot m_i)}$ where $m_0 \cdot m_i$ is multiplication in R.*

CLT (and GGH) encodings do not quite meet the definition of graded encoding systems above, since the homomorphisms required in the definition eventually fail when the "noise" in the encodings becomes too large, analogously to how the homomorphisms may eventually fail in lattice-based homomorphic encryption. However, these noise issues are relatively straightforward (if tedious) to deal with.

The set $\mathcal{E}_i^{(m)}$ of encodings of m in $\mathcal{E}_i$ is analogous to the single element $g_i^m \in G_i$ in the algebraic group setting, which encodes m in group G_i. We sometimes call encodings in $\mathcal{E}_0$ "level-0 encodings". For symmetric graded encodings, we set $\mathcal{E}_1 = \cdots = \mathcal{E}_\kappa$. For asymmetric graded encodings, we keep these sets distinct.

Now, we define some procedures for graded encoding schemes.

Instance Generation $\mathsf{InstGen}(\lambda, \kappa, r)$ takes as input a security parameter λ, the multilinearity parameter κ, a ring dimension parameter r, and outputs $(\mathsf{params}, p_{zt})$, where params is a "description" of a κ-graded encoding system for a ring $R = R_1 \times \cdots \times R_r$, and p_{zt} is a zero-test parameter for $\mathcal{E}_T$. We assume R is chosen such that the density of zero divisors in each R_i is negligible. We let esk denote a master secret key associated to the graded encoding system (which is not revealed).

Remark 6. *Setting $r = 1$ corresponds to the prime order setting, while $r > 1$ corresponds to the composite-order setting.*

Ring Sampler $\mathsf{Samp}(\mathsf{params})$ is a randomized algorithm that outputs a level-0 encoding of a statistically uniform element $m \in R$, though the encoding itself need not be uniform.

Re-Randomization $\mathsf{ReRand}(\mathsf{params}, i, u)$ is a randomized algorithm that takes as input an encoding $u \in \mathcal{E}_i^{(m)}$ for some m, and outputs a new encoding $u' \in \mathcal{E}_i^{(m)}$. The distributional requirement is that, for any encodings $u_1, u_2 \in \mathcal{E}_i^{(m)}$, the distributions of $\mathsf{ReRand}(\mathsf{params}, i, u_1)$ and $\mathsf{ReRand}(\mathsf{params}, i, u_2)$ are statistically indistinguishable.

Remark 7. *Again, due to the noisiness of GGH and CLT encodings, iterative applications of* ReRand *will eventually cause the noise to exceed a threshold, after which encodings become garbage. In that context, it will be understood that* ReRand *must be be correct only "up to noise".*

Addition, Subtraction, and Multiplication These are the $+$, $-$, and $\times$ procedures of the graded encoding system.

Zero-test The procedure $\mathsf{IsZero}(\mathsf{params}, p_{zt}, u)$ takes an encoding u and outputs "true" if $u \in \mathcal{E}_T^{(0)}$ and "false" otherwise.

Now, we define a couple of procedures not provided in [12, 9]. These procedures are directed to graded encoding systems with an encoding space R that can be decomposed nontrivially as a direct product $R_1 \times \cdots \times R_r$.

Subring Generation $\mathsf{SubRGen}(\mathsf{esk}, i, S)$ is a private randomized algorithm that takes as input the graded encoding secret key and a subset $S \subset [r]$. It generates $m \in R$ such that m is random in R_j for $j \in S$ but m is 0 in R_j for $j \in [r] \setminus S$. It then outputs a random encoding in $\mathcal{E}_i^{(m)}$.

Subring Sampling $\mathsf{SubRSamp}(\mathsf{params}, i, S, \mathcal{G}_{S,i}\}$ takes as input the parameters, $i \in [\kappa]$, $S \subset [r]$, and a "generating set" of encodings $\mathcal{G}_{S,i} \subset \mathcal{E}_{S,i}$, where $\mathcal{E}_{S,i} \subset \mathcal{E}_i$ denotes the subset of encodings that encode some m such that m is 0 in R_j for $j \in [r] \setminus S$.

Remark 8. *The form of the "generating set" will depend on the type of encodings. For example, for encodings over an asymmetric algebraic group system of order $N = p_1 \cdots p_r$, $\mathcal{G}_{S,i}$ can be represented by a single element $g_i^{\prod_{j \notin S} p_j}$, which generates the group of order $\prod_{j \in S} p_j$. For "noisy" encodings, where multiplying/exponentiating by big numbers would blow up the noise, generating sets instead consist of many random encodings, and we sample by taking a random subset sum of the encodings, and applying the leftover hash lemma to argue that the result is well-distributed.*

B.2 The Composite-Order Symmetric-Map Scheme Revisited

We describe the translation of our composite-order symmetric map scheme to CLT encodings in full detail, first describing the scheme in terms of the generic graded encoding procedures, and then describing how to implement these procedures in CLT, as well as certain security issues that arise. Translating the prime-order scheme is strictly easier (we nonetheless discuss the translation of prime-order maps into CLT later on), and the asymmetric case does not raise any additional interesting issues.

Recall that our witness encryption scheme, and its security proof, are very modular. To update our WE construction for graded encodings, we only need to update the our implementation of the tribes scheme for PWE. To update the security proof, we only need to translate our assumptions to the graded encoding setting, and prove that the updated assumptions still imply intra-column and inter-column security for the tribes scheme. We begin with the construction.

Create(λ, M)**:** The creation algorithm takes in a security parameter λ and an $n \times \ell \times 2$ tribes matrix M (entries in $\{U, B\}$). It then calls $(\mathsf{params}, p_{zt}) \leftarrow \mathsf{InstGen}(\lambda, n, n+\ell)$ to produce a n-graded encoding system for ring $R = R_1 \times \cdots \times R_{n+\ell}$. The algorithm produces $2n$ encodings $\{u_{i,\beta}\}$ from $\mathcal{E}_1$, each indexed by a row $i \in [n]$ and a slot $\beta \in \{0,1\}$. We let $u_{i,\beta} \in \mathcal{E}_1$ be sampled as follows. Let $S_{i,\beta} \subset [n+\ell]$ consist of indices $i' \in [n] \setminus \{i\}$ as well as all $n+j$ such that $M_{i,j,\beta} = B$. Set

$$u_{i,\beta} \leftarrow \mathsf{SubRGen}(\mathsf{esk}, 1, S_{i,\beta}).$$

The tribes scheme T consists of these $2n$ elements $\{u_{i,\beta}\}$, as well as the parameters params, p_{zt} of the graded encoding system.

Eval(T, x)**:** The evaluation algorithm takes in a tribes scheme T and a boolean vector $x = (x_1, \ldots, x_n) \in \{0,1\}^n$. It sets $u_T \leftarrow u_{1,x_1} \times \cdots \times u_{n,x_n}$ and runs $\mathsf{IsZero}(\mathsf{params}, p_{zt}, u_T)$ to distinguish whether $u_T \in \mathcal{E}_T^{(0)}$ or not. If so, it outputs 0. Otherwise, it outputs 1.

Correctness Clearly $u_T \in \mathcal{E}_T$, up to noise issues. Let us consider now what u_T encodes, so that we can establish that $Eval(Create(\lambda, M), x) = f_M(x)$. By CRT, we can separately consider what is encoded in each subring R_i. We can see that $\cap_{i\in[n]} S_{i,x_i}$ includes no index in $[n]$, and therefore the encoded term is 0 in R_i for all $i \in [n]$.

For $i \in [n+1, \ldots, n+\ell]$, we consider two cases. Suppose $\exists$ a column j such that $M_{i,j,x_i} = B$ for all $i \in [n]$. This is equivalent to supposing that $f_M(x) = 1$. In this case, every S_{i,x_i} includes $n+j$, and thus every u_{i,x_i} encodes a random (likely nonzero) residue in R_{n+j}, and thus u_T likely encodes a nonzero residue in R_{n+j} (due to the negligible density of zero divisors), which implies $u_T \notin \mathcal{E}_T^{(0)}$, resulting in an output that matches f_M. In the other case, no such column j exists. This means that for every column j, there is some S_{i,x_i} which is missing $n+j$, which implies this u_{i,x_i} and hence u_T encodes 0 in R_{n+j}, which implies that u_T encodes 0 in all R_i, and the output again matches f_M.

We now state an assumption in terms of graded encodings, and show that it implies the inter-column security for the tribes scheme above.

Computational Assumption 1_S for Graded Encodings The challenger runs $(\mathsf{params}, p_{zt}) \leftarrow \mathsf{InstGen}(\lambda, n, n+\ell)$ to obtain a graded encoding system associated to $R = R_1 \times \cdots \times R_{n+\ell}$. For S equal to $\{1\}, \ldots, \{n+\ell-1\}$ and $\{n+\ell\} \cup [n] \setminus \{1\}, \ldots, \{n+\ell\} \cup [n] \setminus \{n\}$, it populates a set of encodings $\mathcal{G}_S$ by running the procedure $\mathsf{SubRGen}(\mathsf{esk}, 1, S)$ enough times to obtain a "generating set" (that could later be used in $\mathsf{SubRSamp}$). The challenge term u^* is an encoding that is the output of $\mathsf{SubRGen}(\mathsf{esk}, 1, S^*)$, where S^* equals either $\{n+\ell\} \cup [n]$ or $[n]$. The task is, given the $\mathcal{G}_S$'s and u^*, to distinguish which distribution u^* comes from.

The assumption above is directly analogous to Assumption 1_S for algebraic multilinear groups. One difference is that, in the context of groups, the generating set is a single group element, whereas we cannot assume that this works in general for graded encoding systems.

Lemma B.2. *Assumption 1_S for graded encodings implies inter-column security for the tribes scheme.*

Proof. Without loss of generality, we consider the game with $i^* = n$, $\beta^* = 0$, $j^* = 2$, $k^* = 1$ and any $n \times \ell \times 2$ tribes matrix M satisfying the game requirements for these indices. We relabel the indices in $[n+\ell]$ in a way suitable for this game. In particular, after the relabeling, the assumption instance includes $\mathcal{G}_S$ for S equal to $\{1\},\ldots,\{n\},\{n+2\},\ldots\{n+\ell\}$ (all but $\{n+1\}$) and $\{n+1,n+2\}\cup[n-1]\setminus\{1\},\ldots,\{n+1,n+2\}\cup[n-1]\setminus\{n-1\}$,$\{n+1\}\cup[n-1]$, as well as the challenge term u^*, which is sampled from the subring associated to S^*, where S^* is either $\{n+1,n+2\}\cup[n-1]$ or $\{n+2\}\cup[n-1]$. Our task is to guess the distribution that u^* was sampled from.

We now need to generate $2n$ elements to give to $\mathcal{A}$ to represent the tribes scheme. Recall that in the tribes construction, we did the following. For $i\in[n]$, $\beta\in\{0,1\}$, we set $S_{i,\beta}\subset[n+\ell]$ to consist of all $i'\in[n]\setminus\{i\}$, as well as all $n+j$ such that $M_{i,j,\beta}=B$. Then, we set

$$u_{i,\beta}\leftarrow\mathsf{SubRGen}(\mathsf{esk},1,S_{i,\beta}).$$

We need to simulate this distribution without esk, using the assumption instance.

For every (i,β) with $i\neq i^*$, we observe that $S_{i,\beta}\setminus\{n+1\}$ is the union of sets that have generators sets in the assumption – namely, it can be constructed as a union of some of the sets $\{1\},\ldots,\{n\},\{n+2\},\ldots\{n+\ell\}$. Since $i\neq i^*$, it holds in the inter-column game that $S_{i,\beta}$ contains $\{n+1\}$ (corresponding to k^*) only if it contains $\{n+2\}$ (corresponding to j^*). Also, $S_{i,\beta}$ contains $\{i'\}$ for all $i'\in[n]\setminus\{i\}$. Therefore, in this case, $S_{i,\beta}$ either equals $S_{i,\beta}\setminus\{n+1\}$, or equals the union of that set and the set $\{n+1,n+2\}\cup[n-1]\setminus\{i\}$, which has a generator in the assumption. Therefore, $S_{i,\beta}$ can be expressed as the union of sets with generators in the assumption when $i\neq i^*$.

But since $S_{i,\beta}$ is expressible in this way, we can use the generators in the assumption to generate a well-distributed $u_{i,\beta}$. Let $\mathcal{S}_{i,\beta}$ be the collection of sets from the assumption such that $S_{i,\beta}=\cup_{S\in\mathcal{S}_{i,\beta}}S$. For each $S\in\mathcal{S}_{i,\beta}$, we run $\mathsf{SubRSamp}(\mathsf{params},1,S,\mathcal{G}_S)$ to generate a random element $u_S\in\mathcal{E}_1$. Then, we set $u'_{i,\beta}\leftarrow\sum_{S\in\mathcal{S}_{i,\beta}}u_S$, which encodes value m that is random except that it is 0 in R_j with $j\notin S_{i,\beta}$, but where the encoding itself its not necessarily well-distributed. Finally, we run $u_{i,\beta}\leftarrow\mathsf{ReRand}(\mathsf{params},1,u'_{i,\beta})$ to canonicalize the distribution of the encoding.

For (i,β) with $i=i^*$ and $\beta=1-\beta^*$, constructing $S_{i,\beta}$ as a union of sets from the instance is easier. Depending on whether or not $\{n+1\}$ is included, we deal with the part from $[n+1]$ by using either $\{n+1\}\cup[n-1]$ or $\{1\}\cup\cdots\cup\{n-1\}$. Then we union in the appropriate sets from $\{n+2\},\ldots,\{n+\ell\}$. By taking sums and running $\mathsf{SubRSamp}$ and ReRand as above, we can therefore generate well-distributed $u_{i,\beta}$.

Finally, consider the case $(i,\beta)=(i^*,\beta^*)$. Set $T_{i,\beta}$ to be the union of S^* and $S_{i,\beta}\cap[n+3,n+\ell]$. We can construct $T_{i,\beta}$ from S^* and other sets from the assumption instance. Note that if $S^*=\{n+1,n+2\}\cup[n-1]$, then $T_{i,\beta}$ equals $S_{i,\beta}$ for the case $M_{i^*,k^*,\beta^*}=M_{i^*,j^*,\beta^*}=B$. On other hand, if $S^*=\{n+2\}\cup[n-1]$, then $T_{i,\beta}$ equals $S_{i,\beta}$ for the case $M_{i^*,k^*,\beta^*}=U$ and $M_{i^*,j^*,\beta^*}=B$. (These are the only two cases in the inter-column security game.) Overall, by taking unions of sets from the assumption instance, we can construct sets that correspond to a correct tribes matrix for one of the two cases of the inter-column security game (depending on S^*). Again, we take sums and use $\mathsf{SubRSamp}$ and ReRand as above to obtain well-distributed encodings that correspond to these two cases. If an attacker could distinguish these cases with non-negligible advantage – i.e., distinguish whether the $T_{i,\beta}$ corresponds to M_{i^*,k^*,β^*} equals U or B – then we could leverage the attacker to break Assumption 1 with non-negligible advantage. $\square$

We last show that intra-column security for this tribes scheme is implied by our second computational assumption.

Computational Assumption 2_S for Graded Encodings The challenger runs $(\mathsf{params}, p_{zt}) \leftarrow \mathsf{InstGen}(\lambda, n, n+\ell)$ to obtain a graded encoding system associated to $R = R_1 \times \cdots R_{n+\ell}$. For S equal to $\{1\}, \ldots, \{n+\ell-1\}$, it populates a set of encodings $\mathcal{G}_S$ by running the procedure $\mathsf{SubRGen}(\mathsf{esk}, 1, S)$ enough times to obtain a "generating set" (that could later be used in $\mathsf{SubRSamp}$). The challenge term u^* is an encoding that is the output of $\mathsf{SubRGen}(\mathsf{esk}, 1, S^*)$, where S^* equals either $\{n+\ell, n\}$ or $\{n\}$. The task is, given the $\mathcal{G}_S$'s and u^*, to distinguish which distribution u^* comes from.

Lemma B.3. *Assumption 2_S for graded encodings implies intra-column security for the tribes scheme.*

Proof. Without loss of generality, we consider the game with $i^* = n$, $j^* = 1$, and any $n \times \ell \times 2$ tribes matrix M and alternate column C satisfying the game requirements for these indices. We will break the game into two stages: in the first stage, we will transition to the first column of M being all U entries. In the second stage, we will transition to this column being equal to C.

We describe the reduction to Assumption 2_S for the first stage only, as the second stage is analogous. Relabeling the indices, the assumption instance consists of $\mathcal{G}_S$ for S equals $\{1\}, \ldots, \{n\}, \{n+2\}, \ldots \{n+\ell\}$ (all but $\{n+1\}$) and challenge term u^* associated to S^* that equals either $\{n, n+1\}$ or $\{n\}$. Our task is to guess the distribution u^* was sampled from.

We form the $2n$ group elements to give to an attacker $\mathcal{A}$ as follows. Recall that in the tribes construction, the element $u_{i,\beta}$ is associated to the set $S_{i,\beta} \subset [n+\ell]$, which consists of all $i' \in [n] \backslash \{i\}$, as well as all $n+j$ such that $M_{i,j,\beta} = B$.

For $i < i^* = n$, observe that $S_{i,\beta}$ includes n. Therefore, $S_{i,\beta} \cap \{n, n+1\}$ equals either $\{n, n+1\}$ or $\{n\}$, both of which are sets present in the assumption instance. $S_{i,\beta}$ can therefore be constructed as a union of sets from the assumption. However, instead of using the set $\{n, n+1\}$, we use the challenge set S^*, which is either $\{n, n+1\}$ or $\{n\}$. Consequently, depending on the value of S^*, we generate as a union either the original $S_{i,\beta}$'s, or modified $S_{i,\beta}$'s where all the $n+1$'s have been removed.

For $i = i^* = n$, recall $M_{i^*,j^*,0} = M_{i^*,j^*,1} = U$ in both of the cases of the intra-column security game and therefore neither $S_{i^*,0}$ nor $S_{i^*,1}$ contains $\{n+1\}$. It is therefore straightforward to generate $S_{i^*,\beta}$ as a union of sets from the assumption.

Overall, by taking unions of sets from the assumption instance, we can construct sets that correspond to a correct tribes matrix for one of the two cases of the inter-column security game (depending on S^*). As described in the previous proof, we use $\mathsf{SubRSamp}$ and ReRand as above to obtain well-distributed encodings that correspond to these cases. If an attacker could distinguish these cases with non-negligible advantage, then we could leverage the attacker to break Assumption 2 with non-negligible advantage. □

B.3 Overview of CLT Encodings

CLT encodings have a couple of properties that make them more attractive in our setting than the original multilinear maps of Garg, Gentry and Halevi (GGH) [12]. First, as Garg et al. noted in their paper, GGH encodings are subject to a "weak discrete log" attack. This attack can be avoided by working with "multilinear jigsaw puzzle" pieces [13] consisting of matrices of encodings (rather than individual encodings). However, we find it simpler to work with CLT encodings, which (as far as we know) do not seem to be vulnerable to this attack in the first place. Second, GGH encodings are built for a prime-order encoding space. While it is probably relatively straightforward to modify GGH encodings to support a composite-order encoding space, we prefer to work with CLT encodings, which inherently support a composite integer encoding space already. Unfortunately, the translation from composite order groups to CLT's composite order encoding space is not quite as direct as one would like – the most "direct" translation is subject to attacks, as we discuss in section B.6 – but it is still relatively straightforward.

A κ-linear symmetric CLT encoding system uses a "small" inner modulus $N = p_1 \cdots p_s$ that is the product of $s = s(\lambda, \kappa)$ "small" primes, and a "large" outer modulus $Q = P_1 \cdots P_s$ that is the product of s "large" primes. It uses a random $z \leftarrow \mathbb{Z}_Q^*$. An encoding $c \in \mathcal{E}_1^{(m)}$ is an element of $\mathbb{Z}_Q$ such that

$$c \equiv \frac{[m]_{p_i} + x_i \cdot p_i}{z} \bmod P_i \text{ for } \ i \in [s], \tag{4}$$

where $[m]_{p_i}$ is m reduced modulo p_i into a small range such as $(-p_i/2, p_i/2)$, and the x_i's are random small integers. An encoding in $\mathcal{E}_T$ has a similar form, but with z^κ in the denominator.

For random small integers $h_1, \ldots, h_s$, the system includes a zero-testing parameter p_{zt} for level κ of the form:

$$p_{zt} = \sum_{i=1}^{s} h_i \cdot (z^\kappa \cdot p_i^{-1} \bmod P_i) \cdot \prod_{j \neq i} P_j \bmod Q. \tag{5}$$

If c is a level-κ encoding of $0 \in \mathbb{Z}_N$ – i.e., each $[m]_{p_i} = 0$ – we have:

$$\begin{aligned} c \cdot p_{zt} &= \sum_{i=1}^{s} (x_i \cdot p_i / z^\kappa) \cdot h_i \cdot (z^\kappa \cdot p_i^{-1} \bmod P_i) \cdot \prod_{j \neq i} P_j \bmod Q \\ &= \sum_{i=1}^{s} x_i \cdot h_i \cdot \prod_{j \neq i} P_j \bmod Q \end{aligned}$$

which is a number substantially smaller than Q assuming the x_i's and h_i's satisfy certain smallness constraints – in particular, that each $x_i \cdot h_i \ll P_i$. On the other hand, if c encodes something other than 0, $c \cdot p_{zt}$ likely will not be a small number, due to uncanceled p_i^{-1}'s in the expression above. Thus, p_{zt} enables zero-testing. (Actually, CLT uses a polynomial number of such zero-testing parameters, and they prove that c encodes 0 if it passes the tests with respect to all of them, and does not encode 0 otherwise.)

By CRT, we can add and multiply CLT encodings while preserving their form (per Equation 4) as long as the numerators in Equation 4 do not grow too large – i.e., they do not "wrap" modulo P_i for any i. The P_i's must be chosen large enough to ensure that such wrapping never occurs for the functions we will compute over the encodings. These additions and multiplications induce additions and multiplications on the underlying "messages" that are encoded, much like homomorphic encryption.

Like GGH, CLT generalizes easily to allow asymmetric graded encodings. The simplest way to build asymmetric multilinear CLT encodings is simply to generate a random $z_i \leftarrow \mathbb{Z}_Q^*$ corresponding to source group G_i, rather than a single z. For $i \in [\kappa]$, An encoding in $\mathcal{E}_i^{(m)}$ now has the form

$$c \equiv \frac{[m]_{p_i} + x_i \cdot p_i}{z_i} \bmod P_i \text{ for } \ i \in [s]. \tag{6}$$

The form of the zero-test parameter changes to:

$$p_{zt} = \sum_{i=1}^{s} h_i \cdot (Z \cdot p_i^{-1} \bmod P_i) \cdot \prod_{j \neq i} P_j \bmod Q. \tag{7}$$

where $Z = \prod_{i \in [\kappa]} z_i$. Similar to the symmetric case, multiplying p_{zt} with an encoding in $\mathcal{E}_T^{(0)}$ (which has Z in the denominator) results in a mod-Q number that is small relative to Q.

Intuitively, the asymmetric form of the encodings limits how a user can meaningfully multiply together encodings, so that each monomial it computes corresponds to multiplying together exactly one encoding from each source group, so that it obtains an encoding with Z in the denominator. For example, the multilinear map cannot be used directly to solve decision Diffie-Hellman over elements of G_1, since this would involve multiplying together encodings from G_1, which would induce an uncancellable z_1^2 in the denominator. In the asymmetric setting, it is even plausible that it is hard for an attacker to distinguish whether an encoding from a source group encodes 0, even though zero-testing for $\mathcal{E}_T$ is efficient, as long as we do not give the attacker elements from other source groups that would allow it to distinguish trivially. However, in the sequel, we focus on symmetric CLT encodings for simplicity.

Translating our schemes from algebraic multilinear groups to CLT encodings requires some care. In contrast to encodings over groups, CLT encodings are probabilistic and noisy, and come from a distribution. We have to define these distributions, and show that they are correct in our scheme and in the hybrids of our security proof.

Another issue is representing subrings of $\mathbb{Z}_N$ with CLT encodings. There are two big issues here: 1) how to give a useful noise-resilient description of subrings, and 2) whether it is secure in the CLT setting to give descriptions of subrings. The natural way to represent a subring in the no-noise setting is to give a generator of that subring, which typically can be a single element. Then, to generate a random element in that subring, one simply multiplies the generator by a random number. This strategy does not work in the noisy setting, since multiplying an encoding by a big number also blows up the noise. Instead, our approach is to represent a subring by a large "generating set" – a bunch of encodings that encode elements of the subring – and to generate random elements in the subring by taking random subset sums over the generating set and using the leftover hash lemma. Thus, our CLT-based assumptions end up looking somewhat more complicated than the analogous assumptions in the group setting, since each subring generator is expanded into a larger generating set. Regarding security, we have to ask: Is it safe, for example, to give an encoding of some m that is in the index-p_i subring of $\mathbb{Z}_N$? Unfortunately, it is not! As we discuss in more detail in Section B.6, unless one is extremely careful with the parameter settings, one can use such a CLT encoding to efficiently recover p_i! The original CLT proposal [9] was careful to never give out encodings in which any divisor of N was "isolated" in this way; for the encodings in the parameters, the encoded values are 0 modulo all of the p_i's or none of them. To translate our composite-order constructions, we need to use subrings, and therefore we cannot use CLT's safe "all-or-nothing" approach. However, we still avoid letting any p_i be "isolated" by giving it many – i.e., poly(λ) – "buddies": any encoding that an attacker sees is 0 modulo p_i and all of its prime buddies $\{p_j\}$, or is (whp) nonzero for all of them. As we discuss in Section B.6, this approach seems resilient to attacks.

Alternatively, one can avoid using these subrings by translating our prime-order construction to the CLT setting. For this translation, only "conventional" CLT encodings are needed.

Below, we flesh out the overview above. We provide more details on CLT encodings, and on translating our schemes, assumptions and proofs from the general graded encoding setting to CLT.

B.4 Graded Encoding Procedures for CLT

Here, we describe how the graded encoding procedures from Section B.1 are instantiated in CLT.

Instance Generation $\mathsf{InstGen}(\lambda, \kappa, r)$ takes as input the security parameter λ, the multilinearity parameter κ, and a ring dimension parameter r. It generates, for each $i \in [r]$ and $\theta \in [\Theta = \Theta(\lambda, \kappa)]$, a $\rho = \rho(\lambda, \kappa)$-bit small prime $p_{i,\theta}$ and a $\eta = \eta(\lambda, \kappa)$-bit big prime $P_{i,\theta}$, and sets $N = \prod_{i,\theta} p_{i,\theta}$ and $Q = \prod_{i,\theta} P_{i,\theta}$. For $i \in [r]$, we let R_i denote the ring $\mathbb{Z}_{\prod_\theta p_{i,\theta}}$. The parameter Θ specifies how many primes are associated to each R_i, and it needs to be set large (but polynomial) for security reasons. We let $R = R_1 \times \cdots \times R_r = \mathbb{Z}_N$. To eventually obtain correctness and security against known attacks, one can take $\rho = O(\lambda)$, $\eta = O(\lambda \cdot (\lambda + \kappa))$, and $\Theta = (\rho \cdot \eta)^{1+\epsilon}$, $\epsilon > 0$. Depending on whether the encoding is symmetric or asymmetric, it generates a single value $z \in \mathbb{Z}_Q$ or κ values $z_1, \ldots, z_\kappa \in \mathbb{Z}_Q$. (Below, we will focus on the symmetric case for simplicity unless stated otherwise.)

For parameter $t = t(\lambda, \kappa)$, $\mathsf{InstGen}$ generates $t = t(\lambda, \kappa)$ random numbers $m_j \in \mathbb{Z}_N$, and generates level-0 encodings of them:

$$c_j \equiv [m_j]_{p_{i,\theta}} + x_{ji\theta} \cdot p_{i,\theta} \bmod P_{i,\theta}.$$

where the $x_{ji\theta}$'s are random numbers in $(-2^\rho, 2^\rho)$. It also generates $t + s$ level-1 encodings of 0:

$$c'_j \equiv \frac{x'_{ji\theta} \cdot p_{i,\theta}}{z} \bmod P_{i,\theta}.$$

The main requirement on t is that it is large enough to allow application of the leftover hash lemma when we take a subset sum of the c_j's or c'_j's.

It generates a vector $\mathbf{pzt}$ of zero-test parameters, where each p_{zt} in the vector equals $\sum_{i,\theta} h_{i,\theta} \cdot (z^\kappa \cdot p_{i,\theta}^{-1} \bmod P_{i,\theta}) \cdot \prod_{(i',\theta') \neq (i,\theta)} P_{i',\theta'} \bmod Q$, where the random $h_{i,\theta}$'s are chosen to have (for example) $3\eta/4$ bits (so that $h_{i,\theta}$ is much smaller than $P_{i,\theta}$, but $h_{i,\theta}^2$ is much bigger).

It outputs $(\mathsf{params}, \mathbf{pzt})$ where params includes the basic parameters $\lambda, \kappa, \rho, \eta, s, t, Q$. Certain values, such as z and the prime divisors of N and Q, remain secret as part of esk.

Remark 9. *The setting of prime-order multilinear maps corresponds to setting $r = 1$. Regardless of whether we are translating a prime-order or composite-order scheme, the CLT encoding space is composite (having a factor of Θ more primes that the original encoding space).*

Ring Sampler $\mathsf{Samp}(\mathsf{params})$ generates a random binary vector $b_1 \cdots b_t \in \{0,1\}^t$ and outputs $u \leftarrow \sum_{j=1}^t b_j \cdot c_j \bmod Q$. The statistical uniformity of the encoded value follows by application of the leftover hash lemma to $R = \mathbb{Z}_N$.

Re-Randomization $\mathsf{ReRand}(\mathsf{params}, u)$ works by adding a random encoding in $\mathcal{E}_1^{(0)}$ to $u \in \mathcal{E}_1^{(m)}$ to generate a new encoding $u' \in \mathcal{E}_1^{(m)}$. The random encoding of 0 is generated according to a large-enough distribution to "drown" the distribution of u. In particular, one sets $u' \leftarrow u + \sum_{j=1}^{t+s} b_j \cdot c'_j \bmod Q$, where $b_1, \ldots, b_t$ are sampled uniformly from $\{0,1\}$ and $b_{t+1}, \ldots, b_{t+s}$ are sampled uniformly from $[-2^\delta, 2^\delta]$ for suitable $\delta = \delta(\lambda, \kappa)$, which can be $\tilde{O}(\lambda)$.

Remark 10. *Though we omit details, intuitively the subset sum using the first t encodings of 0 randomizes things "locally" – in particular, it is uniform over the certain cosets of a lattice defined by the last s encodings of 0. The random linear combination over the last s encodings of 0 then randomizes things "globally" by adding a random lattice point drawn uniformly from a huge parallelepiped. CLT [9] proves a "leftover hash lemma over lattices" to establish that this process statically induces the desired canonical distribution.*

Remark 11. *In* ReRand*, if u itself was the (perhaps indirect) output of such a randomization procedure, then we would need to make the b_i's even larger to drown the distribution of u.*

Addition, Subtraction, and Multiplication These operations are performed in the natural way via addition, subtraction, or multiplication modulo Q.

Zero-test The procedure $\mathsf{IsZero}(\mathsf{params}, \mathbf{pzt}, u)$ takes an encoding u and applies the zero-test parameter to distinguish whether $u \in \mathcal{E}_T^{(0)}$. This procedures works simply by multiplying by each individual p_{zt} and seeing whether the result is small, as described above.

The following procedures are not included in CLT, except for the case of $r = 1$.

Subring Generation $\mathsf{SubRGen}(\mathsf{esk}, S)$ is a private randomized algorithm that takes as input a subset $S \subset [r]$ and the graded encoding secret key, which includes z and the factorizations of N and Q. It generates $m \in R = \mathbb{Z}_N$ such that m is random in R_j for $j \in S$ but m is 0 in R_j for $j \in [r] \setminus S$. It sets $u_0 \in \mathbb{Z}_Q$ to be $[m]_{p_{i,\theta}}/z \bmod P_{i,\theta}$. It then sets $u_1 \leftarrow \mathsf{ReRand}(\mathsf{params}, u_0)$.

Observe that when $r = 1$, $\mathsf{SubRGen}(\mathsf{esk}, \{1\})$ outputs an encoding of a random element, while $\mathsf{SubRGen}(\mathsf{esk}, \emptyset)$ outputs an encoding of 0. Both of these functionalities can be performed just using params (no secret information) in the original version of CLT encodings.

Subring Sampling $\mathsf{SubRSamp}(\mathsf{params}, S, \mathcal{G}_S\}$ takes as input the parameters, a subset $S \subset [r]$, and a purported set $\{v_i\}$ of encodings in $\mathcal{E}_S$, where $\mathcal{E}_S$ is the set of encodings that encode that some m such that m is 0 in R_j for $j \in [r] \setminus S$. It runs $\{w_i \leftarrow \mathsf{Samp}(\mathsf{params})\}$, and outputs $u_0 \leftarrow \sum w_i \star v_i$. It then sets $u_1 \leftarrow \mathsf{ReRand}(\mathsf{params}, u_0)$, which should be a statistically random encoding from $\mathcal{E}_S$.

B.5 Comments on Noise Distributions

The bound on the size of the numerator in CLT encodings grows exponentially with κ, but this can be accommodated by setting the parameters large enough (but still polynomial).

When adapting the security proofs for our constructions to CLT, we need to ensure that the distributions (in particular, the noise distributions) generated in the security proof are identical to those generated by the encryption algorithm. However, this is easy to ensure. It is clear that the distributions of the encoded terms (in R) are statistically indistinguishable. To ensure that the encodings themselves are statistically indistinguishable, we can modify the encryption procedure to apply the same ReRand algorithm that is used in the proof, with parameters sufficient to "drown out" the distribution of the initial encoding output by encryption.

B.6 Discussion of the Assumptions

As mentioned above, the most "natural" way to translate a scheme over groups of composite order $N = p_1 \cdots p_r$ to CLT encodings would be to use $\mathbb{Z}_N$ directly as the CLT encoding space. Unfortunately, this approach fails for security reasons (or at least the security seems much more tenuous than for conventional CLT encodings).

Suppose we use the approach above, and we obtain an encoding c in $\mathcal{E}_T$ of some $m \in \mathbb{Z}_N$ such that $[m]_{p_j} \neq 0$ but $[m]_{p_i} = 0$ for all $i \neq j$. (We stress that the original CLT proposal [9] does not give out encodings of this form, and thus is not subject to this attack.) That is,

$$c = \frac{[m]_{p_j} + x_j \cdot p_j}{z^\kappa} \bmod P_j, \qquad c = \frac{x_i \cdot p_i}{z^\kappa} \bmod P_i \text{ for } i \neq j$$

for fairly small x_j and $\{x_i\}$. Set $a \leftarrow c \cdot p_{zt} \bmod Q$. We have that

$$a = ([m]_{p_j} \cdot h_j \cdot p_j^{-1} \mod P_j) \cdot \prod_{k \neq j} P_k + \sum_{i \in [s]} x_i \cdot h_i \cdot \prod_{k \neq i} P_k \bmod Q.$$

We do not have p_j a priori, but we do know that

$$b := a \cdot p_j = [m]_{p_j} \cdot h_j \cdot \prod_{k \neq j} P_k + \sum_{i \in [s]} p_j \cdot x_i \cdot h_i \cdot \prod_{k \neq i} P_k \bmod Q.$$

Furthermore, if the parameters are such that the values $[m]_{p_j} \cdot h_j$ and $\{p_j \cdot x_i \cdot h_i\}$ are all small in relation to the P_i's, then the value b is small in relation to Q. (Recall that the values $\{x_i \cdot h_i\}$ should be small to allow correct zero testing.) If this value is small enough, we can recover p_j via lattice reduction.

Specifically, let B be an approximation of the value b/p_j, based on the distributions of variables. (Note that B is dominated by the summation $\sum_{i \in [s]} x_i \cdot h_i \cdot \prod_{k \neq i} P_k \bmod Q$, and thus is comparable in size to the small mod-Q value that one would obtain by zero-testing an encoding that actually encodes 0.) Consider the two-dimensional lattice L generated by the vectors (B, a) and $(0, Q)$. This lattice contains the vector $\vec{v} := (B \cdot p_j, b)$, of length approximately $B \cdot p_j \cdot \sqrt{2}$. (Let us assume its length is upper-bounded by $2Bp_j$.) On the other hand, the determinant of the lattice is $B \cdot Q$, implying that all vectors in L that are not parallel to $\vec{v}$ must have length at least $B \cdot Q/(2Bp_j) = Q/2p_j$. If $B < Q/4p_j^2$, then $2Bp_j < Q/2p_j$, and $\vec{v}$ is the unique shortest vector in L, which can be recovered easily via lattice reduction. Recovery of $\vec{v}$ means recovery of p_j, a devastating attack on the encodings. While in principle the parameters could conceivably be set carefully to ensure that B is comfortably larger than $Q/4p_j^2$ (to avoid this attack) and comfortably smaller than Q (to allow zero-testing), we note that nothing approaching this amount of care was needed when setting the parameters of the original CLT scheme.

The attack can be extended, to some extent, to encodings of $m \in \mathbb{Z}_N$ such that m is nonzero modulo more than one prime divisor of N. For example, suppose that we have an encoding of m where m has nonzero residues modulo just p_j and p_k. Then, one can define a as above, b as $[a \cdot p_j \cdot p_k]_Q$, B as an approximation of $b/p_j p_k$, and reduce the lattice formed by (B, a) and $(0, Q)$. However, the target vector $(B \cdot p_j \cdot p_k, b)$ in this lattice will not be as short (it will be about p_k times longer), and thus B needs to be correspondingly smaller for lattice reduction to be effective. In general, it seems, the greater the nonzero support of m, the harder it is to use an encoding of m to recover factors of N.

Indeed, this is consistent with the security concept for CLT encodings. Consider a CLT encoding scheme like [9], but where N and Q are each divisible by only two primes. Such an encoding scheme would be subject to essentially the same attack as above. CLT eliminates such attacks by using many primes, only revealing encodings that are zero with respect to all of the primes or none of them, and gluing together zero-testers for individual primes into an aggregate zero tester. To extend CLT encodings so that we can reveal encodings of elements that are in subrings, we use a similar concept: we associate many primes to each subring R_i, and apply CLT's "all-or-nothing" approach within each subring: encodings are zero with respect to all of the primes associated to R_i or none of them.

We mention again that these issues do not arise when we translate our prime order scheme to the CLT setting, since there we can use conventional CRT encodings.

C A Generic Group Proof of Our Assumptions for Asymmetric Multilinear Maps

Here we prove that the assumptions we used in Section 7 hold in the generic group model for asymmetric, multilinear groups. In this model, access to the group operations of $G_1, G_2, \ldots, G_n, G_T$, the multilinear map E, and the zero test in G_T are mediated by an oracle who outputs large random handles as references to group elements. These are hard to guess, so a user can only learn new handles by applying the legitimate operations to the handles he already knows.

In our case, if an attacker could break one of our assumptions in the generic group model, this would be mean that it produced a fixed linear combination of multilinear terms over the variables of our sampled matrix distributions that is identically 0 modulo p for one case of the challenge and not for the other.

We recall our assumptions:

Assumption 1_{AP}: Set $d = 4$. Generate random matrices $R_0, \ldots, R_n$ in $\mathbb{Z}_p^{d\times d}$. Sample diagonal matrices $C_1^1, D_1^1, C_1^2, D_1^2$ as follows. When all sub/super-scripts are matching, C's and D's will always denote two independent samples from the same distribution (these will be used to supply sufficient randomness to give out the group elements for both slots when we instantiate our tribes scheme). For C_1^1, D_1^1, the first entries are random, while the remaining entries are 0. For C_1^2, D_1^2, the second entries are random and the remaining entries are 0.

We additionally sample diagonal matrices $C_i^0, D_i^0, C_i^2, D_i^2, C_i^{1,2}, D_i^{1,2}$ for each $i \geq 2$ as follows. The last two entries are always randomly distributed. The first two entries are both 0 when the superscript is 0. When the superscript is 2, the second entry is random and the first entry remains 0. When the superscript is 1,2, both of these are random.

Two final matrices $T^2, T^{1,2}$ are also sampled, where the last two entries of each are 0 and the second entry of each is random, but the first entry is 0 in T^2 and is random in $T^{1,2}$.

An instance of the assumption includes a description of the multilinear group and:

$$g_i^{R_{i-1}C_i^0R_i^{-1}}, g_i^{R_{i-1}D_i^0R_i^{-1}}, g_i^{R_{i-1}C_i^2R_i^{-1}}, g_i^{R_{i-1}D_i^2R_i^{-1}} g_i^{R_{i-1}C_i^{1,2}R_i^{-1}}, g_i^{R_{i-1}D_i^{1,2}R_i^{-1}} \text{ for } i = 2, \ldots, n$$
$$g_1^{R_0C_1^0R_1^{-1}}, g_1^{R_0D_1^0R_1^{-1}}, g_1^{R_0C_1^1R_1^{-1}}, g_1^{R_0D_1^1R_1^{-1}}, g_1^{R_0C_1^2R_1^{-1}}, g_1^{R_0D_1^2R_1^{-1}}$$
$$\text{Either } g_1^{R_0T^2R_1^{-1}} \text{ or } g_1^{R_0T^{1,2}R_1^{-1}}$$

The task is to distinguish which distribution the final encoded term comes from.

Assumption 2_{AP}: Set $d = 5$. Generate random matrices $R_0, \ldots, R_n$ in $\mathbb{Z}_p^{d\times d}$. Sample diagonal matrices $C_1, D_1, C_2^U, D_2^U, C_2^B, D_2^B, \ldots, C_n^U, D_n^U, C_n^B, D_n^B$ in $\mathbb{Z}_p^{d\times d}$ as follows. The C, D matrices with the same subscripts and superscripts will also represent two independent samples from the same distribution (these will be used to supply sufficient randomness to give out the group elements for both slots when we instantiate our tribes scheme).

The first entries of C_1, D_1 will be set to 0. For the rest, the first entry of each C_i^U, D_i^U will be distributed as 0 and the first entry of each C_i^B, D_i^B will be distributed randomly in $\mathbb{Z}_p$. The remaining 4 entries are sampled as follows. For C_1, D_1, the 1st of these 4 is set randomly, while the remaining three are fixed to 0. For each C_i, D_i with $i > 1$, the first of these four entries will be fixed to 0, while the remaining 3 will be set randomly. (Note it is the same for all of C_i^U, D_i^U and C_i^B, D_i^B). Two more matrices T_2^U, T_2^B are also sampled, distributed the same as D_2^U, D_2^B (though sampled independently).

An instance of the assumption includes a description of the multilinear group and:

$$g_i^{R_{i-1}C_i^U R_i^{-1}}, g_i^{R_{i-1}D_i^U R_i^{-1}}, g_i^{R_{i-1}C_i^B R_i^{-1}}, g_i^{R_{i-1}D_i^B R_i^{-1}} \text{ for } i = 2, \ldots, n$$
$$g_1^{R_0 C_1 R_1^{-1}}, g_1^{R_0 D_1 R_1^{-1}}$$
$$\text{Either } g_2^{R_1 T_2^U R_2^{-1}} \text{ or } g_2^{R_1 T_2^B R_2^{-1}}$$

The task is to distinguish which distribution the final encoded term comes from.

We prove generic security for Assumption 2_{AP} first, as the argument for Assumption 1_{AP} will then proceed very similarly (which we then include for completeness).

Lemma C.1. *Assumption 2_{AP} holds in the generic multi-linear group model.*

Proof. We suppose not. If we consider the entries of the diagonal matrices and the R_i's sampled in Assumption 2_{AP} as variables, we must then have a function f over these variables that is a multilinear polynomial over the diagonal entries and the entries of R_i, R_i^{-1} that is 0 with noticeably different probability when the challenge matrix is T_2^U versus when it is T_2^B.

We let $\mathcal{D}_1$ denote the distribution of the matrices when the challenge is sampled as T_2^U, and let $\mathcal{D}_2$ denote the distribution when the challenge is sampled as T_2^B. We first argue that f must be identically zero on $\mathcal{D}_1$ and not on $\mathcal{D}_2$. To see this, observe that if we were to multiply each R_i^{-1} by $det(R_i)$ (which is nonzero with high probability), we would a polynomial of suitably bounded degree in the variables of R_i, so the Schwartz-Zippel lemma can be applied to conclude that any f is either identically 0 or is 0 only with negligibly small probability (as $1/p$ is negligible). Hence, we must have f identically 0 on one of $\mathcal{D}_1, \mathcal{D}_2$ and not on the other. It is easy to see it must be $\mathcal{D}_1$, as this is more restrictive.

For any specified set S of (diagonal) entries of the diagonal matrices, we can define a projection $\mathcal{D}_{1,S}$ to be the distribution induced by fixing all diagonal entries not in S to 0 and otherwise sampling from $\mathcal{D}_1$. We can define the projected distribution $\mathcal{D}_{2,S}$ analogously. Since all of the nonzero diagonal entries are sampled uniformly and independent from $\mathbb{Z}_p$, it must be the case that f evaluates to 0 on each projected distribution $\mathcal{D}_{1,S}$, for any set S of diagonal entries. To see this, note that f can be organized as a linear combination of terms each divisible by a unique product of n diagonal entries, one from each group, where the "coefficients" are expressions in terms of the R_i, R_i^{-1} variables. Another application of the Schwartz-Zippel lemma then implies that each such projection must be identically 0.

We consider sets S that contain only one diagonal entry in each group G_i. This means there will only be one potentially non-zero diagonal entry among all the matrices given out in each group G_i. We observe there must be some such S where f evaluates to something not identically 0 on the projected distribution $\mathcal{D}_{2,S}$. Otherwise, it would be identically 0 overall, as f can be considered as linear combination of these projections.

So we can express S as a sequence in $\{1,2,3,4,5\}^n$, where the i^{th} entry of the sequence indicated which diagonal entry in group i is left as randomly distributed. The sequence must start with 21, otherwise the projection of f onto $\mathcal{D}_{2,S}$ would be identically 0.

We will next argue that there must also exist a sequence in $\{1,2,3,4\}^n$ indicating another set S' such that the projection of S onto $\mathcal{D}_{2,S'}$ is nonzero. We derive S from S' as follows. Our new sequence will have 1's and 2's in the same position as before, but any subsequence of elements in $\{3,4,5\}$ is replaced by a subsequence of $\{3,4\}$ with the property that if two adjacent elements were the same before, they remain the same now, but if they were different, they remain different. For example, 3455443453 could be replaced by 3433443434.

We now want to show that f must be nonzero on $\mathcal{D}_{2,S'}$. We claim in fact that the distributions $\mathcal{D}_{2,S}$ and $\mathcal{D}_{2,S'}$ are identical. Consider a sample of the distribution $\mathcal{D}_{2,S}$, corresponding to matrices $R_0, R_1, \ldots, R_n$ and diagonal matrices $D_1, \ldots, D_n$, where each R_i is uniformly random and each D_i has only one nonzero (and random) entry, in the position indicated by S.

We can thus consider the distribution of 5×5 matrices $R_0 D_1 R_1^{-1}$, $R_1 D_2 R_2^{-1}$, …, $R_{n-1} D_n R_n^{-1}$. Consider the first time that a 5 appears in the sequence, say at D_{i+1}. Let's suppose it is preceded by a 3 and so we want to change this 5 to a 4 to get from S to S'. We define the matrix $P_{5,4}$ to be the 5×5 matrix corresponding to this permutation. We then define $R_i = R_i' P_{4,5}$, and so $R_i^{-1} = P_{4,5}(R_i')^{-1}$. We can similarly define $R_{i+1} = R_{i+1}' P_{4,5}$. We can then rewrite the distribution of $R_{i-1} D_i R_i^{-1}$ and $R_i D_{i+1} R_{i+1}^{-1}$ as:

$$R_{i-1} D_i (R_i')^{-1} = R_{i-1} D_i P_{4,5} R_i^{-1}, \;\; R_i' D_{i+1} (R')_{i+1}^{-1} = R_i P_{4,5} D_{i+1} P_{4,5} R_{i+1}^{-1}.$$

Now observe that $D_i P_{4,5} = D_i$, since the only nonzero entry in D_i is at the third diagonal position. Also observe that $P_{4,5} D_{i+1} P_{4,5}$ is distributed as a diagonal matrix with a single random entry now in position 4.

More generally, for each position i in the sequence, let P_i denote the permutation matrix that must be applied to get send the i^{th} element of S to the i^{th} element of S'. (This will be the identity matrix when no change is required.) For each $i \geq 1$, define R_i' by the following relation:

$$R_i = R_i' P_{i+1} P_i, \text{ or equivalently, } R_i' = R_i P_i P_{i+1}, \text{ when } P_i, P_{i+1} \text{ are different,}$$

$$\text{and } R_i = R_i' P_i, \text{ or equivalently, } R_i' = R_i P_i, \text{ when } P_i, P_{i+1} \text{ are the same.}$$

So when $P_{i+1} \neq P_i$ for example, $R_i^{-1} = P_i P_{i+1} (R_i')^{-1}$ and $(R_i')^{-1} = P_{i+1} P_i R_i^{-1}$. We can then write the distribution $\mathcal{D}_{2,S}$ as:

$$R_0 D_1 (R_1')^{-1}, \; R_1' D_2 (R_2')^{-1}, \; R_2' D_3 (R_3')^{-1}, \ldots, \; R_{n-1}' D_n (R_n')^{-1}.$$

For i's such that $P_{i-1} \neq P_i \neq P_{i+1}$ for example, we have terms like:

$$R_{i-1} P_{i-1} P_i D_i P_{i+1} P_i R_i^{-1}.$$

We claim that in the case that $P_{i+1} \neq P_i$, the permutation P_{i+1} cannot affect D_i. This is clear if P_{i+1} is a swap between two things that are disjoint between the support of D_i or is the identity.

Suppose this is not the case. This means position $i+1$ in the sequence is being changed, so if positions i and $i+1$ were previously equal, then position i has to be undergoing the same swap, violating our supposition that $P_{i+1} \neq P_i$.

Thus we can assume positions i and $i+1$ were originally distinct. One possibility is that position $i+1$ was originally a 5 and position i was not. Then P_{i+1} is a swap between a 5 and something unequal to the support of D_i, hence P_{i+1} does not change D_i. So we may further assume that position $i+1$ was not originally a 5. Thus, the only reason for it to be swapped is that P_i has swapped the support of D_i onto the support of D_{i+1}. Either P_{i+1} will swap with something other than the support of D_i (hence leaving D_i unaffected), or this a trade of supports, again contradicting $P_i \neq P_{i+1}$. In all cases, we have that $P_i D_i P_{i+1} P_i = P_i D_i P_i$.

Similarly, we want to argue that the present P_{i-1} terms can also be absorbed into the $P_i D_i P_i$ terms (in the sense that they leave them unchanged). Such a term will only be present when $P_i \neq P_{i-1}$ and P_{i-1} is not the identity. Thus P_{i-1} must be implementing a swap of two positions. If D_i and D_{i-1} previously had equal supports, they would be undergoing the same swap, so we may assume these entries of the original sequence are unequal. If P_{i-1} is being swapped onto the support of D_i, then P_i must swap the support of D_i to somewhere disjoint from the action of P_{i-1}. Otherwise, the support of D_{i-1} is being swapped somewhere outside the support of D_i, so P_i is the identity, and P_{i-1} again fails to affect D_i. We may thus conclude that the distribution $\mathcal{D}_{2,S}$ is identical to the distribution $\mathcal{D}_{2,S'}$. By an analogous argument, we also get the same distribution if we replace S' by a sequence S'' which is derived from S by sending all 1's to 3's and all 3's to 5's. (Note it is crucial throughout that we are always replacing something with $\{3, 4, 5\}$, entries which are always non-zero in the matrices, so these replacements will not zeroize the projection.) We then have that f must be nonzero on $\mathcal{D}_{2,S''}$, but this projection is insensitive to the challenge, hence contradicting that f is zero on $\mathcal{D}_{1,S''}$.

$\square$

Lemma C.2. *Assumption 1_{AP} holds in the generic multi-linear group model.*

Proof. We again let $\mathcal{D}_1$ and $\mathcal{D}_2$ denote the two distributions induced by the two cases of the challenge term. In $\mathcal{D}_1$, the first diagonal entry of the challenge matrix is 0, while in $\mathcal{D}_2$ it is nonzero. If these can be distinguished in the generic group model, there must be an algebraic relation f that is identically 0 on $\mathcal{D}_1$ but not on $\mathcal{D}_2$. Furthermore, there is a projection S onto a single diagonal entry in each subgroup G_i such that f is zero on $\mathcal{D}_{1,S}$ but nonzero on $\mathcal{D}_{2,S}$. This S can be expressed as a sequence in $\{1, 2, 3, 4\}^n$. This sequence must begin with 1 (since it differs according to the challenge).

Employing the same techniques as above, we can compress this to a new sequence S' where subsequences in $\{2, 3, 4\}$ have been replaced by subsequences in $\{3, 4\}$. Note that in all matrices in groups $G_2, \ldots, G_n$, the diagonal entries 3, 4 are always random, so replacing a 2 by one of these in a matrix where the second entry is random will not zeroize the projection. Thus we can obtain a sequence S' in $\{1, 3, 4\}$ such that f is also nonzero on $\mathcal{D}_{2,S'}$. But now we can obtain another such sequence S'' by replacing each 1 with a 2, and this will not zeroize the projection b/c the second entries of our diagonal matrices are always random whenever the first entries are. So then f is nonzero on $\mathcal{D}_{2,S''}$, but this is now equal to $\mathcal{D}_{1,S''}$, hence we have obtained a contradiction. $\square$