Witness Encryption from Instance Independent Assumptions

Craig Gentry[1], Allison Lewko[2], and Brent Waters[3] *

[1] IBM Research, T.J. Watson
cbgentry@us.ibm.com
[2] Columbia University
alewko@cs.columbia.edu
[3] 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 [14], is a primitive that allows one to encrypt to an instance of an **NP** language L.

* Supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

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. Concepts related to witness encryption include: (in the computational setting) point-filter functions [16], and (in the statistical setting for languages in SZK) non-interactive instance-dependent commitments [23], including efficiently-extractable ones [15].

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 [22, 3, 8], attribute-based encryption [21] (ABE) for arbitrary circuits, and ABE for Turing Machines [17]. The work of [17] 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 [14], they gave a witness encryption construction for the **NP**-complete Exact Cover problem [19] using multilinear encodings (first suggested in [5] and first constructed by Garg, Gentry, and Halevi [11], 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 and are almost identical to the structure of the ciphertexts from the construction.

The Importance and Difficulty of Using Simple 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, [12] explicitly reduces to a instance-dependent family of assumption, while [20] implicitly does this through a meta-assumption.

Our goal is to create techniques for building witness encryption systems that are provably secure under radically simpler assumptions. To achieve this, we

must first confront an intuitive barrier that is formalized as an impossibility result in [14] (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 [14] 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 $u \in [0, N-1]$ [4] 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

[4] 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.

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 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. [5]

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

[5] We note that complexity leveraging is used elsewhere in "computing on encrypted data". For example, current solutions of Attribute-Based Encryption for circuits [13, 18] are naturally selectively secure and require complexity leveraging to achieve adaptive security.

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 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 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 (which can be found in the full version) utilizes asymmetric groups to reduce the order of the group to the product of ℓ primes. Also in the full version of this paper, 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 fact, in the full version we explain how to use only the multilinear subgroup elimination assumption. We also justify the prime order variants of our assumptions in the multilinear generic group model and 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 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*

$$Decrypt_{\mathrm{PWE}}\big(Encrypt_{\mathrm{PWE}}(1^\lambda, x, t, m), w\big) = m.$$

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

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

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

Definition 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

$$\mathsf{PosPWE\,Adv}_{\mathcal{A},x,m,t}(\lambda) =$$

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

Definition 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 the appendix of the full version. The definition of our appendix follows the original of Garg, Gentry, Sahai, and Waters [14], 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 $\mathrm{negl}(\cdot)$ such that for any $x \notin L$ the attacker must only be able to distinguish encryption with probability at most $\mathrm{negl}(\lambda)$. 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.

Building Witness Encryption from Positional Witness Encryption Building standard witness encryption from Positional WE is straightforward. The proofs follows from the hybrid outlined in the introduction. We describe this formally in the full version of this paper.

3 Tribes Schemes

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.

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

Eval$(T, x \in \{0,1\}^n) \to \{0,1\}$ 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

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

3.2 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[6], 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.

[6] 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.

Definition 4. *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.*

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)[7]. 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 5. *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.*

In the full version, we define a tribes-lite scheme as a relaxed notion of a tribes scheme, where only inter-column security is required. We then demonstrate how to be build a tribes scheme from a tribes-lite scheme.

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.

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.

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

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: **(A)** If X_i appears in ϕ_j, we set $M^\phi_{i,j,1} = U$. **(B)** If $\overline{X_i}$ appears in ϕ_j, we set $M^\phi_{i,j,0} = U$. **(C)** 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 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:

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

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

$$\text{For } i > j, \quad M^t_{i,j,0} = B = M^t_{i,j,1}.$$

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 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, $f_{M^t}(y) = 1$.

Lemma 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, $f_{M^t}(y) = 0$.

This defines an effective encoding of positions t from 0 to $2^n - 1$ (considering t as an integer). We also require an encoding of 2^n. 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 $\{0, 1\}$.

$\mathit{Encrypt}_{\mathrm{PWE}}(1^\lambda, \phi, t, m)$ 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 $\mathit{Create}(\lambda, M)$ to produce a tribes scheme T, and sets $\mathrm{CT} := T$.

$\mathit{Decrypt}_{\mathrm{PWE}}(\mathrm{CT}, w)$ The decryptor runs $\mathit{Eval}(\mathrm{CT}, w)$ 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. Our proof can be found in the full version.

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

We provide a description of our first instance of a tribes schemes by instantiating it in symmetric composite order multilinear groups. Its proof of security and intuition on the assumptions appear in the full version.

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, ..., 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:

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

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.*

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 $N = a_1 \ldots a_n b_1 \ldots b_\nu c$, where $a_1, \ldots, a_n, b_1, \ldots, b_\nu, c$ are $n + \nu + 1$ distinct primes. As in Assumption 1, 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*. 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.*

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.

In the full version, we show that inter-column security for this tribes scheme is implied by the multilinear subgroup elimination assumption, and intra-column security is implied by the multilinear subgroup decision assumption. One can alternatively rely solely on the multlinear subgroup elimination assumption to build a tribes-lite scheme first and then derive a tribes scheme from it.

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. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.
12. 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.
13. 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/`.
14. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.
15. Sanjam Garg, Rafail Ostrovsky, Ivan Visconti, and Akshay Wadia. Resettable statistical zero knowledge. In *TCC*, pages 494–511, 2012.
16. Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 553–562. IEEE, 2005.
17. 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.
18. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits. In *STOC*, 2013.
19. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.
20. 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/`.
21. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
22. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
23. Martin Tompa and Heather Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *FOCS*, pages 472–482, 1987.