Electronic Colloquium on Computational Complexity, Report No. 92 (2010)

A Completeness Theorem for Pseudo-Linear Functions with Applications to UC Security

Charanjit S. Jutla
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Arnab Roy
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

We consider multivariate pseudo-linear functions over finite fields of characteristic two. A pseudo-linear polynomial is a sum of guarded linear-terms, where a guarded linear-term is a product of one or more linear-guards and a single linear term, and each linear-guard is again a linear term but raised to the power 2^m-1, where 2^m is the field size. Pseudo-linear functions over GF(2^m) are given by pseudo-linear polynomials defined over GF2.

Let $f_1, f_2, ..., f_k$ be k pseudo-linear functions in n variables, and let f be another pseudo-linear function in the n variables. We show that if f is a function of the given k functions, then it must be a pseudo-linear function of the given k functions. This generalizes the straightforward claim for just linear functions. We also prove a more general theorem where the k functions can in addition take further arguments, and prove that if f can be represented as an iterated composition of these k functions, then it can be represented as a probabilistic pseudo-linear iterated composition of these functions.

These theorems have implications for automatic proving of universally-composable security theorems for ideal and real functionalities composed of if-then-else programs and data objects from additive group of GF(2^m). It follows that deciding if a simulator exists for such restricted languages is in computational time independent of m.

ISSN 1433-8092

1 Introduction

Before we define pseudo-linear functions, we mention that pseudo-linear functions originate as functions computed by if-then-else or branching programs involving data objects from the additive group of fields of characteristic two. The conditionals are built from equality constraints of linear expressions, and closed under negation and conjunction.

So, consider a finite field $\mathbb{F}_q$, where $q = 2^m$. Then m-bit (bit-wise) exclusive-OR just corresponds to addition in this field. Further, an equality constraint of the form $l_1(\vec{x}) = l_2(\vec{x})$ can then be written as

$$1 + (l_1(\vec{x}) + l_2(\vec{x}))^{q-1}$$

which evaluates to 1 if $l_1(\vec{x}) = l_2(\vec{x})$, and evaluates to zero otherwise. Similarly, $l_1(\vec{x}) = 0$ *and* $l_2(\vec{x}) = 0$ can be written as

$$(1 + l_1(\vec{x})^{q-1}) \cdot (1 + l_2(\vec{x})^{q-1})$$

As a final example, an expression "if ($l_1(\vec{x}) = 0$ *and* $l_2(\vec{x}) = 0$) then $l_3(\vec{x})$ else $l_4(\vec{x})$" can be written as

$$(1 + l_1(\vec{x})^{q-1}) \cdot (1 + l_2(\vec{x})^{q-1}) \cdot (l_3(\vec{x}) + l_4(\vec{x})) + l_4(\vec{x})$$

A **pseudo-linear** multivariate polynomial defined over sub-field $\mathbb{F}_2$ is then a polynomial which is a sum of guarded linear-terms [Dij75]; a *guarded linear-term* is a polynomial which is the product of a linear (over $\mathbb{F}_2$) polynomial and zero or more linear-guards; a *linear-guard* is a linear (over $\mathbb{F}_2$) polynomial raised to the power q-1. Since, in this paper we will only be dealing with pseudo-linear polynomials defined over $\mathbb{F}_2$, from now on we will implicitly assume that. A pseudo-linear polynomial in n variables and defined over $\mathbb{F}_2$, however does yield a function from $(\mathbb{F}_q)^n$ to $\mathbb{F}_q$, which we call a **pseudo-linear function**. Thus, even though the polynomial is defined over $\mathbb{F}_2$, the underlying field will be $\mathbb{F}_q$, and hence the algebra of the polynomials is modulo $(x_i^q = x_i)$ (for i ranging from 1 to n). In formal terms, the objects in consideration are in $\mathbb{F}_2[x_1, ..., x_n]/(x_1^q + x_1, ..., x_n^q + x_n)$. They are also further restricted by the fact that all expressions in the guards are linear instead of affine, but we will later see how to introduce constant additive terms from $\mathbb{F}_q$ (Appendix A).

We observe that pseudo-linear polynomials are closed under pseudo-linear transformations, i.e. given a pseudo-linear polynomial, raising it to the power q-1, and multiplying it by another pseudo-linear polynomial yields just another pseudo-linear polynomial. This follows (by induction) from the observation that

$$(f_1(\vec{x})^{q-1} l_1(\vec{x}) + f_2(\vec{x}))^{q-1} = f_1(\vec{x})^{q-1}(l_1(\vec{x}) + f_2(\vec{x}))^{q-1} + (1 + f_1(\vec{x})^{q-1}) \cdot f_2(\vec{x})^{q-1}$$

where f_1 and f_2 are arbitrary pseudo-linear functions. The observation itself follows by considering the two cases where $f_1(\vec{x})$ is zero or not.

More importantly, the if-then-else programs mentioned above compute exactly the pseudo-linear functions. A more detailed description of such programs and how they relate to pseudo-linear functions can be found in Appendix C.

Above, we saw how a multivariate polynomial $p(\vec{x})$ yields a function from $\mathbb{F}_q^n$ to $\mathbb{F}_q$. More generally, if we are given n polynomials $f_1(\vec{z})$ to $f_n(\vec{z})$ (where $\vec{z}$ are k formal variables), then $p(f_1(\vec{z}), ..., f_n(\vec{z}))$ yields a function from $\mathbb{F}_q^k$ to $\mathbb{F}_q$, which we say is a **pseudo-linear function** of $f_1, ..., f_n$.

While for linear multivariate functions a *completeness theorem* which states that if a linear function f of n variables is a function of k other linear functions (in the same n variables), then f must be a *linear function* of the k linear functions, is well known and rather easy to prove, a similar completeness result for pseudo-linear functions is novel and not so easy to prove.

Thus, the main result of this paper is a theorem which states that if a pseudo-linear multivariate function f of n variables is a *function* of k pseudo-linear functions $f_1, f_1, ..., f_k$ (in the same n variables), then f must be a pseudo-linear function of $f_1, f_2, ..., f_k$. Note that it is given that f itself is a pseudo-linear function in the original n variables.

This theorem has consequences for bounding the time complexity of a search algorithm which seeks to find an arbitrary program (if any) which simulates an if-then-else program using given (multiple) if-then-else programs. Our theorem shows that one can restrict the search for the arbitrary program to pseudo-linear functions, hence reducing the time complexity from being dependent on the field size to just being dependent on the program sizes. For cryptographic applications, this means that an algorithmic search for a simulator in proving a protocol secure in the universally composable model [Can01] is *independent* of the security parameter, as the security parameter is usually related to the field size. Since the program sizes in cryptographic protocols are usually small, this can lead to efficient theorem proving as we discuss later. There are additional issues involved, as the various functionalities, the adversary and the simulator have access to random coins, and the simulation need only be computationally indistinguishable. These issues are discussed in Appendix C. Although, there have been many pieces of work in formal methods for cryptographic protocols [AR00, CH06, MW04, DDMR07], this to the best of our knowledge is a novel approach to theorem proving of security protocols.

To model the fact that the Simulator can iteratively compose various calls to the different functions in the ideal functionality, we *prove a more general theorem* involving arbitrary iterations of k functions $f_1(\vec{z}, \vec{y}), f_2(\vec{z}, \vec{y}), ..., f_k(\vec{z}, \vec{y})$, where $\vec{y}$ are arguments which the Simulator can supply. We prove that if f is a pseudo-linear function of $\vec{z}$, and can be computed by a sub-exponential (in m) length iterated composition of the given k functions, then it can be computed by a *probabilistic* iterated pseudo-linear composition of the given k functions. Note that in this case, we are only able to prove that a randomized pseudo-linear simulator exists. Deciding whether an efficient deterministic pseudo-linear simulator of the given k functions exists remains an open problem. Also note the technical restriction of a sub-exponential length iterated composition which is required to rule out deterministic brute force searches, which a computationally bounded simulator is not allowed anyway. Finally, we remark that our completeness results require sufficiently large fields (as a function of the number of variables in f), but given that most UC proofs only seek proofs of simulatability which do not depend on the security parameters,

our completeness theorem covers all such UC proofs.

The difficulty in proving the completeness theorem stems from the fact that pseudo-linear polynomials can have individual degrees (i.e. of individual variables) exceeding q-1, and hence it may be subject to reduction modulo $x^q = x$. Similar problems occur in local testing of low degree polynomials [JPRZ04, KR04], and we would like to point out that pseudo-linear functions are intimately related to Generalized Reed-Muller Codes [KLP68]. Thus, for example it is not immediately clear what constitutes a basis for pseudo-linear polynomials. We first show a basis for pseudo-linear polynomials, and then show a necessary and sufficient condition involving the basis for a pseudo-linear function of $\vec{x}$ to be a pseudo-linear function of other pseudo-linear functions of $\vec{x}$. A detailed example illustrating these issues and the proof idea can be found in Appendix B.1.

We remark that our theorem does not yet address stateful or persistent-state functions, but we expect to prove similar results for stateful functions in the near future. We also expect to extend our results to other groups and cryptographic constructs with proper axiomatization.

We would also like to mention that the class of pseudo-linear functions do not form an ideal in $F_q[\vec{x}]$, and hence the vast field of Gröbner basis is not applicable.

The rest of the paper is organized as follows. Section 2 describes and proves a basis for pseudo-linear functions. Section 3 proves an interpolation theorem for pseudo-linear function. Section 4 proves the Completeness theorem for pseudo-linear functions. Section 5 defines iterated composition of pseudo-linear functions and proves the Completeness theorem for iterated pseudo-linear functions.

2 A Basis for Pseudo-Linear Functions

In this section we fix a field $\mathbb{F}_q$ of size $q = 2^m$.

Let $\mathcal{L}$ stand for all linear expressions (including zero) in n variables, say $x_1, x_2, ..., x_n$ (the unordered collection will be referred to as X). We define the set of **elementary pseudo-linear (**EPSELIN**)** polynomials to be all polynomials of the form

$$\prod_{l \in J}(1 + l(\vec{x})^{q-1}) \cdot \prod_{l \in \mathcal{L} \setminus J} l(\vec{x})^{q-1} \cdot p(\vec{x})$$

where $p(\vec{x})$ is in $\mathcal{L}$, and J is any subset of $\mathcal{L}$ such that it is closed under addition, i.e. J is a subspace of $\mathcal{L}$. We also include the zero polynomial amongst the elementary pseudo-linear polynomials. Note that if $\mathcal{L} \setminus J$ included a linearly-dependent term of J, then the above polynomial reduces to zero in $\mathbb{F}_q$.

Generalizing (and specializing) the earlier definition of a guard, we will refer to expressions of the form

$$\prod_{l \in J}(1 + l(\vec{x})^{q-1}) \cdot \prod_{l \in \mathcal{L} \setminus J} l(\vec{x})^{q-1}$$

as **guards**.

For the next definition, we will require that the n variables be ordered by their indices. Thus x_1 is considered to be of lesser index then x_2, and so on. This also induces a lexicographic ordering on all equal-sized subsets of the n variables X.

An elementary pseudo-linear polynomial with the above notation will be called a **reduced elementary pseudo-linear (**REPSELIN**)** polynomial if it satisfies the following:

1. Let r be the rank of J ($r \leq \min(n, |J|)$).
2. Let R be the lexicographically greatest set of r variables occuring in J which can be expressed in terms of smaller indexed variables (or just zero) when J is set to zero. This for example, can be accomplished by considering a row-echelon normal form of J.
3. None of the variables in R occur in $p(\vec{x})$.

To justify this definition, we note that if an elementary pseudo-linear polynomial is not reduced, then it is equivalent to a reduced one.

One implication of the above definition is that if $p(\vec{x})$ is non-zero then it itself cannot be in J. Recall, J is closed under addition, by definition of EPSELIN-polynomials. Let r be the rank of J. Let $\bar{J}$ be the r sized subset of J which forms a basis of J, and which define the variables R by the row-echelon normal form of J. Thus, all $l(\vec{x})$ in J *must have at least one variable from* R. Thus, $p(\vec{x})$ cannot be in J.

Finally, we define a REPSELIN-polynomial to be a **basic pseudo-linear** polynomial if the linear term $p(\vec{x})$ is just a variable from X. Note that since the basic polynomial is REPSELIN, from item (3) above it follows that this variable is not from R.

Next we argue that any pseudo-linear polynomial can be expressed as a (xor-) sum of basic pseudo-linear polynomials. For this, we just need to show that any pseudo-linear polynomial of the form

$$\prod_{l \in \mathcal{L} \setminus J} l(\vec{x})^{q-1} \cdot p(\vec{x})$$

where J is a subset of $\mathcal{L}$, can be expressed as sum of EPSELIN-polynomials. This follows easily by induction on the size of J, and by noting that pseudo-linear polynomials with guards

$$\prod_{l \in \mathcal{L} \setminus J} l(\vec{x})^{q-1} \cdot \prod_{l \in J} (1 + l(\vec{x})^{q-1})$$

such that $\mathcal{L} \setminus J$ includes a linearly dependent expression of J are identically zero.

We will now show that the basic pseudo-linear polynomials actually form a *basis* for pseudo-linear polynomials. Before that we need some more notation.

Let $\mathcal{Q}(X)$ be the set of all basic pseudo-linear polynomials in variables X. Further, let $\mathcal{G}(X)$ be the set of all guards amongst these polynomials $\mathcal{Q}(X)$. Let $|\mathcal{G}(X)| = t$. The guards can then be named w.l.o.g. g_1, g_2, ...,g_t. Recall, for each guard g_i, there is associated a subset of variables X, namely R, that do not

occur in any linear terms $p(\vec{x})$. We refer to all linear combinations of $X \backslash R$ as $\mathcal{P}_i(X)$, including the linear term zero. Let $|\mathcal{P}_i(X)| = s_i + 1$. (Note, $(s_i + 1)$ is two to the power size of the subset of variables associated with g_i.) The linear terms in $\mathcal{P}_i(X)$ can be named $p_i^j(\vec{x})$, j ranging from 0 to s_i (not to be confused with exponent). W.l.o.g., zero will always be $p_i^0(\vec{x})$.

Thus, any pseudo-linear function $\phi(\vec{x})$ can be represented as a sum (over $\mathbb{F}_2$) of polynomials from $\mathcal{Q}(X)$, i.e.,

$$\phi(\vec{x}) \ = \ \sum_{i \in T} g_i(\vec{x}) \cdot p_i^{j(\phi,i)}(\vec{x})$$

where T is a subset of $[1..t]$, and each $p_i^{j(\phi,i)}(\vec{x}) \in \mathcal{P}_i(X)$. In fact, we do not even need to take a subset T of $[1..t]$; all zero terms just imply that $j(\phi, i) = 0$, by our notation above that $p_i^0(\vec{x})$ is always taken to be zero. Thus the above representation of $\phi(\vec{x})$ is totally defined by the map $j(\phi, \cdot)$.

While we state and prove the following theorem only for large fields, as only for such fields are the basic pseudo-linear polynomials a basis, a slightly more complicated characterization can be given for smaller fields.

Lemma 1 (Basis) *For Fields of size $q > 2^n$, the basic pseudo-linear polynomials in n variables form a basis for pseudo-linear polynomials in n variables.*

Proof:

We have already shown above that any pseudo-linear polynomial can be represented as a sum of basic pseudo-linear polynomials, in fact defined by the map $j(\phi, \cdot)$ above. So, here we focus on showing that any pseudo-linear function $\phi(\vec{x})$ has a unique such representation.

So, for the sake of contradiction, suppose the everywhere zero function $\mathbf{0}$ has a non-zero representation, and let that be represented by the map $j(\mathbf{0}, i)$. Now consider any $\bar{i}$ where $j(\mathbf{0}, \bar{i}) \neq 0$, and lets call $p_i^{j(\mathbf{0},i)}(\vec{x})$ by just $p(x)$ $(\neq 0)$. In other words, this representation of $\mathbf{0}$ has the term

$$g_{\bar{i}} \cdot p(\vec{x})$$

Let,

$$g_{\bar{i}} = \prod_{l \in \mathcal{L} \backslash J} l(\vec{x})^{q-1} \ \cdot \ \prod_{l \in J} (1 + l(\vec{x})^{q-1})$$

for some subspace $J \subseteq \mathcal{L}$. Let the rank of J be r. Let R be the lexicographically greatest set of r variables occuring in J which can be expressed in terms of smaller indexed variables when J is set to zero. Recall, by definition, none of the variables in R occur in $p(\vec{x})$.

Claim: With the set of equations J set to zero, we can solve for all linear expressions in $\mathcal{L} \backslash J$ to be non-zero, and hence also set $p(\vec{x})$ to non-zero.

This would first of all imply that all guards other than $g_i(\vec{x})$ evaluate to zero: if the guard $g_a(\vec{x})$ is given by subset $J_a \subseteq \mathcal{L}$ $(J_a \neq J)$, then if $J \backslash J_a$ is non-empty,

we get that $\mathcal{L}\backslash J_a$ has an $l(\vec{x})$ from J which makes $l(x)^{q-1}$ zero, and if $J_a\backslash J$ is non-empty, we get that J_a has an $l(\vec{x})$ from $\mathcal{L}\backslash J$ which makes $(1 + l(\vec{x})^{q-1})$ zero.

Further, the guard $g_{\bar{i}}(\vec{x})$ will be non-zero, and hence $g_{\bar{i}}(\vec{x})p(\vec{x})$ would be non-zero, and consequently the given representation $j(\mathbf{0}, i)$ leads to a non-zero function, a contradiction, which would prove the lemma.

Now, to prove the above claim, recall that J is closed under addition, and $p(\vec{x})$ is in $\mathcal{L}\backslash J$. Let r be the rank of J. Consider a basis $\bar{J}$ of a complementary subspace of J. If our underlying field is of size at least 2^{n-r}, we can set J to zero, and each $l_i(\vec{x})$ of $\bar{J}$ ($i \in [1..n-r]$) to e_i, where the e_i ($i \in [1..n-r]$) are linearly independent over $\mathbb{F}_2$. Thus, all linear expressions in $\mathcal{L}\backslash J$ evaluate to non-zero values, as any l in $\mathcal{L}\backslash J$ is a non-trivial linear combination of $\bar{J}$ plus an l' from J. □

Note on small fields. In smaller fields some of the basic pseudo-linear polynomials, which are non-trivial functions in large fields, turn out to be identically zero. Thus the basis is smaller, but more complicated to characterize.

Lemma 2 (Homomorphism) *For any pseudo-linear functions $\phi_1(\vec{x})$ and $\phi_2(\vec{x})$, and for all $i \in [1..t]$,*

$$p_i^{j(\phi_1+\phi_2,i)} = p_i^{j(\phi_1,i)} + p_i^{j(\phi_2,i)}$$

Proof: Follows from the fact that the basic pseudo-linear polynomials form a basis for pseudo-linear polynomials. □

3 Interpolation Property for Pseudo-Linear Functions

Before we prove the main theorem, we need a few more definitions and related lemmas.

Let $f_1, f_2, ..., f_k$ be k pseudo-linear functions in n variables X, over a field $\mathbb{F}_q$ ($q = 2^m$). Collectively, we will refer to these polynomials as F.

For any pseudo-linear polynomial $f(\vec{x})$ in X, let its representation in terms of the basis be given by $j(f, \cdot)$. Since each of the polynomials from F, i.e. $f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})$ is pseudo-linear, it be represented by $j(f_s, \cdot)$ ($s \in [1..k]$). Further, each linear combination of F is represented similarly.

We say that two guards $g_a(\vec{x})$ and $g_b(\vec{x})$ are F-**equivalent** if for every linear combination ϕ of functions from F, it is the case that $j(\phi, a) = 0$ iff $j(\phi, b) = 0$. In this case, we write $a \cong_F b$, which is an equivalence relation.

Lemma 3 *If a and b are F-equivalent then if for some subset $S \subseteq [1..k]$, the linear combination $\sum_{s\in S} p_a^{j(f_s,a)}$ is identically zero , then so is $\sum_{s\in S} p_b^{j(f_s,b)}$.*

The lemma follows by Lemma (2). Thus, if k' is the rank of $p_a^{j(f_s,a)}$ ($s \in [1..k]$), then it is also the rank of $p_b^{j(f_s,b)}$. In fact, we can take the exact same k' indices from ($s \in [1..k]$), w.l.o.g. $[1..k']$, to represent the basis for the k linear expressions, for both a and b.

Let $\mathcal{L}(F)$ denote the set of all linear combinations of functions in F.

For any function $f(\vec{x})$, and any set F of pseudo-linear functions in X, we say that $f(\vec{x})$ has the F**-interpolatable** property if it satisfies the following two conditions:

(i) $\forall i \in [1..t] : \exists \phi_\star \in \mathcal{L}(F) \ : \ j(f,i) = j(\phi_\star, i)$, and

(ii) For every $a, b \in [1..t]$ such that a and b are F-equivalent, w.l.o.g. by Lemma (3), let the first k' functions out of (k functions) $p_a^{j(f_s,a)}$ (out of $p_b^{j(f_s,b)}$), represent their basis (resp. for b). Then, if the $\phi_\star$ in (i) is given by $\sum c_s^a p_a^{j(f_s,a)}$ and $\sum c_s^b p_b^{j(f_s,b)}$, respectively for a and b, then for all $s \in [1..k']$, $c_s^a = c_s^b$.

Lemma 4 *If f is a pseudo-linear function in X, and f satisfies the F-interpolatable property, for some set F of pseudo-linear polynomials in X, then f is a pseudo-linear function of F.*

Proof: Indeed, consider $\hat{T} = [1..t] \ / \cong_F$, where t is the number of guards, i.e. $|\mathcal{G}(X)|$. We pick the smallest elements from $[1..t]$ to represent each equivalence class in $\hat{T}$. Define a function $h(\vec{x})$ to be the following:

$$h(\vec{x}) = \sum_{u \in \hat{T}} \prod_{\phi \in \mathcal{L}(F): j(\phi,u) \neq 0} \phi(\vec{x})^{q-1} \ \cdot \prod_{\phi \in \mathcal{L}(F): j(\phi,u) = 0} (1 + \phi(\vec{x})^{q-1}) \ \cdot \ \phi_u(\vec{x}) \qquad (1)$$

where for each u, ϕ_u is some function $\phi_\star$ satisfying the F-interpolatable property (i) above.

Now by definition, $h(\vec{x})$ is pseudo-linear in F. We now show that $h{=}f$, i.e. for all $\vec{x} \in (\mathbb{F}_q)^n$, $h(\vec{x}) = f(\vec{x})$. Fix any $\vec{x}^*$ in $(\mathbb{F}_q)^n$. Let $J \subseteq \mathcal{L}$, such that all linear functions in J evaluate to zero at $\vec{x}^*$, and all linear functions in $\mathcal{L} \backslash J$ evaluate to non-zero quantities at $\vec{x}^*$. Clearly, J is closed under addition, and hence J corresponds to a guard g_i. In other words, $g_i(\vec{x}^*) = 1$, and for all other $i' \in [1..t]$: $g_{i'}(\vec{x}^*) = 0$. Thus, $f(\vec{x}^*) = p_i^{j(f,i)}(\vec{x}^*)$, and similarly, for all $\phi \in \mathcal{L}(F)$, $\phi(\vec{x}^*) = p_i^{j(\phi,i)}(\vec{x}^*)$. By definition of i (i.e. g_i corresponding to J above, and hence $p_i^{j(\phi,i)} \in \mathcal{L} \backslash J$), it follows that $\phi(\vec{x}^*)$ is zero iff $j(\phi, i) = 0$.

Now, in equation (1), we show that the only u for which the "guards" evaluate to be non-zero (i.e. one), is the one corresponding to the equivalence class of i in $\hat{T}$ (say, u_i). In fact, for i (and its F-equivalent u_i) the "guards" indeed evaluate to 1. For all other i', if the "guards" evaluate to one, then by definition of F-equivalence, those i' are F-equivalent to i.

Thus, $h(\vec{x}^*) = \phi_{u_i}(\vec{x}^*)$, and since ϕ_{u_i} is pseudo-linear in X,

$$\phi_{u_i}(\vec{x}^*) = p_i^{j(\phi_{u_i},i)}(\vec{x}^*) = \sum_s c_s^{u_i} p_i^{j(f_s,i)}(\vec{x}^*) = \sum_s c_s^i p_i^{j(f_s,i)}(\vec{x}^*) = p_i^{j(\phi_i,i)}(\vec{x}^*).$$

Thus, $h(\vec{x}^*) = p_i^{j(f,i)}(\vec{x}^*)$, which is same as $f(\vec{x}^*)$. □

4 The Completeness Theorem for Pseudo-Linear Functions

While the main completeness theorem below is stated and proven for only large finite fields, it holds for all finite fields of characteristic two.

Theorem 5 *Let $f_1, f_2, ..., f_k$ be k pseudo-linear functions in n variables X, over a field F_q ($q = 2^m$), such that $q > 2^n$. Collectively, we will refer to these polynomials as F. Let f be another pseudo-linear function in X. Then, if f is a function of F, then f is a pseudo-linear function of F.*

Proof: We show that if f is not a pseudo-linear function of F, which by Lemma (4) means that it does not satisfy at least one of F-interpolatable properties (i) or (ii), then f is not a function of F.

Since $f(\vec{x})$ is a pseudo-linear polynomial in X, let its representation in terms of the basis be given by $j(f, \cdot)$. Since each of the polynomials from F, i.e. $f_1(\vec{x})$, $f_2(\vec{x})$,...., $f_k(\vec{x})$ is pseudo-linear, it can also be represented by $j(f_s, \cdot)$ ($s \in [1..k]$). Further, each linear combination of F is represented similarly.

So, first consider the case where f does not satisfy (i). In other words, for some $i \in [1..t]$, for no linear combination ϕ of F (including zero) is $j(f, i)$ equal to $j(\phi, i)$. Thus, by Lemma (2), $p_i^{j(f,i)}$ is linearly independent of all $p_i^{j(f_s,i)}$ ($s \in [1..k]$). Let $J \subseteq \mathcal{L}$ correspond to the guard g_i. Thus, $p_i^{j(f,i)}$ and all $p_i^{j(f_s,i)}$ (for $s \in [1..k]$) are linearly independent of J. Let r be the rank of J, and $k' \leq k$ be the rank of $p_i^{j(f_s,i)}$ collectively. Since $p_i^{j(f,i)}$ is linearly independent of all $p_i^{j(f_s,i)}$, we have that $r + k' + 1 \leq n$. Now, the subspace corresponding to J set to zero has dimension $n - r$, and hence has q^{n-r} points. However, we are interested in points where all expressions in $\mathcal{L}\backslash J$ evaluate to non-zero values, which would guarantee that $g_i = 1$, and all other guards are zero. Recall the subspace $\mathcal{P}_i(X)$ generated by all variables not in the set R corresponding to guard g_i. Now, $\mathcal{L}\backslash J$ is a union of cosets of $(n-r)$ dimensional space $\mathcal{P}_i(X)$ shifted by subspace J. Consider a basis $\mathcal{B}$ for $\mathcal{P}_i(X)$, comprising of $p_i^{j(f,i)}$, a k'-ranked basis of $p_i^{j(f_s,i)}$, and $n - r - 1 - k'$ other linearly independent expressions $\mathcal{B}'$.

Assume the field $\mathbb{F}_q$ is of size at least 2^{n+1}, and hence has $n + 1$ linearly independent (over $\mathbb{F}_2$) elements e_i. Thus, for every injective map setting $\mathcal{B}$ to these e_i, there is a distinct solution to J being zero, and all of $\mathcal{L}\backslash J$ evaluating to non-zero values. Thus, there are at least $\binom{n+1}{n-r}(n-r)!$ such points in $(\mathbb{F}_q)^n$.

So, we fix $p_i^{j(f_s,i)}$ to e_s ($s \in [1..k']$; assume w.l.o.g. that the first k' formed the basis), and similarly fix the $\mathcal{B}'$ expressions to $e_{k'+1}$ to e_{n-r-1}. This still leaves at least $(n + 1 - (n - r - 1))$ choices for $p_i^{j(f,i)}$. Thus, we have the situation that there are two points in $(\mathbb{F}_q)^n$ where f evaluates to different values, whereas F has the same value, and hence f cannot be a function of F.

Now, consider the case where f does satisfy condition (i), but condition (ii) is violated. In other words, for each $i \in [1..t]$, $p_i^{j(f,i)}$ is same as some $p_i^{j(\phi_\star,i)}$, but there exist a and b in $[1..t]$ which are F-equivalent, but the $\phi_\star$'s linear representation

coefficients c_s differ for a and b. Again, we will demonstrate two points where F evaluate to the same value, but f evaluates to different values.

Again, let's assume that the underlying field is large enough to have at least k' linearly independent (over $\mathbb{F}_2$) elements, say e_i.

Now we have two sets, J_a corresponding to guard g_a, and J_b, corresponding to guard g_b. However, there is an easy solution for setting J_a to zero, and setting $p_a^{j(f_s,a)}$ ($s \in [1..k']$) to e_s. Similarly, there is a solution for setting J_b to zero and setting $p_a^{j(f_s,a)}$ ($s \in [1..k']$) to e_s. Thus, in both cases all f_s ($s \in [1..k]$) evaluate to the same value, but f being a different linear combination in the two cases, evaluates to different values. □

5 Iterated Composition of Pseudo-Linear Functions

In this section, we consider pseudo-linear functions which can take arguments, modeling oracles which are pseudo-linear functions of secret values and arguments. Thus, for instance it may be required to find if there exists a simulator which given access to functionalities which are pseudo-linear functions of secret parameters X and arguments supplied by simulator/adversary, can compute a given a pseudo-linear function.

This generalizes the problem from the previous sections, where the simulator could not pass any arguments to the given functions. For simplicity, we will deal here with functions which only take a single argument, and thus all the functions can be written as $f_i(\vec{x}, y)$, each pseudo-linear in $\vec{x}$ and y.

So, given a collection of k pseudo-linear functions $F(X, y)$, we now define an **iterated composition of** F. Let $\mathbb{F}_q$ be the underlying field as before. An iterated composition σ of F is a length t sequence of pairs (t an arbitrary number), the first component of the s-th ($s \in [1..t]$) pair of σ being a function ϕ_s from F, and the second component an arbitrary function γ_s of $s-1$ arguments (over $\mathbb{F}_q$).

Given an iterated composition σ of F, one can associate a function f^σ of X with it as follows by induction. For σ of length one, f^σ is just $\phi_1(\vec{x}, \gamma_1())$, recalling that $\phi_1 \in F$. For σ of length t,

$$f^\sigma(\vec{x}) = \phi_t(\vec{x}, \gamma_t(f^{\sigma_{|1}}(\vec{x}), f^{\sigma_{|2}}(\vec{x}), ..., f^{\sigma_{|t-1}}(\vec{x})))$$

where $\sigma_{|j}$ is the prefix of σ of length j.

Since, functions in n variables over $\mathbb{F}_q$ are just polynomials in n variables, there is a finite bound on t, after which no iterated composition of F can produce a new function of the n variables. The collection of all functions that can be obtained by iterated composition of F will be referred to as **terms(F)**. If we restrict γ_s to be pseudo-linear functions of their $s-1$ arguments, we will refer to the iterated composition as **pseudo-linear iterated composition of** F, and the corresponding collection of functions associated with such sequences as pseudo-linear iterated terms or **pl-terms(F)**. Note that in this case γ_1 is just zero.

Note that an arbitrary program can only compute a function of the terms,

whereas an arbitrary pseudo-linear program can only compute a pseudo-linear function of the pseudo-linear terms. We would *like to show* that if a function f of terms(F) is a pseudo-linear function of X, then it is a pseudo-linear function of pl-terms(F). However, as we demonstrate in Appendix B.2, this is not true in general, and a slight extension is required to the pl-terms, so as to enable probabilistic functions. An iterated composition will be called an **extended pseudo-linear iterated composition of** F if γ_s is either a pseudo-linear function or a constant function $c(\vec{x})$ evaluating to an element c in $\mathbb{F}_q$. For each such c, the corresponding collection of functions associated with such sequences will be called **extended-pl-terms(F, c)**.

We will also need to refine the definition of terms(F), by restricting to terms obtained within some T iterated compositions, for some positive integer T. Thus, $\mathbf{terms}_T(F)$ will stand for the collection of functions obtained by iterated compositions of F of length less than T. In particular we will be interested in T which is bounded by polynomials in $\log q$ and/or n, the number of variables in X.

Similar to Section 3, we first state an interpolatable property which is a sufficient condition for a pseudo-linear function of X to be a pseudo-linear function of pl-terms(F).

Recall the functions in F now have an additional argument y. As before, $\mathcal{L}(G)$, for any set of functions G will denote the set of all linear combinations (over $\mathbb{F}_2$) of functions from G. Below we define the class $\mathcal{I}^i(F)$ of pseudo-linear functions in X, for i an arbitrary natural number. In fact, since the inductive definition will sometimes use functions in both X and y, we will just define this class as pseudo-linear functions in X and y, though for different y, they would evaluate to the same value. In other words, for an arbitrary guard $g_a(\vec{x})$, which corresponds to a subset $J \subseteq \mathcal{L}(X)$ (J is closed under addition), there are many **super-guards** when viewed as a function of X and y, namely with subsets $J' \subseteq \mathcal{L}(X, y)$ (J' closed under addition) such that $J \subseteq J'$ *and* $(\mathcal{L}(X) \backslash J) \subseteq (\mathcal{L}(X, y) \backslash J')$. Thus, for all these super-guards, a pseudo-linear function $\phi(X)$ will have the same $j(\phi, \cdot)$ value (see Section 2).

However, and more importantly, with y set to some linear expression $l(\vec{x}) \in \mathcal{P}_a(X)$ (including zero), *exactly one* of these (super-)guards has the property that $J'_{y|l(\vec{x})} = J$ (Note the subscript $y|l(\vec{x})$ means $l(\vec{x})$ is substituted for every occurrence of y in J'). This particular J' is given by

$$J' \;=\; \mathcal{L}(J, \{y + l(\vec{x})\})$$

In this case we say that this super-guard of g_s is consistent with $y + l(\vec{x}) = 0$. The super-guard corresponding to $J' = J$ will be called the **degenerate super-guard** of $g_a(\vec{x})$.

Now we define the pseudo-linear function which is the composition of f_s and h, i.e. $f_s \circ h$, where f_s is a pseudo-linear function in X and y, and h is a pseudo-linear function in X, by defining its components in the basis for pseudo-linear functions. For any guard $g_i(\vec{x})$ (of functions in X), let $g_I(\vec{x}, y)$ be the unique (super-) guard, mentioned in the previous paragraph, which is consistent with y set to $p_i^{j(h,i)}$ (note the map j here is for guards corresponding to X, and in general it will be clear from

context whether we are referring to map j for guards corresponding to X or X, y). Then, define

$$p_I^{j(f_s \circ h, I)}(\vec{x}, y) = p_I^{j(f_s, I)}(\vec{x}, p_i^{j(h,i)}(\vec{x}, y))$$

Further, for all I' which are super-guards of i, we set $p_{I'}^{j(f_s \circ h, I')}$ to be the same value (as $f_s \circ h$ is only a function of X). Note that since each p is just a linear function, this implies that each component of $f_s \circ h$ is a linear function of X (and hence X, y). In particular, $(f_s \circ h)(\vec{x}) = f_s(\vec{x}, h(\vec{x}))$.

Define **Compose**$(F(X, y), H(X))$, where $F(X, y)$ are a set of pseudo-linear functions in X, y and $H(X)$ is a set of pseudo-linear functions in X, to be the set of all functions $f_s \circ h$, where $f_s \in F(X, y)$ and $h \in H(X)$.

For each pseudo-linear function f_s of X and y, we also need to define a pseudo-linear function (in X called **degenerate(f_s)**, which for each guard $g_a(\vec{x})$, defines the corresponding p function using its degenerate super-guard. Thus,

$$p_a^{j(\text{degenerate}(f_s), a)}(\vec{x}) = p_I^{j(f_s, I)}(\vec{x}, 0),$$

where I is the degenerate super-guard of g_a.

Now, we are ready to define the iterated pseudo-linear functions. Define

$$\begin{aligned} \mathcal{I}^0(F) &= \mathcal{L}(\text{Compose}(F, \text{degenerate}(F))) \\ \mathcal{I}^{i+1}(F) &= \mathcal{L}(\mathcal{I}^i(F) \cup \text{Compose}(F, \mathcal{I}^i(F))), \text{ for } i \geq 0. \end{aligned}$$

Since, these functions are just polynomials over finite fields (in fact defined over F_2), the above iteration reaches a fix-point at an i bounded by a function only of n. We will denote the fix-point by just $\mathcal{I}(F)$.

Now, we generalize the definitions of F-equivalence and F-interpolatable from Section 3. Two guards $g_a(\vec{x})$ and $g_b(\vec{x})$ are said to be F^*-equivalent if for every $\phi(\vec{x})$ in $\mathcal{I}(F)$, it is the case that $j(\phi, a) = 0$ iff $j(\phi, b) = 0$.

The definition of F^*-interpolatable property is same as the F-interpolatable property except that $\mathcal{L}(F)$ is replaced by $\mathcal{I}(F)$.

Instead of the closure $\mathcal{I}(F)$, it will also be useful to define the following set of functions

$$\mathcal{C} = \bigcup_i \text{Compose}(F, \mathcal{I}^i(F)) \cup \text{Compose}(F, \text{degenerate}(F)),$$

and it is easy to see that $\mathcal{I}(F)$ is just the linear closure of $\mathcal{C}$.

Lemma 6 *If f is a pseudo-linear function of n variables X over a field $\mathbb{F}_q$, and f satisfies the F^*-interpolatable property, for some set F of pseudo-linear polynomials in X, y, then f can be defined as a pseudo-linear probabilistic function of* **extended-pl-terms**$(F(X, y), Seed)$*, where the probability is over Seed chosen uniformly from $\mathbb{F}_q$, and for each $\vec{x}$ the probability of this definition of f being correct is at least $1 - 2^n/q$.*

Proof: The proof is similar to the proof of Lemma 4, but there is a small difference due to the probabilistic nature of this lemma.

Consider $\hat{T} = [1..t] \,/\, \cong_F$, where t is the number of guards, i.e. $|\mathcal{G}(X)|$. We pick the smallest element from $[1..t]$ to represent each equivalence class in $\hat{T}$. Define a function $h(\vec{x})$ to be the following:

$$h(\vec{x}) = \sum_{u \in \hat{T}} \prod_{\phi \in \mathcal{L}(F): j(\phi,u) \neq 0} \phi(\vec{x})^{q-1} \cdot \prod_{\phi \in \mathcal{L}(F): j(\phi,u)=0} (1 + \phi(\vec{x})^{q-1}) \cdot \phi_u(\vec{x}) \qquad (2)$$

where for each u, ϕ_u is some function $\phi_\star \in \mathcal{C}$ satisfying the F^*-interpolatable property (i).

Now by definition, $h(\vec{x})$ is pseudo-linear in $\mathcal{C}$. Now, the proof that $h{=}f$, i.e. for all $\vec{x} \in (\mathbb{F}_q)^n$, $h(\vec{x}) = f(\vec{x})$, is identical to the proof in Lemma 4. However, h is pseudo-linear only on $\mathcal{C}$, whereas we need to show a function pseudo-linear in **extended-pl-terms**$(F(X,y), \text{Seed})$.

Observe that for any f_s, exactly one of the following cases hold, for all y linearly independent of $\vec{x}$:

Case 1: $f_s(\vec{x}, y) = \text{degenerate}(f_s)(\vec{x})$

Case 2: $f_s(\vec{x}, y) = \text{degenerate}(f_s)(\vec{x}) + y$

Now define a function $\hat{h}(\vec{x}, c)$ to be same as h, except every occurrence of degenerate$(f_s)(\vec{x})$ is replaced by either of the following:

$$\begin{cases} f_s(\vec{x}, c) & \text{in Case 1} \\ f_s(\vec{x}, c) + c & \text{in Case 2} \end{cases}$$

(recall, f_s is a function of X and y, whereas degenerate(f_s) is a function of only X). Then, it is easy to see that $\hat{h}(\vec{x}, c)$ is in **extended-pl-terms**(F, c). We next show that for every $\vec{x}$, with c chosen uniformly from $\mathbb{F}_q$, probability that $\hat{h}(\vec{x}, c) = h(\vec{x})$ is at least $1 - 2^n/q$. For each $\vec{x}$, one and only one guard g_s is satisfied. the probability that for this guard, $c = l(\vec{x})$, for some $l(\vec{x}) \in \mathcal{P}_s(X)$ is at most $1/q$. Hence, by union bound, over all possible $l(\vec{x})$, the probability that c equals any $l(\vec{x})$ is at most $2^n/q$, as $|X| = n$. These are the only cases in which degenerate$(f_s)(\vec{x})$ may differ from $f_s(\vec{x}, c)$ or $f_s(\vec{x}, c) + c$, as the case may be. □

Theorem 7 (Main) *Let $f_1, f_2, ..., f_k$ be k pseudo-linear functions in n variables X and an additional variable y, over a field $\mathbb{F}_q$ such that $q > 2^{2n}$. Collectively, we will refer to these polynomials as $F(X, y)$. Let T be a positive integer less than $2^n (< \sqrt{q})$. Let f be another pseudo-linear function in X. Then, if f is a function of* **terms**$_T(F(X, y))$*, then f can be defined as a pseudo-linear probabilistic function of* **extended-pl-terms**$(F(X, y), Seed)$*, where the probability is over Seed chosen uniformly from $\mathbb{F}_q$, and for each $\vec{x}$ the probability of this definition of f being correct is at least $1 - 1/\sqrt{q}$.*

Proof:

For the sake of leading to a contradiction, suppose that f is not a pseudo-linear probabilistic function of **extended-pl-terms**$(F(X,y), Seed)$. Then by Lemma 6, f does not satisfy the F^*-interpolatable property. Hence, as in Theorem 5, we have two cases. Before we go into the analysis of the two cases, we recall a few relevant definitions, and state some useful properties.

Recall from Section 2, that $\mathcal{Q}(X)$ is the set of all basic pseudo-linear polynomials in variables X, and, $\mathcal{G}(X)$ is the set of all guards amongst these polynomials $\mathcal{Q}(X)$. Further, $|\mathcal{G}(X)| = t$. Also, recall for each guard g_s its corresponding set R from its REPSELIN representation, and the corresponding subspace $\mathcal{P}_s(X)$.

Also, recall the super-guards $g_I(\vec{x}, y)$ corresponding to guards $g_i(\vec{x})$. Thus, if J corresponded to g_i, then some J' such that $J \subseteq J' \subseteq \mathcal{L}(X,y)$, corresponds to super-guard g_I. Further, $(\mathcal{L}(X)\backslash J) \subseteq (\mathcal{L}(X,y)\backslash J')$. Hence, if some $y + l(\vec{x})$ is in J', we can w.l.o.g take as the corresponding R' (of g_I) to be $R \cup \{y\}$. Thus, in such cases $p_I(\vec{x}, y)$ are just linear expressions in $X\backslash R$. If on the other hand, for no $l(\vec{x})$ it is the case that $y + l(\vec{x})$ is in J', then $R' = R$, and $p_I(\vec{x}, y)$ will be a linear expression in $(X\backslash R) \cup \{y\}$.

Now we are ready to analyze the two cases.

Case 1: First, consider the case where f does not satisfy property (1) of F^*-interpolatable.

Then, it is the case that there exists an $s \in [1..t]$, such that for every linear polynomial ϕ in $\mathcal{I}(F,S)$, $j(f,s) \neq j(\phi,s)$. Thus, by Lemma 2, $p_s^{j(f,s)}$ is linearly independent of all $p_s^{j(\phi,s)}$, with $\phi \in \mathcal{C}$.

Let $J \subseteq \mathcal{L}$ correspond to the guard g_s. Thus, $p_s^{j(f,s)}$, as well as all $p_s^{j(\phi,s)}$ ($\phi \in \mathcal{C}$) are linearly independent of J (see the paragraph after definition of REPSELIN polynomials). Let r be the rank of J, and k' be the rank of $p_s^{j(\phi,s)}$ collectively. Thus, $r + k' + 1 \leq n$. Consider a basis $\mathcal{B}$ of $\mathcal{P}_s(X)$ consisting of $p_s^{j(f,s)}$, a basis $\mathcal{B}''$ of $p_s^{j(\phi,s)}$, and another linearly independent set $\mathcal{B}'$ of expressions in $X\backslash R$ (of rank $n-r-k'-1$).

Our aim is to demonstrate two different settings of X to values in $\mathbb{F}_q$, such that $f(\vec{x})$ has different values, while all of **terms**$_T(F(X,y))$ have the same value at the two settings. Now, fix a particular length T iterated composition σ of F. We will now show that each of $\gamma_t(f^{\sigma|1}(\vec{x}), f^{\sigma|2}(\vec{x}), ..., f^{\sigma|t-1}(\vec{x}))$, $t \in [1..T]$, as well as $f^{\sigma|t}(\vec{x})$ is a function of only $\mathcal{B}''$, and is independent of $p_s^{j(f,s)}$, and also independent of $\mathcal{B}'$ defined above. Thus in choosing the two different settings for X, we can first set the basis $\mathcal{B}''$ to some value, which will fix the $\gamma(...)$ values, and then we can set the $\mathcal{B}'$ and $p_s^{j(f,s)}$ to two different values, while assuring that all consistency requirements are met.

For the base case, $\gamma_1()$ is clearly not a function of $p_s^{j(f,s)}$, or $\mathcal{B}'$. Now, for the induction step, consider $f^{\sigma|t-1}(\vec{x})$. which is given by

$$\phi_{t-1}(\vec{x}, \gamma_{t-1}(f^{\sigma|1}(\vec{x}), f^{\sigma|2}(\vec{x}), ..., f^{\sigma|t-1}(\vec{x}))).$$

where ϕ_{t-1} is in F. Now, by induction the $\gamma_{t-1}(...)$ expression is not a function of $p_s^{j(f,s)}$ or $\mathcal{B}'$.

Now, it is possible that $\gamma_{t-1}(...)$ is equal to some $p_s^{j(\phi^*,s)}(\vec{x})$ ($\phi^* \in \mathcal{C}$), in which case $\phi_{t-1}(\vec{x}, \gamma_{t-1}(...))$ would just equal $p_I^{j(\phi_{t-1},I)}(\vec{x}, y)$, for I corresponding to the unique super-guard $g_I(\vec{x}, y)$ which is consistent with $y + p_s^{j(\phi^*,s)}(\vec{x}) = 0$. But, $p_I^{j(\phi_{t-1},I)}$ is either $p_s^{j(\phi^{**},s)}(\vec{x})$ ($\phi^{**} \in \mathcal{C}$) or such an expression plus y, by definition of $\mathcal{C}$ and definition of $p_I^{j(f_s \circ h,I)}(\vec{x}, y)$. In either case, it is a function of only $p_s^{j(\phi,s)}(\vec{x})$ ($\phi \in \mathcal{C}$) by induction.

If $\gamma_{t-1}(...)$ is not equal to any $p_s^{j(\phi^*,s)}(\vec{x})$ ($\phi^* \in \mathcal{C}$), we will show that we can choose $\vec{x}$ so as to assure that $\gamma_{t-1}(...)$ is not equal to any linear expression in $\mathcal{B}'$ (and $p_s^{j(f,s)}$) either, as $t < T < 2^n < \sqrt{q}$. In this case $\phi_{t-1}(\vec{x}, \gamma_{t-1}(...))$ returns $p_I^{j(\phi_{t-1},I)}(\vec{x}, y)$, where I corresponds to the degenerate super-guard of g_s given by $J' = J$. However, such $p_I^{j(\phi_{t-1},I)}(\vec{x}, y)$ is again either $p_s^{j(\phi^{**},s)}(\vec{x})$ ($\phi^{**} \in \mathcal{C}$) or such an expression plus y, since $\mathcal{C}$ includes Compose(F, degenerate(F)).

Now, we demonstrate the two different settings of X to values in $\mathbb{F}_q$. We first choose k' linearly independent (over $\mathbb{F}_2$) values in $\mathbb{F}_q$ and set the basis $\mathcal{B}''$ to these values, so that all expressions $p_s^{j(\phi,s)}$ ($\phi \in \mathcal{C}$) are non-zero. As explained above, the values $\gamma_t(...)$ are then fixed, and let this set of values along with zero be collectively called Γ. Next, we inductively assign values to the basis $\mathcal{B}'$(of size $n - r - k' - 1$) as follows. Let this basis be given by $l_1(\vec{x})$, ..., $l_{n-r-k'-1}(\vec{x})$. For $l_1(\vec{x})$, we pick any value in $\mathbb{F}_q$ which is not equal to any value in $\mathcal{L}(\mathcal{B}'') + \Gamma$, where the sum of two sets is defined naturally. For, the induction step, we choose for $l_i(\vec{x})$ a value in $\mathbb{F}_q$ which is not equal to any value in $\mathcal{L}(\mathcal{B}'' \cup \{l_1(\vec{x}), ..., l_{i-1}(\vec{x})\}) + \Gamma$.

Since $p_s^j(f, s)(\vec{x})$ is linearly independent of $\mathcal{B}''$ (as well as $\mathcal{B}'$), we choose a value for it which is not equal to any value in $\mathcal{L}(\mathcal{B}'' \cup \{l_1(\vec{x}), ..., l_{n-r-k'-1}(\vec{x})\}) + \Gamma$. Further we have at least two choices for it, given that $\mathbb{F}_q \geq 2 + 2^{n-1-r} \times (T + 1)$. This also proves our claim that Γ is never equal to any linear expression in $\mathcal{B}' \cup \{p_s^{j(f,s)}\}$. Further no linear combination of $\mathcal{B}'$ and $\mathcal{B}''$ will be zero. Then, we can choose the R variables corresponding to the guard g_s, which by definition of R are given in terms of the variables already chosen, so that the guard g_s is true in both cases.

Case 2: Now consider the case where condition (i) holds, but condition (ii) of the F^*-interpolatable property fails to hold for f. In other words, for each $i \in [1..t]$, $p_i^{j(f,i)}$ is same as some $p_i^{j(\phi_*,j)}$ ((for $\phi_* \in \mathcal{C}$), but there exist a and b in $[1..t]$ which are F^*-equivalent, but the ϕ_*'s linear representation coefficients c_s differ for a and b. Again, we will demonstrate two points where terms$_T(F(X, y))$ evaluate to the same value, but f evaluates to different values.

We have two sets, J_a corresponding to guard g_a, and J_b, corresponding to guard g_b. Let $k' \leq n$ be the rank of $p_a^{j(\phi,a)}$ ($\phi \in \mathcal{C}$). Let r_a be the rank of J_a, and r_b be the rank of J_b. thus, $r_a + k' \leq n$, and $r_b + k' \leq n$. Let the R sets corresponding to guards g_a and g_b be called R_a and R_b respectively. Let $\mathcal{B}'_a$ be a basis for $X \backslash R_a$ excluding $\mathcal{L}(\mathcal{B}'')$, and similarly $\mathcal{B}'_b$ be a basis for $X \backslash R_b$ excluding $\mathcal{L}(\mathcal{B}'')$. We set the basis $\mathcal{B}''$ of $p_a^{j(\phi,a)}$ to linearly independent over GF2 values e_1 to $e_{k'}$. We set the basis of $p_b^{j(\phi,b)}$ also to the same values, recalling that the two bases, one for a and the other for b, can be chosen to have the same indices. Thus, all functions in $\mathcal{C}$ will have the

same value when guards g_a or g_b are true. As in case 1, it follows that we can assure that by choosing $\mathcal{B}'_a$ and $\mathcal{B}'_b$ appropriately, each of $\gamma_t(f^{\sigma|1}(\vec{x}), f^{\sigma|2}(\vec{x}), ..., f^{\sigma|t-1}(\vec{x}))$, $t \in [1..T]$, as well as $f^{\sigma|t}(\vec{x})$ is only a function of $\mathcal{B}''$, and hence have the same values when guards g_a or g_b are true. However, since f has different linear combinations of $\mathcal{B}''$ at these two guards, it evaluates to different values. Further values for variables in R_a and R_b can be chosen so that guards g_a and g_b are indeed true. □

References

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, August 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CH06] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols (the case of encryption-based mutual authentication and key-exchange). In *TCC*, 2006. Extended version at `http://eprint.iacr.org/2004/334`.

[DDMR07] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol composition logic (pcl). *Electr. Notes Theor. Comput. Sci.*, 172:311–358, 2007.

[Dij75] Edsger Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18:453–457, Winter 1975.

[JPRZ04] C. Jutla, A. Patthak, A. Rudra, and D. Zuckerman. Testing low-degree polynomials over prime fields. In *FOCS*, 2004.

[KLP68] T. Kasami, S. Lin, and W. W. Peterson. New Generalization of the Reed-Muller Codes Part I: Primitive Codes. *IEEE Transactions on Information Theory*, IT-14(2):189–199, March 1968.

[KR04] T. Kaufman and D. Ron. Testing polynomials over general fields. In *FOCS*, 2004.

[MW04] D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.

A Allowing a Few Constants

Let E be a set of linearly independent (over $\mathbb{F}_2$) elements of a field $\mathbb{F}_q$. Now, we *redefine* pseudo-linear polynomials where each linear term is as before defined over

$\mathbb{F}_2$, but can in addition have an addend from E. The same rule also applies to all the linear terms in the guards. Then, we can prove the following theorem.

Theorem 8 *Let $f_1, f_2, ..., f_k$ be k pseudo-linear functions in n variables X, over a field F_q ($q = 2^m$), such that $q > 2^{n+|E|}$. Collectively, we will refer to these polynomials as F. Let f be another pseudo-linear function in X. Then, if f is a function of F, then f is a pseudo-linear function of F.*

Proof is similar to that of Theorem 5, in that we treat E as formal independent variables, and then in the proof of Theorem 5, we set these formal variables to E where we set the k' basis elements of $p_i^{j(f_s,i)}$ to e_s.

A similar version holds for the iterated composition Theorem 7.

B Example

B.1 Pseudo-linear functions

We will consider some simple examples to get a flavor of the problem. Suppose we are given two input functions f_1 and f_2 defined as follows:

$$f_1(x_1, x_2) = x_1 + x_2$$

$$f_2(x_1, x_2) = \left\{ \begin{array}{ll} 0 & \text{if } x_1 = 0 \text{ or } x_2 = 0 \\ x_1 + x_2 & \text{otherwise} \end{array} \right\} = x_2^{q-1} x_1 + x_1^{q-1} x_2$$

We ask if it is possible to extract just x_1 given $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$. That is, can we express $f(x_1, x_2) = x_1$, in terms of f_1 and f_2 alone? To do so, we construct the following truth table:

	x_1	x_2	$x_1 + x_2$	f_1	f_2	f
Row 1	0	0	0	0	0	0
Row 2	x_1	0	x_1	0	x_1	x_1
Row 3	0	x_2	x_2	0	x_2	0
Row 4	x_1	x_1	0	0	0	x_1
Row 5	x_1	x_2	$x_1 + x_2$	$x_1 + x_2$	$x_1 + x_2$	x_1

In the table above we list all linear combinations of the atoms, in this case just x_1, x_2 and $x_1 + x_2$. Each row corresponds to different combinations of cases where each linear combination can be zero or non-zero. Any non-zero entry under a column means that the particular linear combination is non-zero. Simplifications are performed when some of the linear expressions are zero - e.g. Row 4, where we write x_1 under the column x_2 since $x_1 + x_2 = 0 \Rightarrow x_2 = x_1$. It turns out that any pseudo-linear expression projects to a linear expression in any particular row -

thus each such function can be given by a column of linear expressions, e.g. f_1, f_2 and f above. In this particular table for f_1, f_2 and f we can come up with several evidences that f is not a function of f_1, f_2. Consider Row 4: both f_1 and f_2 are 0, whereas f is x_1. Therefore, in accordance with the structure of the row, if we vary x_1, keeping it non-zero and $x_2 = x_1$, we get two pairs (x', x') and (x'', x'') with $x' \neq x''$ such that $f_1(x', x') = f_1(x'', x'') = 0$ and $f_2(x', x') = f_2(x'', x'') = 0$, but $f(x', x') = x' \neq x'' = f(x'', x'')$. Hence f cannot be a function of f_1, f_2. We can construct a counterexample using Row 5 as well: vary x_1 keeping $x_1 + x_2$ constant and keeping $x_1, x_2, x_1 + x_2$ all non-zero - e.g. in $\mathrm{GF}(2^3)$: $(x'_1, x'_2) = (001, 010)$ and $(x''_1, x''_2) = (101, 110)$. The common evidence in both rows is that f is not a linear combination of f_1, f_2.

However, this is not the only type of evidence. Consider the following $f'(x_1, x_2)$:

$$f'(x_1, x_2) = \left\{ \begin{array}{ll} x_1 & \text{if } x_2 = 0 \\ 0 & \text{otherwise} \end{array} \right\} = (1 + x_2^{q-1})x_1$$

Now the table looks like:

	x_1	x_2	$x_1 + x_2$	f_1	f_2	f'
Row 1	0	0	0	0	0	0
Row 2	x_1	0	x_1	0	x_1	x_1
Row 3	0	x_2	x_2	0	x_2	0
Row 4	x_1	x_1	0	0	0	0
Row 5	x_1	x_2	$x_1 + x_2$	$x_1 + x_2$	$x_1 + x_2$	0

Now in each row, f' is a linear combination of f_1, f_2 (including the 0-combination). However, there is a problem with Rows 2 and 3. The problem surfaces when we try to write f' as a combination of f_1, f_2:

	x_1	x_2	$x_1 + x_2$	f_1	f_2	f'
Row 1	0	0	0	0	0	0
Row 2	x_1	0	x_1	0	$f_2\ (= x_1)$	$f_2\ (= x_1)$
Row 3	0	x_2	x_2	0	$f_2\ (= x_2)$	0
Row 4	x_1	x_1	0	0	0	0
Row 5	x_1	x_2	$x_1 + x_2$	$f_1\ (= x_1 + x_2)$	$f_1\ (= x_1 + x_2)$	0

The following pairs can be seen to be counter-examples in $\mathrm{GF}(2^2)$: $(x'_1, x'_2) = (01, 00), (x''_1, x''_2) = (00, 01)$. For these pairs we have: $f_1(x'_1, x'_2) = 00 = f_1(x''_1, x''_2)$, $f_2(x'_1, x'_2) = 01 = f_2(x''_1, x''_2)$, but $f'(x'_1, x'_2) = 01 \neq 00 = f'(x''_1, x''_2)$. This counter-example has been generated by looking at Rows 2 and 3: one of the technical challenges we solve is to systematically come up with counter-examples when arbitrary number of atoms and functions are involved.

Finally, consider the function f'':

$$f''(x_1, x_2) = \left\{ \begin{array}{ll} x_1 & \text{if } x_2 = 0 \\ x_2 & \text{if } x_1 = 0 \text{ and } x_2 \neq 0 \\ 0 & \text{otherwise} \end{array} \right\} = (1 + x_2^{q-1})x_1 + (1 + x_1^{q-1})x_2$$

Now the table looks like:

	x_1	x_2	$x_1 + x_2$	f_1	f_2	f''
Row 1	0	0	0	0	0	0
Row 2	x_1	0	x_1	0	x_1	x_1
Row 3	0	x_2	x_2	0	x_2	x_2
Row 4	x_1	x_1	0	0	0	0
Row 5	x_1	x_2	$x_1 + x_2$	$x_1 + x_2$	$x_1 + x_2$	0

Writing f'' as a combination of f_1, f_2:

	x_1	x_2	$x_1 + x_2$	f_1	f_2	f''
Row 1	0	0	0	0	0	0
Row 2	x_1	0	x_1	0	f_2 $(= x_1)$	f_2 $(= x_1)$
Row 3	0	x_2	x_2	0	f_2 $(= x_2)$	f_2 $(= x_2)$
Row 4	x_1	x_1	0	0	0	0
Row 5	x_1	x_2	$x_1 + x_2$	f_1 $(= x_1 + x_2)$	f_1 $(= x_1 + x_2)$	0

When we "collapse" the table to just the functions we have:

	f_1	f_2	$f_1 + f_2$	f''
Row 1	0	0	0	0
Row 2	0	f_2	f_2	f_2
Row 3	f_1	f_1	0	0

Now we claim that f'' is a function of f_1 and f_2 alone. In fact this can be verified easily:

$$f'' = f_1 + f_2$$

In this particular case we observe that f'' is pseudo-linear in f_1, f_2. We actually prove the general result that if the target function is a function of the input functions, then it is a pseudo-linear function of the input functions.

B.2 Iterated pseudo-linear functions

Consider the input function $f_1(x_1, y)$ and the target function $f(x_1)$ defined as follows:

$$f_1(x_1, y) = \left\{ \begin{array}{ll} x_1 & \text{if } y = x_1 \\ 0 & \text{otherwise} \end{array} \right\} = (1 + (x_1 + y)^{q-1})x_1$$
$$f(x_1) = x_1$$

It is easy to see that the *iterated compositions* of $f_1(x_1, y)$ is just the single function $\mathbf{0}$, which outputs 0 on any input. However, it is possible to compute $f(x_1)$ by calling $f_1(x_1, y)$ as the following algorithm demonstrates:

Algorithm Simulate_f$^{f_1}()$
```
repeat for all non-zero elements y in F_q
  t ← f_1(x_1, y)
  if (t ?= y)
    return t
  y ← next y
end repeat block
return 0
```

In this example, we observe that the complexity of the algorithm is $O(q)$.

Now consider the following input and target functions:

$$f'_1(x_1, y) = \left\{ \begin{array}{ll} 0 & \text{if } y = 0 \text{ or } y = x_1 \\ x_1 & \text{otherwise} \end{array} \right\} = y^{q-1}(x_1 + y)^{q-1}x_1$$
$$f'(x_1) = x_1$$

Here also the *iterated compositions* of $f'_1(x_1, y)$ is just the single function $\mathbf{0}$. Also, it is possible to compute $f(x_1)$ (with high probability) by calling $f'_1(x_1, y)$ as the following algorithm demonstrates:

Algorithm Simulate_f'$^{f'_1}()$
```
choose y randomly from F_q
t ← f_1(x_1, y)
return t
```

In this example, we observe that the complexity of the algorithm is $O(1)$, but it works with probability $1 - O(1/q)$: the probability of y being different from 0 and x_1. For this particular example, it is also possible to come up with an efficient deterministic algorithm - but systematically coming up with efficient deterministic algorithms in all cases where it's possible, seems to be a hard problem. We do show how to systematically come up with randomized efficient algorithms in all the cases where it is possible to do so.

C Proof automation in the UC model

Formally, a proof of security in the UC model boils down to the following: as input, we are given a set of parties and two sets of algorithms:

1. Ideal Functionality: Set of algorithms $F = \{F_1, F_2, \cdots\}$
2. Real Protocol: Set of algorithms $P = \{\Pi_1, \Pi_2, \cdots\}$.

We say that P realizes F if it is possible to construct an algorithm S, called a simulator, that invokes the functions in F, such that the following holds:
For any sequence of calls to algorithms in P, S can come up with a sequence of calls such that the "effect" is "same".

The reader can think of "effect" and "same" as feasibly observable properties. An example "effect" can be an output quantity at the end and "same" could be identical value or close enough probability distribution. The standard model of protocol execution, captured in [Can01], consists of a set of distributed algorithms representing the parties running the protocol, plus an algorithm representing the adversary. The adversary controls a subset of the parties, which in general may be chosen adaptively throughout the execution. In addition, the adversary has some control over the scheduling of message delivery. The parties and adversary interact on a given set of inputs and each party eventually generates local output. The concatenation of the local outputs of the adversary and all parties is called the global output. In the ideal process for evaluating some function f all parties ideally hand their inputs to an incorruptible trusted party, who computes the function values and hands them to the parties as specified. Here the adversary is limited to interacting with the trusted party in the name of the corrupted parties or in a pre-specified manner, defined in the ideal functionalities. Protocol P securely evaluates a function f if for any adversary A (that interacts with the protocol) there exists an ideal-process adversary S such that, for any set of inputs to the parties, the global output of running P with A is indistinguishable from the global output of the ideal process for f with adversary S.

In this section we describe one important language for which we are able to develop a decision procedure, in particular one whose run-time is independent of size of the data types. The ideal functionality is allowed to have just one subroutine F_{in} for inputs from adversary and one for outputs to the adversary F_{out}; rest of the subroutines can receive/send with the adversary, generate new random numbers and perform xor operations. There are two kinds of variables allowed: *ephemeral* variables which are not retained outside current subroutine and *persistent* variables which are retained. The real protocol is a monolithic routine with inputs at the beginning, send / receive / assignments / random number generation / guarded xor operations after that and outputs at the end. While fairly constrained, this language still allows us to model non-trivial cryptographic protocols like secure message transmission using one-time pad, and more complicated functionalities with protocols defined in hybrid models [Can01].

Definition 9 (Nested Guarded Expressions) *Nested Guarded Expressions are*

(guarded xor)	AE	::=	$x_1 \mid x_2 \mid \cdots$	atomic variables
	XE	::=	$AE \mid AE \oplus XE$	xor expression
	BE	::=	$(XE == XE) \mid BE \wedge BE \mid \neg BE$	boolean expression
	CE	::=	`if` BE `then` XE	conditional expression
	GE	::=	$CE \mid CE \oplus GE$	guarded expression
(actions)	a	::=	`send` x	send a term x
			$x :=$ `rcv`	receive term into variable x
			$\langle x_1, x_2, \cdots \rangle :=$ `in`	inputs from the environment
			`out` $\langle x_1, x_2, \cdots \rangle$	outputs to the environment
			$x :=$ `gen`	generate random number
			$x := GE$	guarded expression
(program)	P	::=	a;	single action
			Pa;	sequence of actions
(ideal functionality)	$\mathcal{F}$	::=	$P_{in}, P_{out}, \{P_1, P_2, \cdots\}$	input, output and other subroutines
(real protocol)	Π	::=	P	one program

Table 1: Language definition for the system in consideration.

inductively defined as follows:

- *Guarded Expressions over atoms are Nested Guarded Expressions*
- *Guarded Expressions over Nested Guarded Expressions are Nested Guarded Expressions.*

Theorem 10 *Nested Guarded Expressions are pseudo-linear polynomials in just the atoms.*

Proof: In the base case, the guards are derived according to the following rules ($[P]$ denotes the field polynomial corresponding to expression P) :

$$
\begin{aligned}
&[x] && = x, \text{ for atom } x \\
&[XE_1 \oplus XE_2] && = [XE_1] + [XE_2] \\
&[XE_1 == XE_2] && = [XE_1 + XE_2]^{q-1} \\
&[BE_1 \wedge BE_2] && = [BE_1][BE_2] \\
&[\neg BE] && = 1 + [BE] \\
&[\texttt{if } (BE) \texttt{ then } XE] && = [BE][XE] \\
&[CE_1 \oplus CE_2] && = [CE_1] + [CE_2]
\end{aligned}
$$

It is easy to see that expressions of type GE constructed as above are pseudo-linear in the atoms. For the inductive case, xor-ing two pseudo-linear expressions again is a pseudo-linear expression. The only non-trivial case is the construction

of conditional expressions from pseudo-linear expressions. We have to prove the following: *Any pseudo-linear polynomial raised to the power $q-1$ is a sum of* guard *expressions.* Given this the induction is straightforward.

To prove this recall that PLs can be expressed as sum of EPSELIN terms $\prod_{l\in\mathcal{L}_{/J}} l(\vec{x})^{q-1}.$ $\prod_{l\in J}(1+l(\vec{x})^{q-1}) \cdot p(\vec{x})$. Observe that the product of any two distinct EPSELIN guards $\prod_{l\in\mathcal{L}_{/J_1}} l(\vec{x})^{q-1}\cdot\prod_{l\in J_1}(1+l(\vec{x})^{q-1})$ and $\prod_{l\in\mathcal{L}_{/J_2}} l(\vec{x})^{q-1}\cdot\prod_{l\in J_2}(1+l(\vec{x})^{q-1})$ is 0.

Therefore, we can write down any pseudo-linear polynomial as:

$$GE = (EPS_1 + EPS_2 + \cdots) = (G_1.L_1 + G_2.L_2 + \cdots),$$

where the EPS_i's are EPSELIN terms, the G_i's are guards and the L_i's are the corresponding linear expressions (after gathering all the linear terms with the same G_i together). Now, for any substitution of the atoms $\vec{x}$, at most one of the G_i's is equal to 1 and the rest of the G_i's are 0. This lets us write:

$$(G_1.L_1 + G_2.L_2 + \cdots)^{q-1} = \lambda\vec{x}.\begin{cases} 0 & \text{if all the } G_i(\vec{x})\text{'s are 0} \\ L_1(\vec{x})^{q-1} & \text{if } G_1(\vec{x}) = 1 \\ L_2(\vec{x})^{q-1} & \text{if } G_2(\vec{x}) = 1 \\ \cdots & \end{cases}$$

Hence this is exactly equal to the polynomial $(G_1.L_1^{q-1} + G_2.L_2^{q-1} + \cdots)$ which is a sum of guard expressions in the atoms. □

The central implication of Theorem 10 is that the protocol and the functionalities compute pseudo-linear polynomials in the atoms, which are the environment inputs and the messages received from the adversary as well as the generated random numbers. Thus the **semantics** of the language are functions computing *pseudo-linear* polynomials in the atoms. The completeness theorem (Theorem 7) gives us the intuition (not still a proof - since random numbers and persistent states are involved) that if the protocol is simulatable using the functionalities, then it is sufficient to consider only pseudo-linear combinations of the functionality outputs and terms sent by the adversary, with slight variations due to the additional intricacies. If no such combination exists, there is no simulation possible. The decision procedure and rigorous proofs are part of our forthcoming work.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de