Symbolic Proofs for Lattice-Based Cryptography

Gilles Barthe[*] Xiong Fan[†] Joshua Gancher[†] Benjamin Grégoire[‡]
Charlie Jacomme[§] Elaine Shi[†]

June 14, 2019

Abstract

Symbolic methods have been used extensively for proving security of cryptographic protocols in the Dolev-Yao model, and more recently for proving security of cryptographic primitives and constructions in the computational model. However, existing methods for proving security of cryptographic constructions in the computational model often require significant expertise and interaction, or are fairly limited in scope and expressivity.

This paper introduces a symbolic approach for proving security of cryptographic constructions based on the Learning With Errors assumption (Regev, STOC 2005). Such constructions are instances of lattice-based cryptography and are extremely important due to their potential role in post-quantum cryptography. Following (Barthe, Grégoire and Schmidt, CCS 2015), our approach combines a computational logic and deducibility problems—a standard tool for representing the adversary's knowledge, the Dolev-Yao model. The computational logic is used to capture (indistinguishability-based) security notions and drive the security proofs whereas deducibility problems are used as side-conditions to control that rules of the logic are applied correctly. We then use AutoLWE, an implementation of the logic, to deliver very short or even automatic proofs of several emblematic constructions, including CPA-PKE (Gentry et al., STOC 2008), (Hierarchical) Identity-Based Encryption (Agrawal et al. Eurocrypt 2010), Inner Product Encryption (Agrawal et al. Asiacrypt 2011), CCA-PKE (Micciancio et al., Eurocrypt 2012). The main technical novelty beyond AutoLWE is a set of (semi-)decision procedures for deducibility problems, using extensions of Gröbner basis computations for subalgebras in the (non-)commutative setting (instead of ideals in the commutative setting). Our procedures cover the theory of matrices, which is required for lattice-based assumption, as well as the theory of non-commutative rings, fields, and Diffie-Hellman exponentiation, in its standard, bilinear and multilinear forms. Additionally, AutoLWE supports oracle-relative assumptions, which are used specifically to apply (advanced forms of) the Leftover Hash Lemma, an information-theoretical tool widely used in lattice-based proofs.

1 Introduction

Formal methods, and in particular formal verification, have long been used for building and checking mathematical claims of correctness or security for small but possibly very complex to moderately large and complex systems. In contrast to pen-and-paper counterparts, formally verified claims deliver higher assurance and independently verifiable proofs that can be replayed by third parties. Over the last 20 years, formal methods have been applied successfully to analyze the security of cryptographic protocols in the Dolev-Yao model [DY83], an idealized model in which cryptographic constructions are treated algebraically. By abstracting away from the probabilistic nature of cryptographic constructions, the Dolev-Yao model has served as a suitable and practical foundation for highly or fully automated tools [ABB$^+$05, Bla01, SMCB12]. These

[*] IMDEA Software Institute. Email: gilles.barthe@imdea.org.

[†] Cornell University. Email: xfan@cs.cornell.edu, jrg358@cornell.edu, elaine@cs.cornell.edu.

[‡] Inria Sophia-Antipolis, Email: benjamin.gregoire@inria.fr.

[§] LSV & CNRS & ENS Paris-Saclay & Inria & Université Paris-Saclay. Email: charlie.jacomme@lsv.fr.

tools have subsequently been used for analyzing numerous cryptographic protocols, including recently TLS 1.3. [CHH$^+$17, KBB17]. Unfortunately, the Dolev-Yao model is focused on cryptographic protocols and cannot be used for reasoning about cryptographic primitives. A related approach is to use so-called refinement types (a.k.a. logical assertions) for reasoning about implementations written in a functional programming language [SCF$^+$13]; this approach has also been used for analyzing TLS 1.3. [BFK$^+$13, DFK$^+$17], but is also primarily limited to cryptographic protocols.

An alternative approach is to develop symbolic methods that reason directly in the computational model. This approach applies both to primitives and protocols, and instances of this approach have been instrumented in tools such as CertiCrypt [BGB09], CryptHOL [Loc16] CryptoVerif [Bla06], EasyCrypt [BGHZ11, BDG$^+$13], and FCF [PM15] (see also [IK03, BDKL10] for further approaches not supported by tools). However, these tools require significant user interaction and expertise, in particular when used for reasoning about cryptographic primitives.

A promising approach for analyzing cryptographic primitives in the computational model is to combine computational logics and symbolic tools from the Dolev-Yao model. Prior work has demonstrated that this approach works well for padding-based (combining one-way trapdoor permutations and random oracles) [BCG$^+$13] and pairing-based cryptography [BGS15]. Broadly speaking, computational logics formalize game-playing security proofs; each step of the proof corresponds to a hop, and symbolic side-conditions are used to ensure the validity of the hop. More specifically, computational logics, which can be seen as specializations of [BDKL10], are used to capture computational security goals and to drive security proofs whereas side-conditions use symbolic tools such as deducibility and static equivalence to guarantee that the rules of the logic are applied correctly. In particular, a key idea of this approach is to use deducibility for controlling the application of rules for performing reductions to hardness assumptions, and for performing optimistic sampling, a particularly common and useful transformation which simplifies probabilistic experiments by allowing to replace, under suitable circumstances, sub-computations by uniform samplings.

The use of deducibility in side conditions, as opposed to arbitrary mathematical conditions, is a necessary step for automating application of proof rules, and more generally for automating complete proofs. However, the interest of this approach is conditioned by the ability to check the validity of deducibility problems. The problem of deciding deducibility has been studied extensively in the context of symbolic verification in the Dolev-Yao model, where deducibility formalizes the adversary knowledge [Low96, MS01, Pau00, KMM94, Sch96, RKTC03, CLS03, RT03]. This line of work has culminated in the design and implementations of decision procedures for classes of theories that either have some kind of normal form or satisfy a finite variant property. However, existing decidability results are primarily targeted towards algebraic theories that arise in the study of cryptographic protocols. In contrast, deducibility problems for cryptographic constructions require to reason about mathematical theories that may not have a natural notion of normal form or satisfy the finite variant property.

Thus, a main challenge for computational logics based on deducibility problems is to provide precise and automated methods for checking the latter. There are two possible approaches to address this challenge:

- heuristics: rather than deciding deducibility, one considers weaker conditions that are easier for verification. As demonstrated with AutoG&P, such an approach may work reasonably well in practice. However, it is not fully satisfactory. First, the heuristics may be incomplete and fail to validate correct instances. Second, advanced proof rules that perform multiple steps at once, and proof search procedures, which explores the space of valid derivations, become unpredictable, even for expert users.

- (semi-)decision procedures based on computational mathematics: in this approach, one provides reductions from deducibility problems to computational problems in the underlying mathematical setting. Then, one can reuse (semi-)decision procedures for the computational problems to verify deducibility problems. This approach offers some important advantages. First, it eliminates a potential source of incompleteness, and in particular the possibility that a proof step fails. Second, it is more predictable.

> Predictability is very important when a high level of automation is sought. Indeed, automation is often achieved through advanced tactics. When they involve multiple heuristics, the outcome of advanced tactics cannot be anticipated, which is a major hurdle to the adoption of formal verification tools. Third, it formalizes connections between known mathematical problems, which may have been extensively studied, and verification problems that may arise for the first time. Lastly, it encourages reusing existing algorithms and implementations.

The idea using methods from computational mathematics to reason about deducibility is natural. However, we are not aware of prior work that exploits this connection in relation with the use of deducibility in a computational logic.

1.1 Contributions

We propose symbolic methods for proving security of lattice-based cryptographic constructions. These constructions constitute a prime target for formal verification, due to their potential applications in post-quantum cryptography and their importance in the ongoing NIST effort to standardize post-quantum constructions; see e.g. [Pei16] for a recent survey of the field.

In this paper, we define a logic for proving computational security of lattice-based cryptographic constructions. The logic follows the idea of combining computational proof rules with symbolic side-conditions, as in [BCG$^+$13, BGS15]. One important feature of our logic is that the proof rule for assumptions supports information-theoretic and computational assumptions that are stated using adversaries with oracle accesses. This extension is critical to capture (advanced cases of) the Leftover Hash Lemma [ILL89]. The Leftover Hash Lemma is a powerful information-theoretical tool which allows to replace, under suitable conditions, a subcomputation by a sampling from a uniform distribution. The Leftover Hash Lemma is widely used in cryptographic proofs, in particular in the setting of lattice-based cryptography. We implement our logic in a tool called AutoLWE (`https://github.com/autolwe/autolwe`), and use the tool for proving (indistinguishability-based) security for several cryptographic constructions based on the Learning with Errors (LWE) assumption [Reg05a].

More specifically, our examples include: dual Regev PKE [GPV08], MP-PKE [MP12], ABB-(H)IBE [ABB10] and IPE [AFV11]. All of our mechanized proofs are realistically efficient, running in at most three seconds (Fig. 12); efficiency in this setting is usually not an issue, since cryptographic constructions typically induce small instances of the deducibility problem. Recent progress on more advanced cryptographic constructions based on lattices, like attribute-based encryption [BGG$^+$14] and predicate encryption [GVW15], are closely related to both the structure of the schemes and strategy in the proofs in [GPV08, ABB10, AFV11]. The MP-PKE [MP12] inspires development in some lattice-based constructions, like homomorphic encryption [AP14] and deniable attribute-based encryption [AFL16].

The technical core of our contributions are a set of (semi-)decision procedures for checking deducibility in the theory of Diffie-Hellman exponentiation, in its standard, bilinear and multilinear versions, and in the theories of fields, non-commutative rings, and matrices. In particular, we give decision procedures for checking deducibility in the theory of Diffie-Hellman exponentiation. This procedure has immediate applications to reasoning about security of cryptographic constructions based on bilinear and multilinear maps. The central idea behind our algorithm is to transform a deducibility problem into a problem from commutative algebra. The latter can be resolved through standard computations of Gröbner basis. Furthermore, we give a semi-decision procedure for checking deducibility in the theory of matrices. This has immediate applications to reasoning about security of lattice-based constructions. In this case, our algorithm extracts from a deducibility question a problem from non-commutative algebra. The problem can be resolved through semi-decision procedures based on non-commutative variants of Gröbner bases known as Subalgebra Analog of Gröbner Basis on Ideals (SAGBI) [Nor98].

2 Example: Dual Regev Encryption

In this section, we describe an example public-key encryption scheme and show how it will be encoded in our formal system. We provide some mathematical background in Section 5.2. Recall that a public-key cryptosystem is given by three probabilistic algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ for generating keys, encryption, and decryption, such that with overwhelming probability, decryption is the inverse of encryption for valid key pairs.

We consider the Dual Regev Encryption scheme [GPV08], an optimization of Regev's original encryption [Reg05b]. We focus on a simple version that encrypts single bits; however, standard techniques can be used to encrypt longer messages.

Definition 2.1 (Dual Regev Encryption). *Below, let $\lambda = n$ be the security parameter, $m = O(n \log q), q = O(m)$ and χ (or χ^n) be discrete Gaussian distribution over $\mathbb{Z}$ (or $\mathbb{Z}^n$).*

- *The key generation algorithm, $\mathsf{KeyGen}(1^\lambda)$, chooses a uniformly sampled random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\boldsymbol{r} \in \{-1, 1\}^m$ sampled uniformly, interpreted as a vector in $\mathbb{Z}_q^m$. The public key is $\mathsf{pk} = (\mathbf{A}, \boldsymbol{u})$, where $\boldsymbol{u} = \mathbf{A}\boldsymbol{r}$, and the secret key is $\mathsf{sk} = \boldsymbol{r}$.*

- *To encrypt a message $b \in \{0, 1\}$, the encryption algorithm $\mathsf{Enc}(\mathsf{pk}, b)$ chooses a random vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, a vector $\boldsymbol{x}_0$ sampled from χ^n and an integer x_1 sampled from χ. The ciphertext consists of the vector $\boldsymbol{c}_0 = \boldsymbol{s}^\mathsf{T}\mathbf{A} + \boldsymbol{x}_0^\mathsf{T}$ and the integer $c_1 = \boldsymbol{s}^\mathsf{T}\boldsymbol{u} + x_1 + b\lceil q/2 \rceil$, where T denotes the transpose operation on matrices.*

- *The decryption algorithm checks whether the value $c_1 - \langle \boldsymbol{r}, \boldsymbol{c}_0 \rangle$ is closer to 0 or $b\lceil q/2 \rceil$ modulo p, and returns 0 in the first case, and 1 in the second.*

Decryption is correct with overwhelming probability, since we compute that $c_1 - \langle \boldsymbol{r}, \boldsymbol{c}_0 \rangle = x_1 + b\lceil q/2 \rceil - \langle \boldsymbol{r}, \boldsymbol{x}_0 \rangle$, so the norm of the term $x_1 - \langle \boldsymbol{r}, \boldsymbol{x}_0 \rangle$ will be much smaller than $b\lceil q/2 \rceil$.

Gentry, Peikert and Vaikuntanathan [GPV08] show that Dual Regev Encryption achieves chosen-plaintext indistinguishability under the *decisional LWE assumption*, defined below. Traditionally, chosen-plaintext indistinguishability is modeled by a probabilistic experiment, where an adversary proposes two messages m_0 and m_1, and is challenged with a ciphertext $c^\star$ corresponding to an encryption of message m_b, where b is sampled uniformly at random. The adversary is then requested to return a bit b'. The winning condition for the experiment is $b = b'$, which models that the adversary guesses the bit b correctly. Formally, one defines the advantage of an adversary $\mathcal{A}$ against chosen-plaintext security as:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{cpa}} = \left| \Pr{}_G[\, b = b' \,] - \frac{1}{2} \right|$$

where G is the probabilistic experiment that models chosen-plaintext security and $\frac{1}{2}$ represents the probability that a trivial adversary which flips a coin b' at random guesses the bit b correctly. We note that in our case, since the message space is $\{0, 1\}$, we can wlog set $m_0 = 0$ and $m_1 = 1$; thus, the adversary only needs to be queried once in this experiment.

The formal definition of G, instantiated to Dual Regev Encryption, is shown in Figure 1. We inline the key generation and encryption subroutines. In line 1, the public key $(\mathbf{A}, \boldsymbol{u})$ and its associated secret key $\boldsymbol{r}$ are randomly sampled. In lines 2 and 3, the message bit b is sampled uniformly, and the ciphertext (c_0, c_1) of this message is generated. Finally, in line 4, the adversary outputs a bit b', given as input the public key and the ciphertext.

Now, we outline the hardness assumptions and lemmas used in the proof of Dual Regev Encryption.

Learning with Errors

The *Learning With Errors* (LWE) assumption [Reg05b] is a computational assumption about the hardness of learning a linear function from noisy samples. We make use of the decisional variant, in which one distinguishes a polynomial number of "noisy" inner products with a secret vector from uniform.

Definition 2.2 (LWE). *Let n, m, q, and χ be as in Definition 2.1. Given $\boldsymbol{s} \in \mathbb{Z}_q^n$, let $\mathsf{LWE}_{\boldsymbol{s},\chi}$ (dubbed* the LWE distribution*) be the probability distribution on $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ obtained by sampling $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ at uniform, sampling $\mathbf{e}$ from χ^n, and returning the pair $(\boldsymbol{A}, \boldsymbol{s}^\mathsf{T}\boldsymbol{A} + \boldsymbol{e})$. The decision-$\mathsf{LWE}_{q,n,m,\chi}$ problem is to distinguish $\mathsf{LWE}_{\boldsymbol{s},\chi}$ from uniform, where $\boldsymbol{s}$ is uniformly sampled.*

We say the decision-$\mathsf{LWE}_{q,n,m,\chi}$ problem is infeasible if for all polynomial-time algorithms $\mathcal{A}$, the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{lwe}}(1^\lambda)$ is negligibly close to $1/2$ as a function of λ:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{lwe}}(1^\lambda) = |\mathsf{Pr}[\mathcal{A} \text{ solves LWE}] - 1/2|$$

The works of [Reg05b, Pei09, BLP$^+$13] show that the LWE assumption is as hard as (quantum or classical) solving GapSVP and SIVP under various settings of n, q, m and χ.

Leftover Hash Lemma

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be a collection of m samples of uniform vectors from $\mathbb{Z}_q^n$. The Leftover Hash Lemma (LHL) states that, given enough samples, the result of multiplying $\mathbf{A}$ with a random $\{-1, 1\}$-valued matrix $\mathbf{R}$ is statistically close to uniform. Additionally, this result holds in the presence of an arbitrary linear leakage of the elements of $\mathbf{R}$. Specifically, the following leftover hash lemma is proved in [ABB10] (Lemma 13).

Lemma 2.1 (Leftover Hash Lemma). *Let q, n, m be as in Definition 2.1. Let k be a polynomial of n. Then, the distributions $\{(\mathbf{A}, \mathbf{AR}, \mathbf{R}^\mathsf{T}\mathbf{w})\}$ $\{(\mathbf{A}, \mathbf{B}, \mathbf{R}^\mathsf{T}\mathbf{w})\}$ are negligibly close in n, where $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$ in both distributions, $\mathbf{R} \overset{\$}{\leftarrow} \{0, 1\}^{m\times k}$, $\mathbf{B} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times k}$, and $\mathbf{w} \in \mathbb{Z}_q^m$ is any arbitrary vector.*

Given the above, security of Dual Regev Encryption is stated as follows:

Proposition 2.2 ([GPV08]). *For any adversary $\mathcal{A}$ against chosen-plaintext security of Dual Regev Encryption, there exists an adversary $\mathcal{B}$ against LWE, such that:*

- $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{cpa}} \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{lwe}} + \epsilon_{LHL}$*;*
- $t_{\mathcal{A}} \approx t_{\mathcal{B}}$*;*

where $\mathbf{Adv}_{\mathcal{B}}^{\mathsf{lwe}}$ denotes the advantage of $\mathcal{B}$ against decisional LWE problem, ϵ_{LHL} is a function of the scheme parameters determined by the Leftover Hash Lemma, and $t_{\mathcal{A}}$ and $t_{\mathcal{B}}$ respectively denote the execution time of $\mathcal{A}$ and $\mathcal{B}$.

Security proof We now outline the proof of Proposition 2.2.

The proof proceeds with a series of *game transformations*, beginning with the game in Figure 1. The goal is to transform the game into one in which the adversary's advantage is obviously zero. Each transformation is justified semantically either by semantic identities or by probabilistic assertions, such as the LWE assumption; in the latter case, the transformation incurs some error probability which must be recorded.

The first transformation performs an information-theoretic step based on the Leftover Hash Lemma. The Leftover Hash Lemma allows us to transform the joint distribution $(\mathbf{A}, \mathbf{A}\boldsymbol{r})$ (where $\mathbf{A}$ and $\boldsymbol{r}$ are independently randomly sampled) into the distribution $(\mathbf{A}, \boldsymbol{u})$ (where $\boldsymbol{u}$ is a fresh, uniformly sampled variable).

```
Game $G^{\mathsf{pke}}_{\mathsf{org}}$ :
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \boldsymbol{r} \xleftarrow{\$} \{-1,1\}^m$;
let $\boldsymbol{u} = \mathbf{A}\boldsymbol{r}$;
$b \xleftarrow{\$} \{0,1\}, \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n, \boldsymbol{x}_0 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m}, x_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}}$;
let $\boldsymbol{c}_0 = \boldsymbol{s}^{\mathsf{T}}\mathbf{A} + \boldsymbol{x}_0, c_1 = \boldsymbol{s}^{\mathsf{T}}\boldsymbol{u} + x_1 + b\lceil q/2\rceil$;
$b' \leftarrow \mathcal{A}(\mathbf{A}, \boldsymbol{u}, \boldsymbol{c}_0, c_1)$;
```

Figure 1: IND-CPA security of dual-Regev PKE.

(This invocation does not use the linear leakage $\boldsymbol{w}$ from Lemma 2.1). In order to apply this lemma, we *factor* the security game from Figure 1 into one which makes use of $\boldsymbol{A}$ and $\boldsymbol{u}$, but not $\boldsymbol{r}$. That is, if G_0 is the original security game, then we have factored G into

$$G_0 = G'\{\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m};\ \boldsymbol{r} \xleftarrow{\$} \{-1,1\}^m;\ \mathsf{let}\ \boldsymbol{u} = \mathbf{A}\boldsymbol{r}\}_p,$$

where $G'\{\cdot\}_p$ is a game context with a hole at position p, such that G' does not make reference to $\boldsymbol{r}$ except in the definition of $\boldsymbol{u}$. By the Leftover Hash Lemma, we may now move to the game:

$$G_1 = G'\{\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m};\ \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n\}_p.$$

This transformation effectively removes $\boldsymbol{r}$ from the security game, thus removing any contribution of the secret key $\boldsymbol{r}$ to the information gained by the adversary $\mathcal{A}$. This transformation incurs the error probability ϵ_{LHL}. The resultant game is shown in Figure 2.

```
Game $G_2$ :
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n$;
$b \xleftarrow{\$} \{0,1\}, \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n, \boldsymbol{x}_0 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m}, x_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}}$;
let $\boldsymbol{c}_0 = \boldsymbol{s}^{\mathsf{T}}\mathbf{A} + \boldsymbol{x}_0, c_1 = \boldsymbol{s}^{\mathsf{T}}\boldsymbol{u} + x_1 + b\lceil q/2\rceil$;
$b' \leftarrow \mathcal{A}(\mathbf{A}, \boldsymbol{u}, \boldsymbol{c}_0, c_1)$;
```

Figure 2: Dual-Regev PKE: Game 2

The second transformation performs a reduction step based on the LWE assumption. Indeed, note that after the first transformation, the ciphertexts $(\boldsymbol{c}_0, c_1)$ contain an LWE distribution of dimension $n \times (m+1)$, with the message bit added to c_1. By applying LWE, we then may safely transform $\boldsymbol{c}_0$ to be uniformly random, and c_1 to be uniformly random added to to the message bit. The resulting security game is shown in Figure 3.

Game G_3 :
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n;$
$b \xleftarrow{\$} \{0,1\}, \boldsymbol{r_0} \xleftarrow{\$} \mathbb{Z}_q^m, r_1 \xleftarrow{\$} \mathbb{Z}_q;$
let $\boldsymbol{c}_0 = \boldsymbol{r_0}, c_1 = r_1 + b\lceil q/2 \rceil;$
$b' \leftarrow \mathcal{A}(\mathbf{A}, \boldsymbol{u}, \boldsymbol{c}_0, c_1);$

Figure 3: Dual-Regev PKE: Game 3

The next transformation applies a semantics-preserving transformation known as *optimistic sampling*. To remove the message bit from the adversary input, note that the term c_1 is equal to the sum of r_1 and $b\lceil q/2 \rceil$, where r_1 is uniformly sampled and does not appear anywhere else in the game. Because of this, we know that c_1 itself is uniformly random. Thus, we can safely rewrite the body of c_1 to be equal to a fresh uniformly sampled r_1. The resulting game is shown in Figure 4.

Game G_4 :
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^n;$
$b \xleftarrow{\$} \{0,1\}, \boldsymbol{r_0} \xleftarrow{\$} \mathbb{Z}_q^m, r_1 \xleftarrow{\$} \mathbb{Z}_q;$
let $\boldsymbol{c}_0 = \boldsymbol{r_0}, c_1 = r_1;$
$b' \leftarrow \mathcal{A}(\mathbf{A}, \boldsymbol{u}, \boldsymbol{c}_0, c_1);$

Figure 4: Dual-Regev PKE: Game 4

In this final game, there is no dependence between the challenge given to the adversary and the challenge b, so the probability that the adversary guesses b is upper bounded by $\frac{1}{2}$.

The most important point about the above proof is that while the cryptographic theory underlying the Leftover Hash Lemma and Learning with Errors assumption is in nature analytic, the proof of security which uses them is only algebraic. That is, no complicated analytic arguments must be made in order to carry out the above proof; instead, each transformation is a straightforward syntactic transformation of the security game.

Our logic is designed to handle game transformations such as the ones in the above proof. Our implemented security proof for Dual Regev Encryption is shown in Figure 5. In lines 1-3, we apply the Leftover Hash Lemma. The `move` tactic is used to reorder samplings in the security game, as long as the two reorderings are semantically equivalent. The `assumption_decisional` tactic is used to apply hardness assumptions and information-theoretic lemmas. Note that all required factorings of games in this proof are performed automatically, handled by our use of the SAGBI method in Section 4.3. This is reflected by the "!" at the end of the tactic, which asks the proof system to automatically factor the game. (More complicated applications of `assumption_decisional` do require the user to provide some hints to the proof system about how to factor the game. These hints are minimal, however.) The arrow `->` after the tactic specifies that we wish to apply the transformation in the forward direction. (It is possible to apply the LHL and the LWE assumption in reverse, as well. This is used in later proofs.) Throughout, we use the `//` tactic to normalize the game. This tactic unfolds let bindings, and applies a syntactic normal form algorithm to all expressions in the game. The `mat_fold` and `mat_unfold` tactics are used to reason about uniformity of matrices of the form $\mathbb{Z}_q^{n \times (m+k)}$: the `mat_unfold` tactic will separate a uniform sampling of type $\mathbb{Z}_q^{n \times (m+k)}$ into two uniform samplings of types $\mathbb{Z}_q^{n \times m}$ and $\mathbb{Z}_q^{n \times k}$ respectively; the `mat_fold` does the corresponding inverse

```
 1  (* apply LHL *)
    move A 1.
 3  assumption_decisional! LHL -> u; //.

 5  (* fold A, u into single matrix Au *)
    mat_fold 1 2 Au; //.
 7
    (* apply LWE assumption *)
 9  move s 2.
    assumption_decisional! LWE -> w; //.
11
    (* unfold LWE distribution *)
13  rnd w (λ w. tr w) (λ w. tr w); //.
    mat_unfold 2 wa wb; //.
15  rnd wb (λ B. tr B) (λ B. tr B); //.

17  (* perform optimistic sampling *)
    move wb 4.
19  rnd wb (λ B. B - (b?Mu(()):0_{1,1}))
           (λ B. B + (b?Mu(()):0_{1,1})); //.
21  indep!.

23  qed.
```

Figure 5: AutoLWE proof for Dual Regev Encryption.

operation.

The `rnd` tactic is used to reason about transformations of uniform samplings: given two functions f, f^{-1} which must be mutual inverses, the `rnd` tactic allows one to "pull" a uniform sampling through f^{-1}. This is used in two ways in the proof: on lines 13 and 15, we use `rnd` to show that instead of sampling a matrix, we may instead sample its transpose. Whenever the original matrix is used, we now take the transpose of the new sampled matrix. Similarly, on line 19 we use `rnd` to perform an optimistic sampling operation, in which `B` is transformed in order to remove the additive factor `b?Mu(()):0_{1,1}`. Here, `Mu` is an uninterpreted function from the unit type to 1 by 1 matrices, modelling the message content $\lceil q/2 \rceil$, and `0_{1,1}` is the constant zero matrix of dimension 1 by 1. The notation `_?_:_` is the standard ternary if-then-else construct; thus, we can model the expression $b\lceil q/2 \rceil$ present in the Dual Regev scheme as the expression `b?Mu(()):0_{1,1}`.

Finally, the `indep!` tactic is used to reason about games such as the game in Figure 4, in which the adversary trivially has no advantage. Detail about the proof rules present in our logic is given in Section 3.4.

3 Logic

Our logic reasons about probabilistic expressions P, built from atomic expressions of the form $\Pr_G[\ \phi\ ]$, where G is a game, and ϕ is an event. Games are probabilistic programs with oracle and adversary calls, and ϕ is the winning condition of the game. The proof rules of the logic formalize common patterns of reasoning from the game-playing approach to security proofs. In their simpler form, proof steps will transform a proof goal $\Pr_G[\ \phi\ ] \le p$ into a proof goal $\Pr_{G'}[\ \phi'\ ] \le p'$, with $p = p' + c$, and G' a game derived from G; alternatively, they will directly discharge the proof goal $\Pr_G[\ \phi\ ] \le p$ (and give a concrete value for p) when the proof goal is of a simple and specific form, e.g. bounding the probability that an adversary guesses a uniformly distributed and secret value.

In order to be able to accommodate lattice-based constructions, the following novelties are necessary:

Dimensions			
d	$::=$	n	dimension variable
	$\mid$	$d_1 + d_2$	addition
	$\mid$	1	constant dimension 1
Types			
t	$::=$	$\mathbb{B}$	boolean value
	$\mid$	$\mathbb{Z}_q$	prime field of order q
	$\mid$	$\mathbb{Z}_q^{d_1 \times d_2}$	integer matrix
	$\mid$	$\mathsf{list}_d\, t$	list
	$\mid$	$t \times \ldots \times t$	tuple
Expressions			
M	$::=$	$\mathbf{0}$	null matrix
	$\mid$	$\mathbf{I}$	identity matrix
	$\mid$	$[M]$	constant list
	$\mid$	$M + M$	addition
	$\mid$	$M \times M$	multiplication
	$\mid$	$-M$	inverse
	$\mid$	$M \parallel M$	concatenation
	$\mid$	$\mathsf{sl}\ M$	left projection
	$\mid$	$\mathsf{sr}\ M$	right projection
	$\mid$	$M^\top$	transpose

Figure 6: Syntax of expressions (selected)

the expression language includes vectors and matrices; new rules for probabilistic samplings and for oracle-relative assumptions (both in the information-theoretic and computational forms). These extensions do not pose any foundational challenge, but must be handled carefully to obtain the best trade-off between generality and automation.

3.1 Games

Games consist of a security experiment in which an adversary with oracle access interacts with a challenger and of an assertion that determines the winning event.

Expressions The expression language operates over booleans, lists, matrices, and integers modulo q, and includes the usual algebraic operations for integer modulo q and standard operators for manipulating lists and matrices. The operations for matrices include addition, multiplication and transposition, together with *structural operations* that capture the functionalities of block matrices, and can be used for (de)composing matrices from smaller matrices. *concatenation*, *split left*, and *split right*. The type of lists, list_d, denotes a list of length d. Lists are manipulated symbolically, so do not support arbitrary destructuring. Lists may be constructed through the *constant list* operation $[\cdot]$, which takes a type τ to the type $\mathsf{list}_d\ \tau$, for any d. All of the matrix operations are lifted pointwise to lists.

The syntax of expressions (restricted to expressions for matrices) is given in Figure 6. Selected typing rules for expressions are given in the Appendix, in Figure 13. Expressions are deterministic, and are interpreted as values over their intended types. Specifically, we first interpret dimensions as (positive) natural

Assertions (event expressions)

ϕ	$::=$	e	expression
	$\mid$	$\exists\, b_1, \ldots, b_k.\, e$	existential queries
	$\mid$	$\forall\, b_1, \ldots, b_k.\, e$	universal queries
where			
b	$::=$	$x \in Q_{\mathsf{o}}$	x ranges over queries

for all queries

Game commands

gc	$::=$	$\mathsf{let}\ x = e$	assignment
	$\mid$	$x \xleftarrow{\$} \mu$	sampling from distr.
	$\mid$	$\mathsf{assert}(\phi)$	assertion
	$\mid$	$y \leftarrow \mathcal{A}(x)\ \mathsf{with}\ \overrightarrow{\mathcal{O}}$	adversary call

Oracle commands

oc	$::=$	$\mathsf{let}\ x = e$	assignment
	$\mid$	$x \xleftarrow{\$} \mu$	sampling from distr.
	$\mid$	$\mathsf{guard}(b)$	guard

Oracle definitions

$\mathcal{O}$	$::=$	$\mathsf{o}(x) = \{\overrightarrow{oc};\ \mathsf{return}\ e\}$

Game definitions

G	$::=$	$\{\overrightarrow{gc};\ \mathsf{return}\ e\}; \overrightarrow{\mathcal{O}}$

where $\mathcal{A}$ and $\mathcal{O}$ range over adversary and oracle names respectively.

Figure 7: Syntax of games

numbers. This fixes the interpretation of types. Expressions are then interpreted in the intended way; for instance, transposition is interpreted as matrix transposition, etc.

Games Games are defined by a sequence of commands (random samplings, assignments, adversary calls) and by an assertion. The command defines the computational behavior of the experiment whereas the assertion defines the winning event. Each adversary call contains a list of oracles that are available to the adversary; oracles are also defined by a sequence of commands (random samplings, assignments, assert statements) and by a return expression. The grammars for oracle definitions and game definitions are given in Figure 7.

The operational behavior of oracles is defined compositionally from the operational behavior of commands:

- random sampling $x \xleftarrow{\$} \mu$: we sample a value from μ and store the result in the variable x;
- assignments: $\mathsf{let}\ x = e$: we evaluate the expression e and store the result in the variable x;
- assertion $\mathsf{guard}(b)$: we evaluate b and return $\perp$ if the result is false. Guards are typically used in decryption oracles to reject invalid queries.

In addition, we assume that every oracle $\mathcal{O}$ comes with a value $\delta_{\mathcal{O}}$ that fixes the maximal number of times that it can be called by an adversary. To enforce this upper bound, the execution is instrumented with a counter $c_{\mathcal{O}}$ that is initially set to 0. Then, whenever the oracle is called, one checks $c_{\mathcal{O}} \geq \delta_{\mathsf{O}}$; if so, then $\perp$ is returned. Otherwise, the counter $c_{\mathcal{O}}$ is increased, and the oracle body is executed. In order to interpret events, we further instrument the semantics of the game to record the sequence of interactions between the adversary and the oracle. Specifically, the semantics of oracles is instrumented with a query set variable $Q_{\mathcal{O}}$ that is initially set to $\emptyset$. Then, for every call the query parameters are stored in $Q_{\mathcal{O}}$. (Following [BDKL10] it would be more precise to hold a single list of queries, rather than a list of queries per oracle, but the latter suffices for our purposes.)

Informally, adversaries are probabilistic computations that must execute within a specific amount of resources and are otherwise arbitrary. One simple way to give a semantics to adversaries is through syntax, i.e. by mapping adversary names to commands, and then interpret these commands using the afore described semantics. However, our language of games is too restrictive; therefore, we map adversary names to commands in a more expressive language, and then resort to the semantics of this richer language. For convenience of meta-theoretic proofs, e.g. soundness, it is preferable to choose a language that admits a set-theoretical semantics. For instance, one can use the probabilistic programming language pWhile to model the behavior of the adversaries.

The semantics of games is defined compositionally from the operational behavior of commands, oracles, and adversaries:

- assertion $\mathsf{assert}(\phi)$: we evaluate ϕ and abort if the result is false.
- adversary call $y \leftarrow \mathcal{A}(e)$ with $\overrightarrow{\mathcal{O}}$: we evaluate e, call the adversary $\mathcal{A}$ with the result as input, and bind the output of the adversary to y. The adversary is provided with access to the oracles $\overrightarrow{\mathcal{O}}$.

Finally, the interpretation of $\Pr_G[\ \phi\ ]$ is to be the probability of ϕ in the sub-distribution obtained by executing G.

Throughout the paper, we assume that the games satisfy the following well-formedness conditions and (without loss of generality) hygiene conditions: (WF1) all variables must be used in scope; (WF2) commands must be well-typed; (Hyg1) adversary and oracle names are distinct; (Hyg2) bound variables are distinct.

3.2 Reasoning about expressions

Our indistinguishability logic makes use of two main relations between expressions: equality and deducibility. Equality is specified through a set of axioms $\mathcal{E}$, from which further equalities can be derived using standard rules of equational reasoning: reflexivity, symmetry, transitivity of equality, functionality of operators, and finally instantiation of axioms. We write $\Gamma \vdash_{\mathcal{E}} e = e'$ if e and e' are provably equal from the axioms $\mathcal{E}$ and the set of equalities Γ. Throughout the paper, we implicitly assume that the set of axioms includes standard identities on matrices.

Deducibility is defined using the notion of contexts. A context C is an expression that only contains a distinguished variable $\bullet$. We write $e \vdash^{C}_{\mathcal{E}} e'$, where e, e' are expressions and C is a context, if $\vdash_{\mathcal{E}} C[e] = e'$. We write $e \vdash_{\mathcal{E}} e'$ if there exists a context C such that $e \vdash^{C}_{\mathcal{E}} e'$. Similarly, we write $\Gamma \models e \vdash^{C}_{\mathcal{E}} e'$ if $\Gamma \vdash_{\mathcal{E}} C[e] = e'$ and $\Gamma \models e \vdash_{\mathcal{E}} e'$ if there exists a context C such that $\Gamma \models e \vdash_{\mathcal{E}} e'$. More generally, a (general) context C is an expression that only contains distinguished variables $\bullet_1, \dots, \bullet_n$. We write $e_1, \dots, e_n \vdash^{C}_{\mathcal{E}} e'$, where $e_1, \dots, e_n, e'$ are expressions and C is a context, if $\vdash_E C[e_1, \dots, e_n] = e'$. We write $e_1, \dots, e_n \vdash_{\mathcal{E}} e'$ if there exists a context C such that $e_1, \dots, e_n \vdash^{C}_{\mathcal{E}} e'$. Similarly, we write $\Gamma \models e_1, \dots, e_n \vdash^{C}_{\mathcal{E}} e'$ if $\Gamma \models C[e_1, \dots, e_n] =_{\mathcal{E}} e'$ and $\Gamma \models e_1, \dots, e_n \vdash_{\mathcal{E}} e'$ if there exists a context C such that $\Gamma \models e_1, \dots, e_n \vdash_{\mathcal{E}} e'$. Intuitively, a context is a recipe that shows how some expression may be computed given other expressions. If we consider matrices, we may have $M + N, O, N \vdash M \times O$ with the context $C(\bullet_1, \bullet_2, \bullet_3) := (\bullet_1 - \bullet_3) \times \bullet 2$.

3.3 Strongest postcondition

A desirable property of any logic is that one can replace equals by equals. In particular, it should always be possible to replace an expression e by an expression e' that is provably equivalent to e. However, it is often desirable to use a stronger substitution property which allows to replace e by an expression e' that is provably equivalent to e relative to the context in which the replacement is to be performed. To achieve this goal, our proof system uses a *strongest postcondition* to gather all facts known at a position p in the main command. The computation of $sp_p(G)$ is done as usual, starting from the initial position of the program with the assertion true and adding at each step the assertion ϕ_c corresponding to the current command c, where:

$$\begin{array}{rcl} \phi_{\mathsf{let}\ x = e} & = & x = e \\ \phi_{\mathsf{guard}(b)} & = & b \\ \phi_{\mathsf{assert}(e)} & = & e \\ \phi_{\forall/\exists\, b_1,\ldots,b_k.\, e} & = & \mathsf{true} \end{array}$$

3.4 Judgment and proof rules

Our computational logic manipulates judgments of the form $P \preceq P'$ where P and P' are probability expressions drawn from the following grammar:

$$P, P' ::= \epsilon \mid c \mid P + P' \mid P - P' \mid c \times P \mid |P| \mid \Pr_G[\ \phi\ ],$$

where ϵ ranges over variables, c ranges over constants, $|P|$ denotes absolute value, and $\Pr_G[\ \phi\ ]$ denotes the success probability of event ϕ in game G. Constants include concrete values, e.g. 0 and $\frac{1}{2}$, as well as values whose interpretation will depend on the parameters of the scheme and the computational power of the adversary, e.g. its execution time or maximal number of oracle calls.

Proof rules are of the form

$$\frac{P_1 \preceq \epsilon_1 \quad \ldots \quad P_k \preceq \epsilon_k}{P \preceq \epsilon}$$

where P_is and P are probability expressions, ϵ_is are variables and finally ϵ is a probability expression built from variables and constants.

Figure 8 present selected rules of the logic. In many cases, rules consider judgments of the form $\Pr_G[\ \phi\ ] \preceq \epsilon$; similar rules exist for judgments of the form $|\Pr_G[\ \phi\ ] - \Pr_{G'}[\ \phi'\ ]| \preceq \epsilon$.

Rules [FALSE] and [CASE] formalize elementary axioms of probability theory. Rules [REFL] and [ADD] formalize elementary facts about real numbers. Rule [EQ] can be used to replace a probability expression by another probability expression that is provably smaller within the theory of reals. For instance, derivations commonly use the identity $\epsilon_1 \leq |\epsilon_1 - \epsilon_2| + \epsilon_2$.

Rules [SWAP], [INSERT], [SUBST] are used for rewriting games in a semantics-preserving way. Concretely, rule [SWAP] swaps successive commands (at position p) that can be reordered (are dataflow independent in the programming language terminology). By chaining applications of the rule, one can achieve more general forms of code motion. Rule [INSERT] inserts at position p command that does not carry any operational behaviour. Rule [SUBST] substitutes at position p an expression e by another expression e' that is contextually equivalent at p, i.e. $sp_p(G) \models e =_{\mathcal{E}} e'$ holds.

The rule [RAND] performs a different transformation known as optimistic sampling. It replaces a uniform sampling from t by $s \xleftarrow{\$} t';\ \mathsf{return}\ C[s]$. To ensure that this transformation is correct, the rule checks

$$\text{[\textsc{False}]}\ \frac{}{\Pr_G[\,\mathsf{false}\,] \preceq 0} \qquad \text{[\textsc{Case}]}\ \frac{\Pr_G[\,\phi \wedge c\,] \preceq \epsilon_1 \quad \Pr_G[\,\phi \wedge \neg c\,] \preceq \epsilon_2}{\Pr_G[\,\phi\,] \preceq \epsilon_1 + \epsilon_2}$$

$$\text{[\textsc{Refl}]}\ \frac{}{\Pr_G[\,\phi\,] \preceq \Pr_G[\,\phi\,]} \qquad \text{[\textsc{Add}]}\ \frac{P \preceq \epsilon_1 \qquad P' \preceq \epsilon_2}{P + P' \preceq \epsilon_1 + \epsilon_2} \qquad \text{[\textsc{Eq}]}\ \frac{P \preceq \epsilon \qquad \vdash P' \le P}{P' \preceq \epsilon}$$

$$\text{[\textsc{Swap}]}\ \frac{\Pr_{G\{c';\,c\}_p}[\,\phi\,] \preceq \epsilon}{\Pr_{G\{c;\,c'\}_p}[\,\phi\,] \preceq \epsilon} \qquad \text{[\textsc{Insert}]}\ \frac{\Pr_{G\{c;\,c'\}_p}[\,\phi\,] \preceq \epsilon}{\Pr_{G\{c'\}_p}[\,\phi\,] \preceq \epsilon}\ \boxed{\begin{array}{l} c \text{ sampling, let,} \\ \text{or } \mathsf{guard}(\mathsf{true}) \end{array}}$$

$$\text{[\textsc{Subst}]}\ \frac{\Pr_{G\{e\}_p}[\,\phi\,] \preceq \epsilon}{\Pr_{G\{e'\}_p}[\,\phi\,] \preceq \epsilon}\ \boxed{sp_p(SE) \models e =_{\mathcal{E}} e'}$$

$$\text{[\textsc{Abstract}]}\ \frac{\left|\Pr_{G_1'}[\,\phi_1\,] - \Pr_{G_2'}[\,\phi_2\,]\right| \preceq \epsilon}{\left|\Pr_{G_1}[\,\phi_1\,] - \Pr_{G_2}[\,\phi_2\,]\right| \preceq \epsilon}\ \boxed{\begin{array}{l} G_1 \equiv G_1'[\mathcal{B}] \\ G_2 \equiv G_2'[\mathcal{B}] \end{array}}$$

$$\text{[\textsc{Rand}]}\ \frac{\Pr_{G\{s \xleftarrow{\$} t';\, \mathsf{let}\ r = C[s]\}_p}[\,\phi\,] \preceq \epsilon}{\Pr_{G\{r \xleftarrow{\$} t\}_p}[\,\phi\,] \preceq \epsilon}\ \boxed{sp_p(G) \models C'[C] =_{\mathcal{E}} \bullet}$$

$$\text{[\textsc{RFold}]}\ \frac{\Pr_{G\{x \xleftarrow{\$} \mathbb{Z}_q^{d_1 \times (d_2 + d_2')};\, \mathsf{let}\ x_1 = \mathsf{sl}\ x;\, \mathsf{let}\ x_2 = \mathsf{sr}\ x\}_p}[\,\phi\,] \preceq \epsilon}{\Pr_{G\{x_1 \xleftarrow{\$} \mathbb{Z}_q^{d_1 \times d_2};\, x_2 \xleftarrow{\$} \mathbb{Z}_q^{d_1 \times d_2'}\}_p}[\,\phi\,] \preceq \epsilon}$$

$$\text{[\textsc{RUnfold}]}\ \frac{\Pr_{G\{x_1 \xleftarrow{\$} \mathbb{Z}_q^{d_1 \times d_2};\, x_2 \xleftarrow{\$} \mathbb{Z}_q^{d_1 \times d_2'};\, \mathsf{let}\ x = x_1 \| x_2\}_p}[\,\phi\,] \preceq \epsilon}{\Pr_{G\{x \xleftarrow{\$} \mathbb{Z}_q^{d_1 \times (d_2 + d_2')}\}_p}[\,\phi\,] \preceq \epsilon}$$

$$\text{[\textsc{Upto}]}\ \frac{\Pr_{G\{\mathsf{guard}(c)\}_p}[\,\phi\,] \preceq \epsilon_1 \qquad \Pr_{G\{\mathsf{guard}(c)\}_p}[\,\exists x \in Q_o.\, c(x) \neq c'(x)\,] \preceq \epsilon_2}{\Pr_{G\{\mathsf{guard}(c')\}_p}[\,\phi\,] \preceq \epsilon_1 + \epsilon_2}\ \boxed{p \text{ first position in } o}$$

$$\text{[\textsc{Guess}]}\ \frac{\Pr_{G;\, x \leftarrow \mathcal{A}()}[\,\phi\,] \preceq \epsilon}{\Pr_G[\,\exists x \in Q_o.\, \phi\,] \preceq \epsilon}$$

$$\text{[\textsc{Find}]}\ \frac{\Pr_{G;\, x \leftarrow \mathcal{A}(e)}[\,\phi_1 \wedge \phi_2\,] \preceq \epsilon}{\Pr_G[\,(\exists x \in Q_o.\, \phi_1) \wedge \phi_2\,] \preceq \epsilon}\ \boxed{\begin{array}{l} C \text{ efficient and} \\ sp_{|G|}(G) \models C[(e, x)] =_{\mathcal{E}} \phi_1 \end{array}}$$

Figure 8: Selected proof rules

that C is provably bijective at the program point where the transformation arises, using a candidate inverse context C' provided by the user. Rules [RFOLD] and [RUNFOLD] are dual and are used to manipulate random samplings of matrices. The rule [RFOLD] is used to turn two uniform samplings of matrices into one uniform sampling of the concatenation; conversely, the rule [RUNFOLD] may be used to turn one uniform sampling of a concatenation into uniform samplings of its component parts. (We also have similar rules [LFOLD] and [LUNFOLD] in order to manipulate the vertical component of the dimension.) These rules are primarily used to apply axioms which are stated about matrices of compound dimension.

The rule [ABSTRACT] is used for applying computational assumptions. The rule can be used to instantiate a valid judgment with a concrete adversary. The side-conditions ensure that the experiments G_1 and G_2 are syntactically equivalent to the experiment $G'_1[\mathcal{B} := B]$ and $G'_2[\mathcal{B} := B]$, where the notation $G'[\mathcal{B} := B]$ represents the game obtained by inlining the code of $\mathcal{B}$ in G'. Because of the requirement on syntactic equivalence, it is sometimes necessary to apply multiple program transformations before applying an assumption.

The rule [UPTO] rule is used for replacing $\mathsf{guard}(c')$ at position p in an oracle with $\mathsf{guard}(c)$. According to the usual principle for reasoning up to failure events, the rule yields two proof obligations: bound the probability of the original event and the probability that the adversary performs a query where the results of c and c' differ.

The rules [GUESS] and [FIND] rules are used to deal with winning events involving existential quantification.

The logic also contains a rule for hybrid arguments. The rule is similar to [BGS15] and ommitted For lack of space.

3.5 Soundness

All proof rules of the logic are sound. To state soundness, we lift the interpretation of games to an interpretation of judgments and derivations. This is done by first defining a fixed interpretation of dimensions that is used for all the games of the derivation. Then, we define the interpretation of P inductively. We say that judgment $P \preceq P'$ is *valid* iff the inequality holds for every valid interpretation of P and P'. Finally, one can prove that $P \preceq P'$ is valid whenever $P \preceq P'$ is derivable in the logic.

3.6 Axioms Used

Here, we describe the axioms used to prove the schemes in Sections 2 and 5 secure. Each axiom is *decisional*, in that it is a claim about the closeness of two games. This is modeled by having both games end with a bit output *b*, so that each axiom is a claim of the form $|\Pr_{G_0}[\, b \,] - \Pr_{G_1}[\, b \,]| \preceq \epsilon$. This allows us to apply the [ABSTRACT] rule from Figure 8.

3.6.1 Learning with Errors

Recall from Section 2 that the LWE assumption states that the distribution $(\boldsymbol{A}, \boldsymbol{s}^{\mathsf{T}}\boldsymbol{A}+\boldsymbol{e})$ is indistinguishable from uniform, where $\boldsymbol{A}$ and $\boldsymbol{s}$ are uniformly sampled elements of $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^n$ respectively, and $\boldsymbol{e}$ is sampled from some given error distribution.

Our concrete encoding is given in Figure 9. Since our logic only deals with uniform samplings, in order to encode more complicated sampling algorithms such as the error distribution for LWE, we separate the sampling algorithm into a *coin sampling* stage and a *deterministic* stage. In the coin sampling stage, an element of $\{0, 1\}^c$ is sampled, where c is the number of coins the sampling algorithm will use. (Since the sampling algorithm is polynomial time, c will be a polynomial of the security parameter.) In the deterministic stage, we call an uninterpreted function (here, Chi) which uses the sampled coins to produce the output of the distribution.

In various applications of the LWE assumption, the parameter settings of Figure 9 will alter slightly – for instance, in the Dual Regev scheme from Section 2, we do not use m on the nose, but rather $m + 1$. This difference is immaterial to the validity of the assumption.

Game G_0^{LWE} :
$\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$; $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^{n}$;
$c_e \xleftarrow{\$} \{0,1\}^{c_{\mathrm{Chi}}}$; let $\boldsymbol{e} = \mathrm{Chi}(c_e)$;
$b \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{s}^T\boldsymbol{A} + \boldsymbol{e})$;

Game G_1^{LWE} :
$\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$; $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^{m}$;
$b \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{u})$;

Figure 9: The LWE assumption, encoded in AutoLWE.

3.6.2 Leftover Hash Lemma

The most subtle part of our proofs is often not applying the LWE assumption, but rather applying the Leftover Hash Lemma. This is because the LHL is an *information-theoretic* judgment rather than a computational one; information-theoretic judgments enjoy stronger composition properties than computational judgments.

Recall that the (basic) LHL states that the distribution $(\boldsymbol{A}, \boldsymbol{AR}, \boldsymbol{wR})$ is statistically close to the distribution $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{wR})$, where $\boldsymbol{A}$ is a uniformly random element of $\mathbb{Z}_q^{n\times m}$, $\boldsymbol{R}$ is a uniformly random element of $\{-1, 1\}^{m\times k}$ (interpreted as a matrix), and $\boldsymbol{w}$ is a fixed arbitrary vector in $\mathbb{Z}_q^m$. For the LHL to hold, however, we can actually relax the requirements on $\boldsymbol{A}$: instead of $\boldsymbol{A}$ being sampled uniformly, we only require that $\boldsymbol{A}$ is sampled from a distribution which is *statistically close* to uniform.

In the literature, it is often the case that the lemma being applied is not the LHL on the nose, but rather this weakened (but still valid) form in which $\boldsymbol{A}$ only need to be close to uniform. In many of our proofs, this occurs because $\boldsymbol{A}$ is not uniformly sampled, but rather sampled using an algorithm, TrapGen, which produces a vector $\boldsymbol{A}$ statistically close to uniform along with a *trapdoor* T_A, which is kepts secret from the adversary.

By combining the LHL with the TrapGen construction, we obtain the security games in Figure 10. Both games are displayed at once: the expressions which vary between the two games are annotated with which game they belong in. In order to model how $\boldsymbol{R}$ is sampled, we sample the component bits of $\boldsymbol{R}$ from $\{0, 1\}^{d_{LHL}}$, and apply a symbolic function, bitinj, which converts these component bits into a matrix. Note in this security game that $\mathbf{w}$ comes from a symbolic adversary, $\mathcal{A}_1$. This models the universal quantification of $\mathbf{w}$ in the LHL. Additionally, note that $\mathcal{A}_2$ actually receives the trapdoor $T_{\mathbf{A}}$. This is counterintuitive, because adversaries in the cryptosystems do not have access to the trapdoor. However, remember that here we are constructing the adversary *for the LHL*; giving $\mathcal{A}_2$ the trapdoor reflects the assertion that the distribution $(\mathbf{A}, \mathbf{AR}, \mathbf{wR}, T_{\mathbf{A}})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B}, \mathbf{wR}, T_{\mathbf{A}})$, which follows from the information theoretic nature of the LHL.

While we use the assumption from Figure 10 in our proofs, we also use several small variations which are also valid. One such variation is in the proof of Dual Regev, where we do not use the TrapGen algorithm, but rather sample $\boldsymbol{A}$ uniformly (and do not give the adversary $T_{\boldsymbol{A}}$); additionally, we do not include this linear leakage $\boldsymbol{w}$. Another such variation is used in our CCA proof from Section 5. In this instance, we do not transform $\boldsymbol{AR}$ to $\boldsymbol{B}$, but rather to $\boldsymbol{AR}+\boldsymbol{B}$ (thus generalizing our [RAND] rule.) Additionally, we must state the LHL in the CCA proof to be relative to the decryption oracle, which makes use of $\boldsymbol{R}$. This relativized lemma is still valid, however, since the decryption oracle does not leak any information about $\boldsymbol{R}$. It will be interesting future work in order to unify these small variations of the LHL.

```
Game $G_\beta^{LHL}$ :
  $c \xleftarrow{\$} \{0,1\}^{d_{TG}}$; let $(\mathbf{A}, T_{\mathbf{A}}) = \text{TrapGen}(c)$;
  $r \xleftarrow{\$} \{0,1\}^{d_{LHL}}$; let $R = \text{bitinj}(r)$;
  $\overset{\text{if }\beta=1}{\overline{\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m};}}$ $\mathbf{w} \leftarrow \mathcal{A}_1()$;
  $b \leftarrow \mathcal{A}_2(\mathbf{A}, \overset{\text{if }\beta=0}{\overline{\mathbf{AR}}} \quad \overset{\text{if }\beta=1}{\overline{\mathbf{B}}}\ , \mathbf{wR}, T_{\mathbf{A}}, \mathbf{w})$;
```

Figure 10: The LHL assumption combined with TrapGen, encoded in AutoLWE.

3.6.3 Distribution Equivalences

In addition to the two main axioms above, we also rely on several opaque probabilistic judgments about distributions from which the adversary may sample, but are written in terms of private variables which the adversary may not access. For instance, in an Identity-Based Encryption scheme, the adversary could have access to a KeyGen oracle, which must use the master secret key in order to operate. This is the case in Section 5.2. In the concrete proof, there is a step in which we change the implementation of the KeyGen oracle from one uninterpreted function to another. Transformations of this sort are encoded using oracle-relative assumptions, which are generalizations of axioms in AutoG&P which allow adversaries to query oracles.

For example, in Figure 11, we state closeness of the distributions $D_0(\boldsymbol{s_0}, \cdot)$ and $D_1(\boldsymbol{s_1}, \cdot)$, where both $\boldsymbol{s_0}$ and $\boldsymbol{s_1}$ are unknown to the adversary. (As before, each distribution is separated into a coin sampling stage and a deterministic stage.) Note that $\boldsymbol{s_0}$ and $\boldsymbol{s_1}$ need not be of the same type, since the adversary does not see them. Jumping ahead in (H)IBE part in the case study, D_0, D_1 correspond to the real/simulated key generation algorithms, where s_0 is the master secret key, and s_1 is the secret trapdoor information the simulator knows in order to answer secret key queries.

```
Game $G_0$ :
  $\boldsymbol{s_0} \xleftarrow{\$} \mathbb{Z}_q^m$;
  $b \leftarrow \mathcal{A}()$
   with $O(x) = \{$
     $c_0 \xleftarrow{\$} \{0,1\}^{d_0}$;
     ret $D_0(c_0, s_0, x)$;
   $\}$
```

```
Game $G_1$ :
  $\boldsymbol{s_1} \xleftarrow{\$} \mathbb{Z}_q^k$;
  $b \leftarrow \mathcal{A}()$
   with $O(x) = \{$
     $c_1 \xleftarrow{\$} \{0,1\}^{d_1}$;
     ret $D_1(c_1, s_1, x)$;
   $\}$
```

Figure 11: Example axiom capturing computational closeness of distributions.

4 Deciding deducibility

Several rules involve deducibility problems as side-conditions. For instance, in the [ABSTRACT] rule from Fig 8, we may transform a bound involving G_1 and G_2 into a bound involving G_1' and G_2', if there exists a common subgame $\mathcal{B}$ which can be used to factor the former pair into the latter. Finding this subgame $\mathcal{B}$ will induce deducibility subproblems. In order to automate the application of the rules, it is thus necessary to provide algorithms for checking whether deducibility problems are valid. As previously argued, it is desirable whenever possible that these algorithms are based on decision procedures rather than heuristics.

In this section, we provide decision procedures for the theory of Diffie-Hellman exponentiation, both in its basic form and in its extension to bilinear groups, and for the theory of fields. The decision procedures for Diffie-Hellman exponentiation are based on techniques from Gröbner bases. In addition to being an important independent contribution on its own, the algorithms for Diffie-Hellman exponentiation also serve as a natural intermediate objective towards addressing the theory of matrices (although the problems are formally independent). For the latter, we require significantly more advanced algebraic tools. For the clarity of exposition, we proceed incrementally. Concretely, we start by considering the case of fields and non-commutative rings. We respectively provide a decision procedure and a semi-decision procedure. Subsequently, we give a reduction from deducibility for matrices to deducibility for non-commutative rings. The reduction yields a semi-decision procedure for matrices. The algorithms for non-commutative rings and matrices are based on so-called SAGBI [RS90] (Subalgebra Analog to Gröbner Basis for Ideals) techniques, which as justified below provide a counterpart of Gröbner basis computations for subalgebras.

4.1 Diffie-Hellman exponentiation

Diffie-Hellman exponentiation is a standard theory that is used for analyzing key-exchange protocols based on group assumptions. It is also used, in its bilinear and multilinear version, in AutoG&P for proving security of pairing-based cryptography. In this setting, the adversary (also often called attacker in the symbolic setting) can multiply groups elements between them, i.e perform addition in the field, and can elevate a group element to some power he can deduce in the field. Previous work only provides partial solutions: for instance, Chevalier et al [CKRT03] only consider products in the exponents, whereas Dougherty and Guttman [DG14] only consider polynomials with maximum degree of 1 (linear expressions).

The standard form of deducibility problems that arises in this context is defined as follows: let Y be a set of names sampled in $\mathbb{Z}_q$, g some group generator, $\mathcal{E}$ the equational theory capturing field and groups operations, some set $X \subset Y$, $f_1, ...f_k, h \in \mathbb{K}[Y]$ be a set of polynomials over the names, and Γ be a coherent set of axioms. The deducibility problem is then:

$$\Gamma \models X, g^{f_1}, ..., g^{f_k} \vdash_{\mathcal{E}} g^h$$

Proposition 4.1. *Deducibility for Diffie-Hellman exponentiation is decidable.*

The algorithm that supports the proof of the proposition proceeds by reducing an input deducibility problem to an equivalent membership problem of the saturation of some $\mathbb{Z}_q[X]$-module in $\mathbb{Z}_q[Y]$, and by using an extension for modules [Eis13] of Buchberger's algorithm [Buc76] to solve the membership problem.

The reduction to the membership problem proceeds as follows: first, we reduce deducibility to solving a system of polynomial equations. We then use the notion of saturation for submodules and prove that solving the system of polynomial equations corresponding to the deducibility problem is equivalent to checking whether the polynomial h is a member of the saturation of some submodule M. The latter problem can be checked using Gröbner basis computations.

4.2 Fields and non-commutative rings

Another problem of interest is when we consider deducibility inside the field rather than the group. The deducibility problem can then be defined as follows: let Y be a set of names sampled in $\mathbb{Z}_q$, $\mathcal{E}$ the equational theory capturing field operations, $f_1, ...f_k, h \in \mathbb{K}[Y]$ be a set of polynomials over the names, and Γ be a coherent set of axioms. The deducibility problem is then:

$$f_1, ..., f_k \vdash_{\mathcal{E}} h$$

We emphasize that this problem is in fact not an instance of the problem for Diffie-Hellman exponentiation. In the previous problem, if we look at field elements, the adversary could compute any polynomial in $K[X]$ but he may now compute any polynomial in $K[f_1, ..., f_k]$, the subalgebra generated by the known polynomials.

Decidability is obtained thanks to [SS88], where they solve the subalgebra membership problem using methods based on classical Gröbner basis.

Proposition 4.2. *Deducibility for fields is decidable.*

If we wish to characterize the full adversary knowledge as done for Diffie-Hellman exponentiation using Gröbner basis, we would have to resort to so-called SAGBI [RS90] (Subalgebra Analog to Gröbner Basis for Ideals) techniques, which form the counterpart of Gröbner basis computations. However, some finitely generated subalgebras are known to have infinite SAGBI bases [RS90], thus it can only provide semi-decision for the membership problem.

For the case of non-commutative rings, we are not aware of any counterpart to [SS88], we resort to the non-commutative SAGBI [Nor98] theory.

Proposition 4.3. *Deducibility for non-commutative rings is semi-decidable.*

It is an open problem whether one can give a decision procedure for non-commutative rings. We note that the problem of module membership over a non-commutative algebra is undecidable [Mor94], as there is a reduction from the word problem over a finitely presented group. On the other hand, the problem is known to be decidable for some classes of subalgebras, notably in the the homogeneous case where all monomials are of the same degree.

4.3 Matrices

The case of matrices introduces a final difficulty: expressions may involve structural operations. To address the issue, we show that every deducibility problem in the theory of matrices is provably equivalent to a deducibility problem that does not involve structural operations, nor transposition—said otherwise, a deducibility problem in the theory of non-commutative rings.

Proposition 4.4. *Deducibility for matrices is semi-decidable.*

The algorithm that supports the proof of semi-decidability for matrices operates in two steps:

1. it reduces the deducibility problem for matrices to an equivalent deducibility problem for non-commutative rings;

2. it applies the semi-decision procedure for non-commutative rings.

The reduction to non-commutative rings is based on a generalization of the techniques introduced in [BDK$^+$10] for the theory of bitstrings—note that the techniques were used for a slightly different purpose, i.e. deciding equivalence between probabilistic expressions, rather than for proving deducibility constraints.

The general idea for eliminating concatenation and splitting comes from two basic facts:

- $\mathcal{M} \vdash M \parallel N \Leftrightarrow \mathcal{M} \vdash M \wedge \mathcal{M} \vdash N$
- $\mathcal{M} \cup \{M \parallel N\} \vdash T \Leftrightarrow \mathcal{M} \cup \{M, N\} \vdash T$

For transposition, we observe that it commutes with the other operations, so in a proof of deducibility, we can push the transposition applications to the leaves. Everything that can be deduced from a set of matrices $\mathcal{M}$ and the transpose operation can also be deduced if instead of the transpose operation we simply provide the transposition of the matrices in $\mathcal{M}$.

5 Implementations and Case Studies

The implementation of our logic, called AutoLWE, is available at:

`https://github.com/autolwe/autolwe`

AutoLWE is implemented as a branch of AutoG&P and thus makes considerable use of its infrastructure.

Moreover, we have used AutoLWE to carry several case studies (see Figure 12): an Identity-Based Encryption scheme and an Hierarchical Identity-Based Encryption scheme by Agrawal, Boneh and Boyen [ABB10], a Chosen-Ciphertext Encryption scheme from Micciancio and Peikert [MP12], and an Inner Product Encryption scheme and proof from Agrawal, Freeman, and Vaikuntanathan [AFV11]. These examples are treated in Sections 5.2, 5.4, 5.3 and 5.5 respectively.

Globally, our tool performs well, on the following accounts: formal proofs remains close to the pen and paper proofs; verification time is fast (less than 3 seconds), and in particular the complexity of the (semi-)decision procedures is not an issue; formalization time is moderate (requiring at most several hours of programmer effort per proof). One of the main hurdles is the Leftover Hash Lemma, which must be applied in varying levels of sophistication. The Leftover Hash Lemma (and more generally all oracle-relative assumptions) increase the difficulty of guessing (chained) applications of assumptions, and consequently limits automation.

Case study			Proof
Reference	Scheme	Property	LoC
Gentry et al. '08 [GPV08]	dual-Regev PKE	IND-CPA	11
Micciancio et al. '12 [MP12]	MP-PKE	IND-CCA	98
Agrawal et al. '10 [ABB10]	ABB-IBE	IND-sID-CPA	56
Agrawal et al. '10 [ABB10]	ABB-HIBE	IND-sID-CPA	77
Agrawal et al. '11 [AFV11]	AFV-IPE	IND-wAH-CPA	106

Figure 12: Overview of case studies. All proofs took less than three seconds to complete.

5.1 Implementation

Security games are written in a syntax closely resembling that shown in Figure 1. See Figure 5 for an example concrete proof in our system. Each line of the proof corresponds to a proof rule in our logic, as seen in Figure 8. All tactic applications are fully automated, except for the application of oracle-relative assumptions. The user must provide some hints to AutoLWE about how the security game needs to be factored in order to apply an oracle-relative assumption. The system in [BGS15] additionally supports a proof search tactic which automatically finds a series of tactics to apply to finish the goal; we do not have a version of that in our setting.

5.1.1 Oracle-relative Assumptions

AutoG&P allows one to add user defined axioms, both to express decisional assertions (two distributions are computationally close) and computational assertions (a certain event has small chance of happening). In AutoG&P, these user-defined axioms are stated in terms of symbolic adversaries, which are related to the main security game by rules such as [ABSTRACT] in Section 3.4. However, the symbolic adversaries present in axioms may not have oracles attached to them. While these restricted adversaries can be used to define the LWE assumption, they are not expressive enough to state the oracle-relative axioms we use throughout our proofs. In AutoLWE, we remove this restriction. An example axiom we now support which we did not before is that in Figure 11.

Recall that in order to apply a user defined axiom using [ABSTRACT], we must factor the security game into one which is in terms of the axiom's game. This is done essentially by separating the security game into sections, where each section either reflects the setup code for the axiom, or an instantiation of one of the adversaries in the axiom. We still do this factoring in the case of oracle-relative axioms, but we must also factor oracles in the security game in terms of oracles in the axiom. Once this second step of factoring is done, oracles in the axiom can be compared syntactically to factored oracles in the security game.

5.1.2 Theory of Lists and Matrices

Note that in our case studies, we manipulate both matrices and *lists* of matrices (often simultaneously). Thus, both our normal form algorithm and our deducibility reduction from Section 4.3 must be lifted to apply to lists of matrices as well. This is what allows our system to reason about the more complicated HIBE scheme in a manner similar to the IBE scheme, which does not use lists.

In order to do this, we do not implement our main algorithms on expressions of matrices directly, but instead over a general *signature* of matrices, encoded as a certain type of an ML module. We then instantiate this signature both with matrices and lists of matrices. By doing so, we receive an implementation for our new algorithms which operate uniformly across these two types of expressions.

5.1.3 Deduction algorithms

Many implementations of Gröbner basis computations can be found online, but all of them are only usable for polynomial ideals. In order to handle module and non-commutative subalgebra, we thus implemented generic versions of the Buchberger algorithm for $K[X]$-module and the SAGBI algorithm and plugged them into AutoLWE. The algorithms performed well: we could prove all the LWE examples, and the pairing-based examples very quickly, using the SAGBI methods. The efficiency of the computations contrasts with the complexity of the algorithms, which is high because the saturation squares up the number of inputs terms and the Gröbner Basis can be at worst a double exponential. However, we are dealing with relatively small instances of our problem that are extracted from concrete primitives.

5.2 Identity-Based Encryption

Mathematical background. Let Λ be a discrete subset of $\mathbb{Z}^m$. For any vector $\boldsymbol{c} \in \mathbb{R}^m$, and any positive parameter $\sigma \in \mathbb{R}$, let $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi||\boldsymbol{x} - \boldsymbol{c}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with center $\boldsymbol{c}$ and parameter σ. Next, we let $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$ be the discrete integral of $\rho_{\sigma,\boldsymbol{x}}$ over Λ, and let $\chi_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{y}) := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\sigma,\boldsymbol{c}}(\Lambda)}$. Let S^m denote the set of vectors in $\mathbb{R}^m$ whose length is 1. The norm of a matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$ is defined to be $\mathsf{sup}_{\boldsymbol{x}\in S^m}||\mathbf{R}\boldsymbol{x}||$. We say a square matrix is full rank if all rows and columns are linearly independent.

Identity-based encryption is a generalization of public key encryption. In IBE, the secret key and ciphertext are associated with different identity strings, and decryption succeeds if and only if the two identity strings are equivalent. The security model, IND-sID-CPA, requires adversary to declare challenge identity upfront before seeing the public parameters, and allows adversary to ask for secret key for any identity except for the challenge identity, and CPA security holds for ciphertext associated with the challenge identity.

The IBE scheme our system supports is constructed by Agrawal et al. [ABB10]. The scheme operates as follows:

- Matrix $\mathbf{A}$ is generated by algorithm TrapGen, which outputs a random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a small norm matrix $\mathbf{T} \in Z_q^{m\times m}$ such that $\mathbf{A} \cdot \mathbf{T_A} = 0$. Matrices $\mathbf{A}_1, \mathbf{B}$ are sampled randomly from $\mathbb{Z}_q^{n\times m}$, and $\boldsymbol{u}$ is sampled randomly from $\mathbb{Z}_q^n$. Set $\mathsf{pp} = (\mathbf{A}, \mathbf{A}_1, \mathbf{B}, \boldsymbol{u})$ and $\mathsf{msk} = \mathbf{T_A}$.

- To encrypt a message $\mu \in \{0, 1\}$ with identity $\mathsf{id} \in \mathbb{Z}_q^n$, one generates a uniform $\boldsymbol{s} \in \mathbb{Z}_q^n$, error vector $\boldsymbol{e}_0 \leftarrow \chi^m$ and error integer $e_1 \leftarrow \chi$ from discrete Gaussian, a random $\mathbf{R} \in \{0, 1\}^{m\times m}$, and computes ciphertext
$$\mathsf{ct} = \boldsymbol{s}^\mathsf{T}\left[\mathbf{A}||\mathbf{A}_1 + M(\mathsf{id})\mathbf{B}||\boldsymbol{u}\right] + (\boldsymbol{e}^\mathsf{T}||\boldsymbol{e}^\mathsf{T}\mathbf{R}||e') + (0||0||\lceil q/2\rceil\mu)$$

- The secret key for identity $\mathsf{id} \in \mathbb{Z}_q^n$ is generated by procedure $\boldsymbol{r} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_1{+}M(\mathsf{id})\mathbf{B}, \mathbf{T_A}, \boldsymbol{u})$, where we have $\boldsymbol{r}$ is statistically close to χ^{2m}, and $[\mathbf{A}||\mathbf{A}_1 + M(\mathsf{id})\mathbf{B}]\,\boldsymbol{r} = \boldsymbol{u}$.

The idea of the proof is first to rewrite $\mathbf{A}_1$ as $\mathbf{AR} - M(\mathsf{id}^*)\mathbf{B}$, where id^* is the adversary's committed identity. If we do so, we then obtain that the challenge ciphertext is of the form

$$\mathbf{s}^\mathsf{T}\left[\mathbf{A}||\mathbf{AR}||\mathbf{u}\right] + (\mathbf{e}^\mathsf{T}||\mathbf{e}^T\mathbf{R}||e') + (0||0||\lceil q/2\rceil\mu),$$

where $\mathbf{A}$ comes from TrapGen. We then apply a computational lemma about SampleLeft, in order to rewrite the KeyGen oracle to be in terms of another probabilistic algorithm, SampleRight. This is a statement about equivalence of distributions from which the adversary may sample, so must be handled using an oracle-relative assumption. This is done as described in Section 3.6.3. The computational lemma states that, for appropriately sampled matrices,

$$\text{SampleLeft}(\mathbf{A}, \mathbf{AR} + \mathbf{B}, T_{\boldsymbol{A}}, \boldsymbol{u}) \approx \text{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, T_{\boldsymbol{B}}, \boldsymbol{u}),$$

where $\mathbf{A}$ is sampled from TrapGen in the first and uniform in the second, and $\mathbf{B}$ is sampled uniformly in the first and from TrapGen in the second. By applying this transformation to our KeyGen oracle, we transform our matrix $\mathbf{A}$ from one sampled from TrapGen to uniform.

Now that $\mathbf{A}$ is uniform, we finish the proof by noticing that our challenge ciphertext is equal to $\mathbf{b}||\mathbf{bR}||b{+}\lceil q/2\rceil\mu$, where $(\mathbf{b}, b)$ forms an LWE distribution of dimension $n \times m + 1$. Thus we may randomize b to uniform, and apply the `rnd` tactic to erase μ from the ciphertext.

The main point of interest in this proof is the initial rewrite $\mathbf{A}_1 \to \mathbf{AR} - M(\mathsf{id}^*)\mathbf{B}$. Given that $\mathbf{A}_1$ is uniform, we may first apply optimistic sampling to rewrite $\mathbf{A}_1$ to $\mathbf{A}_2 - M(\mathsf{id}^*)\mathbf{B}$, where $\mathbf{A}_2$ is uniformly sampled. Thus, we now only need to perform the rewrite $\mathbf{A}_2 \to \mathbf{AR}$. This rewrite is not at all trivial, because $\mathbf{A}$ at this point in the proof comes from TrapGen. However, as noted in Section 3.6.2, it is sound to apply the LHL in this case, because TrapGen generates matrices which are close to uniform in distribution. Thus, we can use the LHL as encoded in Figure 10.

5.3 CCA1-PKE

The CCA1-PKE scheme we study is proposed by Micciancio and Peikert [MP12]. In comparison with the CPA-PKE scheme [GPV08] described in Section 2, the security model of CCA1-PKE is stronger: the adversary can query a decryption oracle for any ciphertext he desires before receiving the challenge ciphertext. The scheme operates as follows:

- Matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is sampled randomly and $\mathbf{R} \leftarrow \{-1, 1\}^{m \times m}$. Set $\mathsf{pk} = (\mathbf{A}, \mathbf{AR})$ and $\mathsf{sk} = \mathbf{R}$.
- Let $M : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n \times m}$ be an embedding from $\mathbb{Z}_q^n$ to matrices, such that for distinct $\mathbf{u}$ and $\mathbf{v}$, $M(\mathbf{u}) - M(\mathbf{v})$ is full rank. To encrypt a message $\mu \in \{0, 1\}$, one generates a uniform $\mathbf{s} \in \mathbb{Z}_q^n$, a uniform $\mathbf{u} \in \mathbb{Z}_q^n$, a uniform matrix $R' \in \{-1, 1\}^{m \times m}$ and an error vector $\mathbf{e} \in \mathbb{Z}_q^m$ sampled from a discrete Gaussian, and computes the ciphertext

  $$\mathbf{c}_0 = \mathbf{u}, \mathbf{c}_1 = \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathbf{u}} + (\mathbf{e}^{\mathsf{T}} || \mathbf{e}^{\mathsf{T}} * R') + (0 || \mathsf{Encode}(\mu))$$

  where $\mathbf{A}_{\mathbf{u}} := [\mathbf{A} || -\mathbf{AR} + M(\mathbf{u})\mathbf{G}]$, $\mathbf{G}$ is a publicly known gadget matrix, and $\mathsf{Encode} : \{0, 1\} \to \mathbb{Z}_q^m$ sends μ to $\mu \lceil q/2 \rceil (1, \ldots, 1)$.
- To decrypt a ciphertext $(\mathbf{u} := \mathbf{c}_0, \mathbf{c}_1)$ with $\mathsf{sk} = \mathbf{R}$ and $\mathbf{u} \neq 0$, one computes $\mathbf{A}_{\mathbf{u}}$ and calls a procedure $\mathsf{Invert}(\mathbf{A}_{\mathbf{u}}, \mathbf{R}, \mathbf{c}_1)$, which will output $\mathbf{s}$ and $\mathbf{e}$ such that $\mathbf{c}_1 = \mathbf{s}^T \mathbf{A}_{\mathbf{u}} + \mathbf{e}$, where $\mathbf{e}$ has small norm. By doing a particular rounding procedure using $\mathbf{c}_1, \mathbf{s}, \mathbf{e}$, and $\mathbf{R}$, the message bit μ can be derived.

The main subtlety of the proof is that the secret key $\mathbf{R}$ is used in the decryption oracle. Because of this, we must apply the Leftover Hash Lemma relative to this oracle, by using oracle-relative axioms. As we will see, not all uses of the LHL are valid in this new setting; care must be taken to ensure that the axioms derived from the LHL are still cryptographically sound.

The high-level outline of the proof is as follows: first, we note that instead of using a fresh $\mathbf{R}'$ to encrypt, we can actually use the secret key $\mathbf{R}$. This is justified by the following corollary of the Leftover Hash Lemma: the distribution $(\mathbf{A}, \mathbf{AR}, \boldsymbol{e}, \boldsymbol{e}\mathbf{R}')$ is statistically close to the distribution $(\mathbf{A}, \mathbf{AR}, \boldsymbol{e}, \boldsymbol{e}\mathbf{R})$ where $\mathbf{A}, \mathbf{R}, \mathbf{R}'$, and $\boldsymbol{e}$ are sampled as in the scheme. This corollary additionally holds true relative to the decryption oracle, which makes use of $\mathbf{R}$.

Once we use $\mathbf{R}$ to encrypt instead of $\mathbf{R}'$, we again use the Leftover Hash Lemma to transform $\mathbf{AR}$ into $-\mathbf{AR} + M(\boldsymbol{u})G$, where $\boldsymbol{u}$ is generated from the challenge encryption. Again, this invocation of the Leftover Hash Lemma is stated relative to the decryption oracle. Crucially, note here that we do *not* transform $\mathbf{AR}$ directly into uniform, as we did before: the reason being is that this transformation would actually be *unsound*, because it would decouple the public key from $\mathbf{R}$ as it appears in the decryption oracle. Thus, we must do the transformation $\mathbf{AR} \to -\mathbf{AR} + M(\boldsymbol{u})G$ in one step, which is cryptographically sound relative to the decryption oracle. (Currently, we must write this specialized transformation as a unique variant of the Leftover Hash Lemma, as discussed in Section 3.6.2; future work will involve unifying these separate variants.)

At this point, we may apply the LWE assumption along with a more routine invocation of the LHL in order to erase the message content from the challenge ciphertext, which finishes the proof.

5.4 Hierarchical Identity-Based Encryption

Hierarchical IBE is an extension of IBE. In HIBE, the secret key for ID string id can delegate secret keys for ID strings id', where id is a prefix for id. Moreover, decryption succeeds if the ID string for the secret key is a prefix of (or equal to) the ID string for the ciphertext. The security model can be adapted according to the delegation functionality.

The HIBE construction our system supports is described in [ABB10]. The ID space for HIBE is $\mathsf{id}_i \in (\mathbb{Z}_q^n)^d$. The secret key for ID string $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, where $\mathsf{id}_i \in \mathbb{Z}_q^n$, is a small-norm matrix $\mathbf{T}$, such that $\mathbf{F}_{\mathsf{id}}\mathbf{T} = 0$, and

$$\mathbf{F}_{\mathsf{id}} = [\mathbf{A}_0 || \mathbf{A}_1 + M(\mathsf{id}_1)\mathbf{B} || \cdots || \mathbf{A}_\ell + M(\mathsf{id}_\ell)\mathbf{B}]$$

We note that $\mathbf{T}$ can be computed as long as we know the secret key for id', where id' is a prefix of id. Ciphertext for ID string id can be generated similarly with respect to matrix $\mathbf{F}_{\mathsf{id}}$.

The security proof of HIBE is similar to the counterpart of IBE. The challenge ID string $\mathsf{id}^* = (\mathsf{id}_1^*, \ldots, \mathsf{id}_\ell^*)$ is embedded in pp as

$$\forall i \in [\ell], \mathbf{A}_i = \mathbf{A}\mathbf{R}_i - M(\mathsf{id}_i^*)\mathbf{B},\ \forall \ell < j \leq d, \mathbf{A}_j = \mathbf{A}\mathbf{R}_j$$

For admissible query $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_k)$, where id is not a prefix of id^*, we have

$$\mathbf{B}_k = [(M(\mathsf{id}_1) - M(\mathsf{id}_1^*))\mathbf{B}|| \cdots ||(M(\mathsf{id}_k) - M(\mathsf{id}_k^*))\mathbf{B}] \neq 0$$

Then we can generate secret key for id using information $\mathbf{B}_k$ and $\mathbf{R}_k = (\mathbf{R}_1|| \cdots ||\mathbf{R}_k)$. In previous cases, we manipulate and apply rewriting rules to matrices. However, in order to reason about the security in a similar manner to pen-and-paper proof, we introduce the *list* notation, and adapt our implementation to operate uniformly across these two types of expressions.

5.5 Inner Product Encryption

The IPE scheme our scheme supports is described in [AFV11]. We briefly recall their construction as

- Matrix $\mathbf{A}$ is generated by algorithm $\mathsf{TrapGen}$. Matrices $\{matB_i\}_{i\in[d]}$ are sampled randomly from $\mathbb{Z}_q^{n\times m}$, and random vector $\boldsymbol{u}$ is from $\mathbb{Z}_q^n$. The public parameters $\mathsf{pp} = (\mathbf{A}, \{\mathbf{B}_i\}_{i\in[d]}, \boldsymbol{u})$, and $\mathsf{msk} = \mathbf{T}_\mathbf{A}$.

- Secret key $\mathsf{sk}_{\boldsymbol{v}} = \boldsymbol{r}$ for vector $\boldsymbol{v} \in \mathbb{Z}_q^d$ is computed by algorithm $\boldsymbol{r} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \sum_{i\in[d]} \mathbf{B}_i\mathbf{G}^{-1}(v_i\mathbf{G}), \mathbf{T}_\mathbf{A}, \boldsymbol{u})$, where for operation $\mathbf{G}^{-1}(\cdot) : \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_q^{m\times m}$, for any $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, it holds that $\mathbf{G}\cdot\mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$ and $\mathbf{G}^{-1}(\mathbf{A})$ has small norm.

- To encrypt a message $\mu \in \{0,1\}$ for attribute $\boldsymbol{w}$, one generates a uniform $\boldsymbol{s} \in \mathbb{Z}_q^n$, error vector $\boldsymbol{e}_0 \leftarrow \chi^m$ and error integer $e_1 \leftarrow \chi$ from discrete Gaussian, random matrices $\{\mathbf{R}_i\}_{i\in[d]} \in \{0,1\}^{m\times m}$, and computes ciphertext $(\boldsymbol{c}_0, \{\boldsymbol{c}_i\}_{i\in[d]}, c)$ as

$$\boldsymbol{c}_0 = \boldsymbol{s}^\mathsf{T}\mathbf{A} + \boldsymbol{e}_0^\mathsf{T},\ \boldsymbol{c}_i = \boldsymbol{s}^\mathsf{T}(\mathbf{B}_i + w_i\mathbf{G}) + \boldsymbol{e}_0^\mathsf{T}\mathbf{R}, c = \boldsymbol{s}^\mathsf{T}\boldsymbol{u} + e + \lceil q/2 \rceil\mu$$

The main challenge in the proof is to answer secret key queries for any vector $\boldsymbol{v}$ as long as $\langle \boldsymbol{v}, \boldsymbol{w}_0\rangle$, $\langle \boldsymbol{v}, \boldsymbol{w}_1\rangle$ are both not 0, where $(\boldsymbol{w}_0, \boldsymbol{w}_1)$ is declared by adversary upfront. The attribute $\boldsymbol{w}_b$ (b is a random bit) is first embedded in pp, i.e. $\mathbf{B}_i = \mathbf{A}\mathbf{R}_i - w_{bi}\mathbf{G}, \forall i \in [d]$, where $\mathbf{R}_i$ is a small matrix. By unfolding the matrix for query $\boldsymbol{v}$, we have

$$\left[\mathbf{A}|| \sum_{i\in[d]} \mathbf{B}_i\mathbf{G}^{-1}(v_i\mathbf{G})\right] = \left[\mathbf{A}||\mathbf{A}\sum_{i\in[d]} \mathbf{R}_i\mathbf{G}^{-1}(v_i\mathbf{G}) + \langle \boldsymbol{w}_b, \boldsymbol{v}\rangle\mathbf{G}\right]$$

If $\langle \boldsymbol{w}_b, \boldsymbol{v}\rangle \neq 0$, the algorithm $\mathsf{SampleRight}$ can be used to generate secret key for $\boldsymbol{v}$.

The sequence of hybrids generated in symbolic proof is a bit different from the pen-and-paper proof. In particular, instead of transforming from embedding of challenge attribute $\boldsymbol{w}_0$ directly to embedding of $\boldsymbol{w}_1$, we use the original scheme as a middle game, i.e. from embedding of $\boldsymbol{w}_0$ to original scheme, then to embedding of $\boldsymbol{w}_1$. The reason for using the original scheme again in the proof is that when using LHL to argue the indistinguishability between $(\mathbf{A}, \{\mathbf{B}_i = \mathbf{A}\mathbf{R}_i - w_{0i}\mathbf{G}\}_i)$ and $(\mathbf{A}, \{\mathbf{B}_i = \mathbf{A}\mathbf{R}_i - w_{1i}\mathbf{G}\}_i)$, the real public parameters $(\mathbf{A}, \{\mathbf{B}_i\}_i)$ actually serves as a middleman. Therefore, to ensure the consistency with respect to public parameters and secret key queries, the real scheme is used to make the transformation valid.

6 Related work

For space reasons, we primarily focus on related works whose main purpose is to automate security proofs in the computational model.

Corin and den Hartog [CdH06] show chosen plaintext security of ElGamal using a variant of a general purpose probabilistic Hoare logic. In a related spirit, Courant, Daubignard, Ene, Lafourcade and Lakhnech [CDE$^+$08] propose a variant of Hoare logic that is specialized for proving chosen plaintext security of padding-based encryption, i.e. public-key encryption schemes based on one-way trapdoor permutations (such as RSA) and random oracles. Later, Gagné, Lafourcade, Lakhnech and Safavi-Naini [GLLS09, GLL13] adapt these methods to symmetric encryption modes and message authentication codes.

Malozemoff, Katz and Green [MKG14] and Hoang, Katz and Malozemoff [HKM15] pursue an alternative approach for proving security of modes of operations and authenticated encryption schemes. Their approach relies on a simple but effective type system that tracks whether values are uniform and fresh, or adversarially controlled. By harnessing their type system into a synthesis framework, they are able to generate thousands of constructions with their security proofs, including constructions whose efficiency compete with state-of-the-art algorithms that were discovered using conventional methods. Using SMT-based methods, Tiwari, Gascón and Dutertre [TGD15] introduce an alternative approach to synthesize bitvector programs, padding-based encryption schemes and modes of operation.

Our work is most closely related to CIL [BDKL10], ZooCrypt [BCG$^+$13] and AutoG&P [BGS15]. Computational Indistinguishability Logic (CIL) [BDKL10] is a formal logic for reasoning about security experiments with oracle and adversary calls. CIL is general, in that it does not prescribe a syntax for games, and side-conditions are mathematical statements. CIL does not make any provision for mechanization, although, as any mathematical development, CIL can be formalized in a proof assistant, see [CDL11]. ZooCrypt [BCG$^+$13] is a platform for synthesizing padding-based encryption schemes; it has been used successfully to analyze more than a million schemes, leading to the discovery of new and interesting schemes. ZooCrypt harnesses two specialized computational logics for proving chosen-plaintext and chosen-ciphertext security, and effective procedures for finding attacks. The computational logics use deducibility to trigger proof steps that apply reduction to one-wayness assumptions, and to compute the probability of bad events using a notion of symbolic entropy. However, ZooCrypt is highly specialized.

AutoG&P [BGS15] introduce a computational logic and provide an implementation of their logic, called AutoG&P, for proving security of pairing-based cryptographic constructions. Their logic uses deducibility for ensuring that proof rules are correctly enforced. Their implementation achieves a high level of automation, thanks to a heuristics for checking deducibility, and a proof search procedure, which decides which proof rule to apply and automatically selects applications of computational assumptions. We build heavily on this work; in particular, AutoLWE is implemented as an independent branch of AutoG&P. The main differences are:

- AutoLWE supports oracle-relative assumptions and general forms of the Leftover Hash Lemma, and (semi-)decision procedures for deducibility problems, for the theories of Diffie-Hellman exponentiation, fields, non-commutative rings and matrices. In contrast, AutoG&P only support more limited assumptions and implements heuristics for the theory of Diffie-Hellman exponentiation;

- AutoG&P supports automated generation of EasyCrypt proofs, which is not supported by AutoLWE. Rather than supporting generation of proofs a posteriori, a more flexible alternative would be to integrate the features of AutoG&P and AutoLWE in EasyCrypt.

Theodorakis and Mitchell [TM18] develop a category-theoretical framework for game-based security proofs, and leverage their framework for transferring such proofs from the group-based or pairing-based to the lattice-based setting. Their results give an elegant proof-theoretical perspective on the relationship between cryptographic proofs. However, they are not supported by an implementation. In contrast, we implement our computational logic. Furthermore, proofs in AutoLWE have a first-class status, in the form of proof scripts. An interesting direction for future work is to implement automated compilers that transform proofs from the group- and pairing-based settings to the lattice-based settings. Such proof compilers would offer a practical realization of [TM18] and could also implement patches when they fail on a specific step.

7 Conclusion

We have introduced a symbolic framework for proving the security of cryptographic constructions based on the (decisional) Learning with Errors assumption. A natural step for future work is to broaden the scope of our methods to deal with other hardness assumptions used in lattice-based cryptography, including the Ring Learning with Errors assumption, the Short Integer Solution assumption. A further natural step would then be to analyze lattice-based key exchange protocols [Pei14, BCD$^+$16]. To this end, it would be interesting to embed the techniques developed in this paper (and in [BGS15]) into the EasyCrypt proof assistant [BGHZ11, BDG$^+$13], and to further improve automation of EasyCrypt for typical transformations used for proving security of protocols.

Acknowledgements.

This work is partially supported by ONR Grant N000141512750 (Barthe), by IBM under Agreement 4915013672 (Fan), and by NSF Grant 1704788 and NSF CNS-1617676 (Shi, Gancher).

References

[ABB$^+$05] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May 2010.

[AFL16] Daniel Apon, Xiong Fan, and Feng-Hao Liu. Deniable attribute based encryption for branching programs from LWE. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 299–329. Springer, Heidelberg, October / November 2016.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2011.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Heidelberg, August 2014.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 223–238, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BCD$^+$16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018. ACM, 2016.

[BCG$^+$13] Gilles Barthe, Juan Manuel Crespo, Benjamin Grégoire, César Kunz, Yassine Lakhnech, Benedikt Schmidt, and Santiago Zanella Béguelin. Fully automated analysis of padding-based encryption in the computational model. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1247–1260. ACM, 2013.

[BDG$^+$13] Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. Easycrypt: A tutorial. In Alessandro Aldini, Javier Lopez, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 146–166. Springer, 2013.

[BDK$^+$10] Gilles Barthe, Marion Daubignard, Bruce M. Kapron, Yassine Lakhnech, and Vincent Laporte. On the equality of probabilistic terms. In Edmund M. Clarke and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers*, volume 6355 of *Lecture Notes in Computer Science*, pages 46–63. Springer, 2010.

[BDKL10] Gilles Barthe, Marion Daubignard, Bruce M. Kapron, and Yassine Lakhnech. Computational indistinguishability logic. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10*, pages 375–386. ACM Press, October 2010.

[BFK$^+$13] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. Implementing TLS with verified cryptographic security. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 445–459. IEEE Computer Society, 2013.

[BGB09] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In Zhong Shao and Benjamin C. Pierce, editors, *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 90–101. ACM, 2009.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

[BGHZ11] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 71–90. Springer, Heidelberg, August 2011.

[BGS15] Gilles Barthe, Benjamin Grégoire, and Benedikt Schmidt. Automated proofs of pairing-based cryptography. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1156–1168. ACM, 2015.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*, pages 82–96. IEEE Computer Society, 2001.

[Bla06] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *27th IEEE Symposium on Security and Privacy, S&P 2006*, pages 140–154. IEEE Computer Society, 2006.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[Buc76] B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.*, 10(3):19–29, August 1976.

[CDE$^+$08] Judicaël Courant, Marion Daubignard, Cristian Ene, Pascal Lafourcade, and Yassine Lakhnech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 371–380. ACM, 2008.

[CdH06] Ricardo Corin and Jerry den Hartog. A probabilistic hoare-style logic for game-based cryptographic proofs. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2006.

[CDL11] Pierre Corbineau, Mathilde Duclos, and Yassine Lakhnech. Certified security proofs of cryptographic protocols in the computational model: An application to intrusion resilience. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs - First International Conference, CPP 2011, Kenting, Taiwan, December 7-9, 2011. Proceedings*, volume 7086 of *Lecture Notes in Computer Science*, pages 378–393. Springer, 2011.

[CHH$^+$17] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1773–1788. ACM, 2017.

[CKRT03] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the security of protocols with diffie-hellman exponentiation and products in exponents. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science: 23rd Conference, Mumbai, India, December 15-17, 2003. Proceedings*, pages 124–135, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[CLS03] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 271–280, June 2003.

[DFK$^+$17] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Jonathan Protzenko, Aseem Rastogi, Nikhil Swamy, Santiago Zanella Béguelin, Karthikeyan Bhargavan, Jianyang Pan, and Jean Karim Zinzindohoue. Implementing and proving the TLS 1.3 record layer. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 463–482. IEEE Computer Society, 2017.

[DG14] Daniel J. Dougherty and Joshua D. Guttman. Decidability for lightweight diffie-hellman protocols. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 217–231, 2014.

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Trans. Information Theory*, 29(2):198–207, 1983.

[Eis13] David Eisenbud. *Commutative Algebra: with a view toward algebraic geometry*, volume 150. Springer Science & Business Media, 2013.

[GLL13] Martin Gagné, Pascal Lafourcade, and Yassine Lakhnech. Automated security proofs for almost-universal hash for MAC verification. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2013.

[GLLS09] Martin Gagné, Pascal Lafourcade, Yassine Lakhnech, and Reihaneh Safavi-Naini. Automated security proof for symmetric encryption modes. In Anupam Datta, editor, *Advances in Computer Science - ASIAN 2009. Information Security and Privacy, 13th Asian Computing Science Conference, Seoul, Korea, December 14-16, 2009. Proceedings*, volume 5913 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2009.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.

[HKM15] Viet Tung Hoang, Jonathan Katz, and Alex J. Malozemoff. Automated analysis and synthesis of authenticated encryption schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 84–95. ACM, 2015.

[IK03] Russell Impagliazzo and Bruce M. Kapron. Logics for reasoning about cryptographic constructions. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 372–383. IEEE Computer Society, 2003.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 12–24. ACM, 1989.

[KBB17] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, pages 435–450. IEEE, 2017.

[KMM94] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, Jun 1994.

[KMT12] Steve Kremer, Antoine Mercier, and Ralf Treinen. Reducing Equational Theories for the Decision of Static Equivalence. *Journal of Automated Reasoning*, 48(2):197–217, 2012.

[Len85] A.K. Lenstra. Factoring multivariate polynomials over finite fields. *Journal of Computer and System Sciences*, 30(2):235 – 248, 1985.

[Loc16] Andreas Lochbihler. Probabilistic functions and cryptographic oracles in higher order logic. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 503–531. Springer, 2016.

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[MKG14] Alex J. Malozemoff, Jonathan Katz, and Matthew D. Green. Automated analysis and synthesis of block-cipher modes of operation. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 140–152. IEEE Computer Society, 2014.

[Mor94] Teo Mora. An introduction to commutative and noncommutative gröbner bases. *Theoretical Computer Science*, 134(1):131–173, 1994.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MS01] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM CCS 01*, pages 166–175. ACM Press, November 2001.

[Nor98] Patrik Nordbeck. Canonical subalgebraic bases in non-commutative polynomial rings. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, ISSAC '98, pages 140–146, New York, NY, USA, 1998. ACM.

[Pau00] Lawrence Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6, 12 2000.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

[PM15] Adam Petcher and Greg Morrisett. The foundational cryptography framework. In Riccardo Focardi and Andrew C. Myers, editors, *Principles of Security and Trust - 4th International Conference, POST*, volume 9036 of *Lecture Notes in Computer Science*, pages 53–72. Springer, 2015.

[Reg05a] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[Reg05b] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[RKTC03] M. Rusinowitch, R. Küsters, M. Turuani, and Y. Chevalier. An np decision procedure for protocol insecurity with xor. In *Logic in Computer Science, Symposium on(LICS)*, volume 00, page 261, 06 2003.

[RS90] Lorenzo Robbiano and Moss Sweedler. Subalgebra bases. In Winfried Bruns and Aron Simis, editors, *Commutative Algebra*, pages 61–87, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions and composed keys is np-complete. *Theoretical Computer Science*, 299(1):451 – 475, 2003.

[SCF$^+$13] Nikhil Swamy, Juan Chen, Cédric Fournet, Pierre-Yves Strub, Karthikeyan Bhargavan, and Jean Yang. Secure distributed programming with value-dependent types. *J. Funct. Program.*, 23(4):402–451, 2013.

[Sch96] Steve Schneider. Security properties and csp. In *Proceedings of the 1996 IEEE Conference on Security and Privacy*, SP'96, pages 174–187, Washington, DC, USA, 1996. IEEE Computer Society.

[SMCB12] Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David A. Basin. Automated analysis of diffie-hellman protocols and advanced security properties. In Stephen Chong, editor, *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 78–94. IEEE Computer Society, 2012.

[SS88] David Shannon and Moss Sweedler. Using gröbner bases to determine algebra membership, split surjective algebra homomorphisms determine birational equivalence. *J. Symb. Comput.*, 6(2-3):267–273, December 1988.

[TGD15] Ashish Tiwari, Adrià Gascón, and Bruno Dutertre. Program synthesis using dual interpretation. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 482–497. Springer, 2015.

[TM18] Eftychios Theodorakis and John C. Mitchell. Semantic security invariance under variant computational assumptions. *IACR Cryptology ePrint Archive*, 2018:51, 2018.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 619–636, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

A Proofs of section 4.1

In group theory, a multilinear map is a function which goes from a set of groups into a target group, and is linear with respect to all its arguments. They have been used in the past years to develop new schemes, such as Boneh-Boyen Identity Based Encryption [BB04] or Waters' Dual System Encryption [Wat09].

Given a multilinear map $\hat{e}$, $g_1, .., g_n, g_t$ a set of groups generators, let X be a set of public names sampled in $\mathbb{Z}_q$, Y be a set of private names sampled in $\mathbb{Z}_q$, $f_1, ...f_k, h \in \mathbb{K}[X, Y]$ be a set of polynomials over both public and secret names and Γ be a coherent set of axioms.

Our deducibility problem is to decide if $\Gamma \models X, g_{i_1}^{f_1}, ..., g_{i_k}^{f_k} \vdash_{\mathcal{E}} g_t^h$ Without loss of generality, we consider here the case of a bilinear map, to simplify the writing, but the proofs scale up to multilinear maps.

A.1 Saturation into the target group

First, we reduce our problem to the case of a single group. This result comes from the Proposition 1 of [KMT12]. Their constructive proof can trivially be used to obtain the following proposition:

Proposition A.1. *For any sets X and Y, polynomials $f_1, ...f_n, h \in \mathbb{K}[X, Y]$ and groups elements $g_{i_1}^{f_1}, ..., g_{i_n}^{f_n}$, we denote*

$$\begin{aligned}(g_t^{e_i}) = \ & \{\hat{e}(g_{i_j}, g_{i_k}) | 1 \leq j \leq k \leq n, g_{i_j} \in \mathbb{G}_1, g_{i_k} \in \mathbb{G}_2\} \\ & \cup \{\hat{e}(g_{i_j}, 1) | 1 \leq j \leq n, g_{i_j} \in \mathbb{G}_1, \} \\ & \cup \{\hat{e}(1, g_{i_j}) | 1 \leq j \leq n, g_{i_j} \in \mathbb{G}_2, \}\end{aligned}$$

Then $\Gamma \models X, g_{i_1}^{f_1}, ..., g_{i_n}^{f_n} \vdash_{\mathcal{E}} g_t^h \Leftrightarrow \Gamma \models X, g_t^{e_1}, ..., g_t^{e_N} \vdash_{\mathcal{E}-\hat{e}} g_t^h$.

We obtain a problem where we only have elements in the target group, we can therefore reduce the general problem to the single group case.

A.2 Reduction to polynomials

Lemma A.2. *For any sets X and Y, polynomials $w_1, ...w_N, h \in \mathbb{K}[X, Y]$ we have $\Gamma \models X, g_t^{w_1}, ..., g_t^{w_N} \vdash_{\mathcal{E}} g_t^h$ if and only if:*

$$\exists (e_i, g_i) \in \mathbb{K}[X], (\forall i, \Gamma \models g_i \neq 0) \wedge \sum_i e_i \times \frac{f_i}{g_i} = h$$

Proof. If $\Gamma = \emptyset$, the adversary can construct elements of the form $(g_t^{w_i})^{e_i}$, where $e_i \in \mathbb{K}[X]$, i.e e_i is a polynomial constructed over variables fully known by the adversary, and then multiply this kind of term, yielding a sum in the exponent. If $\Gamma \neq \emptyset$, he may also divide by some $g_t^{g_i}$, with $g_i \in \mathbb{K}[X]$. We capture here the three capabilities of the adversary, which when looking in the exponent immediately translate into the formula on the right side. □

To handle this new problem, we notice that we can actually compute the set $\{g | \Gamma \models g \neq 0\}$. Indeed, for each axiom $f \neq 0$, we can extract a finite set of non zero irreducible polynomials by factorizing them (for example using Lenstra algorithm [Len85]). Any non annulling polynomial will be a product of all these irreducible polynomials. We can then obtain a finite set $Gs = (g_i)$ such that $G = \{g | \Gamma \models g \neq 0\} = \{\prod_{g \in Gs} g^{k_g} | \forall g, k_g \in \mathbb{N}\}$. With these notations, we can simplify proposition 1, because we know the form of the g_i. Moreover, as we do not want to deal with fractions, we multiply by the common denominator of all the $\frac{w_i}{g_i}$.

Lemma A.3. *For any sets X and Y, polynomials $w_1, ...w_N, h \in \mathbb{K}[X,Y]$ we have $\Gamma \models X, g_t^{w_1}, ..., g_t^{w_N} \vdash_{\mathcal{E}} g_t^h$ if and only if:*

$$\exists (e_i) \in \mathbb{K}[X], (k_g) \in \mathbb{N}, \sum_i e_i \times w_i = h \prod_{g \in Gs} g^{k_g}$$

We do not prove this lemma, we will rather reformulate it using more refined mathematical structures and then prove it. Let us call $M = \{\sum_i e_i \times w_i | e_i \in \mathbb{K}[X]\}$ the free $\mathbb{K}[X]$-module generated by the (w_i). We recall that a S-module is a set stable by multiplication by S and addition, and that $\langle (w_i) \rangle_S$ is the S-module generated by (w_i). We also recall the definition of the saturation :

Definition A.1. *Given a S-module T, $f \in S$ and $S \subset S'$, the saturation of T by f in S' is $T :_{S'} (f)^\infty = \{g \in S' | \exists n \in \mathbb{N}, f^n g \in T\}$.*

The previous lemma can be reformulated using saturation; if M is the module generated by $w_1, ..., w_N$:

Lemma A.4. $\Gamma \models X, g_t^{w_1}, ..., g_t^{w_N} \vdash_{\mathcal{E}} g_t^h \Leftrightarrow h \in M :_{\mathbb{K}[X,Y]} (g_1 ... g_n)^\infty$

Proof. We recall that:

$$M :_{\mathbb{K}[X,Y]} (g_1 ... g_n)^\infty = \{x \in \mathbb{K}[X,Y] | \exists k \in \mathbb{N}, (g_1 ... g_n)^k \times x \in M\}$$

$\Rightarrow$ We have $\sum_i e_i \times w_i = h \prod_{g \in Gs} g^{k_g}$. With $K = max(k_g)$, we multiply both sides by $\prod_g g^{K - k_g}$ to get $h \prod_{g \in Gs} g^K = \sum_i \prod_g g^{K-k_g} e_i \times w_i \in M$. Which proves that $h \in M :_{\mathbb{K}[X,Y]} (g_1 ... g_n)^\infty$.
$\Leftarrow$ If $h \in M :_{\mathbb{K}[X,Y]} (g_1 ... g_n)^\infty$, we instantly have $(e_i) \in \mathbb{K}[X], k \in \mathbb{N}$ such that

$$h \prod_{g \in Gs} g^{k_g} = \sum_i e_i f_i$$

□

We then simplify the saturation, by transforming it into the membership of the intersection of modules.

Lemma A.5. *For any sets X and Y, $f_1, ...f_n, h \in \mathbb{K}[X,Y]$, $g \in \mathbb{K}[X]$,let $M = \{\sum_i e_i \times f_i | e_i \in \mathbb{K}[X]\}$. Then, with t a fresh variable $M :_{\mathbb{K}[X,Y]} g^\infty = \langle (f_i) \cup ((gt-1)Y^{\boldsymbol{j}})_{\boldsymbol{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]} \cap \mathbb{K}[X,Y]$.*

Proof. $\subset$. Let there be $v \in M :_{\mathbb{K}[X,Y]} g^\infty$. Then, we have k such that $g^k \times v \in M$. The following equalities shows that v is in the right side set $v = g^k t^k v - (1 + gt + ... + g^{k-1}t^{k-1})(gt-1)v$. Indeed, $g^k t^k v \in M\mathbb{K}[X,t]$, so we have $(e_i) \in \mathbb{K}[X,t]$ such that $g^k t^k v = \sum_i e_i f_i$. Moreover, $g^k \times v \in M$ and $g \in \mathbb{K}[X]$ implies that $deg_Y(v) \subset \{deg_Y(f_i)\}$. So we do have $(e_i') \in \mathbb{K}[X,t]$ and $(\boldsymbol{j_i}) \subset \{deg_Y(f_i)\}$ such that

$$(1 + gt + ... + g^{k-1}t^{k-1})(gt-1)v = \sum e_i'(gt-1)Y^{\boldsymbol{j_i}}$$

Finally, $v \in \langle (f_i) \cup ((gt-1)Y^{\boldsymbol{j}})_{\boldsymbol{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]} \cap \mathbb{K}[X,Y]$.

$\supset$. Let there be $v \in \langle (f_i) \cup ((gt-1)Y^{\boldsymbol{j}})_{\boldsymbol{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]} \cap \mathbb{K}[X,Y]$. Then we have $(e_i), (e_i') \in K[X,t]$ and $(\boldsymbol{j_i}) \subset \{deg_Y(f_i)\}$ such that :

$$v = \sum_i e_i f_i + \sum_i e_i'(gt-1)Y^{\boldsymbol{j_i}}$$

We have that $v \in \mathbb{K}[X,Y]$, so v is invariant by t. So, if we subsitute t with $\frac{1}{g}$, we have that $v = \sum_i e_i(X, \frac{1}{g}) f_i$. Let us consider g^k the common denominator of all those fractions and call $e_i'' = g^k e_i \in \mathbb{K}[X]$. We finally have $g^k \times v = \sum_i e_i'' f_i \in M$, which means that $v \in M :_{\mathbb{K}[X,Y]} g^\infty$. □

The Buchberger algorithm allows us to compute a Gröbner basis of any free $\mathbb{K}[X]$-module [Eis13] and then decide the membership problem for a module. We thus solve our membership problem using this method.

Theorem A.6. *For any sets X and Y, polynomials $f_1, ...f_n, h \in \mathbb{K}[X,Y]$, group elements $g_{i_1}, ..., g_{i_n}$ and a set of axioms Γ we can decide if $\Gamma \models X, g_{i_1}^{f_1}, ..., g_{i_n}^{f_n} \vdash_{\mathcal{E}} g_t^h$*

Proof. To decide if h is deducible, we first reduce to a membership problem with lemma A.4 that can be solved using lemma A.5 by computing the Gröbner basis of $\langle (f_i) \cup ((gt-1)Y^{\boldsymbol{j}})_{\boldsymbol{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]}$, keeping only the elements of the base that are independent of t and then checking if the reduced form of h is 0. □

As a side note, being able to decide the deducibility in this setting allows us to decide another classical formal method problem, the static equivalence. Indeed the computation of the Gröbner basis allows us to find generators of the corresponding syzygies (Theorem 15.10 of [Eis13]), which actually captures all the possible distinguishers of a frame.

B Proofs for section 4.3

$$[0]\ \frac{}{\Gamma \vdash 0 : {\mathbb{Z}_q}^{n,m}} \qquad [\mathrm{ID}]\ \frac{}{\Gamma \vdash \mathbf{I} : {\mathbb{Z}_q}^{n,n}} \qquad [\mathrm{TR}]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{m,n}}{\Gamma \vdash M^\top : {\mathbb{Z}_q}^{n,m}}$$

$$[\mathrm{SL}]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{n,m+m'}}{\Gamma \vdash \mathsf{sl}\ M : {\mathbb{Z}_q}^{n,m}} [\mathrm{SR}]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{n,m+m'}}{\Gamma \vdash \mathsf{sr}\ M : {\mathbb{Z}_q}^{n,m'}} \qquad [\text{-}]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{n,m}}{\Gamma \vdash -M : {\mathbb{Z}_q}^{n,m}} \qquad [\in]\ \frac{M \in \mathcal{M}}{\mathcal{M} \vdash M}$$

$$[\times]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{n,\ell} \quad \Gamma \vdash M' : {\mathbb{Z}_q}^{\ell,n}}{\Gamma \vdash M \times M' : {\mathbb{Z}_q}^{n,m}} \qquad [+]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{n,m} \quad \Gamma \vdash M' : {\mathbb{Z}_q}^{n,m}}{\Gamma \vdash M + M' : {\mathbb{Z}_q}^{n,m}}$$

$$[\text{——}]\ \frac{\Gamma \vdash M : {\mathbb{Z}_q}^{n,m} \quad \Gamma \vdash M' : {\mathbb{Z}_q}^{n,m'}}{\Gamma \vdash M || M' : {\mathbb{Z}_q}^{n,m+m'}}$$

Figure 13: Typing rules for matrix operators.

We provide a more detailed proof of Proposition 4.4. To reason about matrices deducibility, written $\mathcal{M} \vdash M$ for a set of matrices $\mathcal{M}$ and a matrix M, we use the natural formal proof system $\mathcal{K}$ which matches the operations on expressions (see Figure 13), that we extend with the equality rule

$$[\mathrm{EQ}]\ \frac{\mathcal{M} \vdash M_1 \quad \mathcal{M} \vdash M_1 = M_2}{\mathcal{M} \vdash M_2}$$

For ease of writing, we denote $\left(\frac{A}{B}\right) := (A^\top || B^\top)^\top$.

Splits elimination

Proposition B.1. *Given a set of matrices $\mathcal{M}$ and a matrix M, we can obtain $\mathcal{S}(\mathcal{M})$ a set of matrices without any concats, such that $\mathcal{M} \vdash M \Leftrightarrow \mathcal{S}(\mathcal{M}) \vdash H$.*

Proof. We notice that the concat operations commute with all the other operators: $(A||B) + (C||D) = (A{+}C||B{+}D)$, $(A||B){-}(C||D) = (A{-}C||B{-}D)$, $A{\times}(B||C) = (AB||AC)$, $(A||B){\times}\frac{C}{D} = AC{+}BD$, $(A||B)^\top = (\frac{A^\top}{B^\top})$. Given a set of matrices $\mathcal{M}$, we rewrite the matrices so that the concatenations operators are at the top. We can see the matrices as block matrices with submatrices without any concat, and then, we can create a set $\mathcal{S}(\mathcal{M})$ containing all the submatrices. This preserves deducibility thanks to the Eq rule for the rewriting, and to the split rules for the submatrices. □

Definition B.1. *We call $\mathcal{N}$ the proof system based on $\mathcal{K}$ without splits, and write the deducibility with $\mathcal{M} \vdash_{\mathcal{N}} M$.*

Lemma B.2. *If $\mathcal{M} \vdash (R||S)$ with a proof π (resp. $\mathcal{M} \vdash (\frac{R}{S})$) then $\mathcal{M} \vdash R$ and $\mathcal{M} \vdash S$ with smaller proofs (resp. $\mathcal{M} \vdash R$, $\mathcal{M} \vdash S$) .*

Proof. We prove it by induction on the size of the proof, and by disjunction on the last rule applied.

Base case $|\pi| = 2$, then the proof is a concat on axioms and we can then obtain the sub matrix directly, with a proof of size one.

Induction case

- The last rule is
$$[\textsc{Tr}]\ \frac{(\dfrac{R}{S})}{(R|S)}$$
Then, we directly obtain by induction on $(\frac{R}{S})$ smaller proofs for R and S.

- The last rule is
$$[\times]\ \frac{M \quad (N^l||N^r)}{(MsN^l|MN^r)}.$$
Then, by induction on the proof of N, we obtain proofs of size smaller than $|\pi| - 1$ of N^l and N^r, and we just have to add a $[\times]$ to those proofs, yielding smaller proofs of MN^l and MN^r.

- If the last rule is [+], [-], [——], the proof can be done similarly to the two previous cases.

- The last rule is
$$[\textsc{sL}]\ \frac{((M|N)|L)}{(M|N)}$$
Then, we have a proof of $((M|N)|L)$ of size $|\pi| - 1$, so by induction we have a proof of $(M|N)$ smaller than $|\pi| - 1$, and by adding a sL, we for instance obtain M with a proof smaller than $|\pi|$.

- [SR] is similar.

□

Lemma B.3. *If $\mathcal{M}$ is a set of matrix without concatenations, and if $\mathcal{M} \vdash M$, then $\mathcal{M} \vdash_{\mathcal{N}} M$.*

Proof. We prove it by induction on the size of the proof, and by disjunction on the last rule applied.

Base case $|\pi| = 1$, then the only problem might be if the rule used was a split, but as we have matrices without concatenations, this is not possible.

Induction case

- If the last rule is [TR], [+], [-], [——], we conclude by applying the induction hypothesis to the premise of the rule.
- The last rule is

$$[\text{sL}]\ \frac{(M|N)}{M}$$

  Then, we have a proof of $(M|N)$ of size $\pi - 1$, and with lemma B.2 we have a smaller proof of M, on which we can then apply our induction hypothesis to obtain a proof of M without split.
- *splitR* is similar.

□

Concatenations elimination

Definition B.2. *We call $\mathcal{T}$ the proof system based on N without concatenations, and write the deducibility $\mathcal{M} \vdash_{\mathcal{T}} M$.*

Lemma B.4. *If $\mathcal{M}$, M, N do not contain any concat, then:*

$$\mathcal{M} \vdash_{\mathcal{N}} (M|N) \Leftrightarrow \mathcal{M} \vdash_{\mathcal{T}} M \wedge \mathcal{M} \vdash_{\mathcal{T}} N$$

Proof. The left implication is trivial. For the right one, we once more do a proof by induction on the size of the proof.

Base case $|\pi| = 1$, the last rule is a [——], and we do have a proof of M and N.

Induction case

- The last rule is

$$[\times]\ \frac{M \quad (N^l||N^r)}{(MN^l|MN^r)}$$

  Then, by induction on the proof of N, we obtain proofs of size smaller than $|\pi| - 1$ of N^l and N^r without concats, and we just have to add a [×] to those proofs, yielding proofs of MN^l and MN^r without concats.
- If the last rule is [TR], [+], [-], [sL], [sR], we proceed as in the previous case
- The last rule is [——]. Then the induction rule instantly yields the expected proofs. Then, we have a proof of $(M|N)$ of size $\pi - 1$, and with lemma B.2 we have a smaller proof of M, on which we can then apply our induction hypothesis to obtain a proof of M without split.

□

Lemma B.5. $\mathcal{M} \vdash_{\mathcal{N}} M \Leftrightarrow \forall G \sqsubseteq M, \mathcal{M} \vdash_{\mathcal{T}} G$ *Where $G \sqsubseteq H$ denotes the fact that G is a sub matrix of M without any concatenation.*

Proof. The left implication is trivial, we prove the right one. As was done in Lemma B.1, we can see M has a bloc matrix, i.e with all the concat at the top.

We are given a proof of $\mathcal{M} \vdash_{\mathcal{N}} M$, which must contain all its concatenations at the bottom of the proof tree. If we look at all the highest concat rule in the proof such that no concat is made before, we have some proof of $\mathcal{M} \vdash_{\mathcal{N}} (M_i|M_j)$, and thanks to lemma B.4, we have $\mathcal{M} \vdash_{\mathcal{T}} M_i \wedge (A_i) \vdash_{\mathcal{T}} M_j$. Applying this to all the highest concat rules in the proof yields the result. □

Removal of the transposition

Definition B.3. *We call $\mathcal{V}$ the proof system based on $\mathcal{N}$ without concat, and write the deducibility $\mathcal{M} \vdash_{\mathcal{V}} M$.*

The transposition commutes with the other operations, given a matrix M we define the normal form $N(M)$ where the transposition is pushed to the variables. We extend the notation for normal form to sets of matrices.

Lemma B.6. $\mathcal{M} \vdash_{\mathcal{T}} M \Leftrightarrow \mathcal{M} \cup (N(\mathcal{M}^t))) \vdash_{\mathcal{V}} N(M)$

Proof. $\Leftarrow$ This is trivial, as the normal form can be deduced using the rule [EQ].
$\Rightarrow$ Given the proof of M, we can commute the trans rule with all the others, and obtain a proof tree where all the transposition are just after a [$\in$] rule. Then, any [$\in$] followed by [TRANS] can be replaced by a [$\in$] and a [EQ] when given the input $\mathcal{M} \cup (N(\mathcal{M}^t)))$ instead of $\mathcal{M}$. We can thus construct a valid proof of $\mathcal{M} \cup (N(\mathcal{M}^t))) \vdash_{\mathcal{V}} N(M)$ □

Conclusion The proof of proposition 4.4 is a direct consequence of Lemmas B.1, B.3, B.5 and B.6.