Automated Unbounded Analysis of Cryptographic Constructions in the Generic Group Model

Miguel Ambrona(✉), Gilles Barthe, and Benedikt Schmidt

IMDEA Software Institute, Madrid, Spain
{miguel.ambrona,gilles.barthe,benedikt.schmidt}@imdea.org

Abstract. We develop a new method to automatically prove security statements in the Generic Group Model as they occur in actual papers. We start by defining (i) a general language to describe security definitions, (ii) a class of logical formulas that characterize how an adversary can win, and (iii) a translation from security definitions to such formulas. We prove a Master Theorem that relates the security of the construction to the existence of a solution for the associated logical formulas. Moreover, we define a constraint solving algorithm that proves the security of a construction by proving the absence of solutions.

We implement our approach in a fully automated tool, the gga^{∞} tool, and use it to verify different examples from the literature. The results improve on the tool by Barthe et al. (CRYPTO'14, PKC'15): for many constructions, gga^{∞} succeeds in proving standard (unbounded) security, whereas Barthe's tool is only able to prove security for a small number of oracle queries.

1 Introduction

The gold standard in provable security is to demonstrate security in the standard model. However, proofs in the standard model sometimes rely on non-standard hardness assumptions. In such situations, it is essential to prove that the hardness assumptions used in the security proofs meet some minimal requirements, for instance the absence of algebraic attacks. The accepted method for validating new DDH-like assumptions is to show absence of generic attacks, i.e. attacks that solely exploit the underlying algebraic structure, using the Generic Group Model [32,33,35,38] or its bilinear and multilinear variants [11,17]. The Generic Group Model provides an algebraic setting for describing a wide class of DDH-like assumptions, and is supported by so-called Master Theorems that give a purely algebraic condition that ensures the security of an assumption in the Generic Group Model (or its variants). Very roughly, the proof of the Master Theorems uses the Schwartz-Zippel Lemma to prove a security reduction between the Generic Group Model and a Symbolic Generic Group Model, in which the security experiment is purely deterministic. Security in the Symbolic Generic Group Model is trivially equivalent to a purely algebraic condition. For

© International Association for Cryptologic Research 2016
M. Fischlin and J.-S. Coron (Eds.): EUROCRYPT 2016, Part II, LNCS 9666, pp. 822–851, 2016.
DOI: 10.1007/978-3-662-49896-5_29

instance, the algebraic condition for a decisional assumption requires to prove that the two sets of polynomials extracted from the left and right games have the same linear dependencies. Therefore, and unavoidably, the difficulty of checking the algebraic condition increases as the assumption becomes more complex, as witnessed by unfortunate failures [24,28,39]. For some recent hypotheses, several pages of error-prone calculations are required for proving that the algebraic condition holds, and several authors have used computer algebra systems to carry part of the verifications. These examples suggest the importance of building general tools to assist proofs of security assumptions in the Generic Group Model. One such tool is the Generic Group Analyzer [11], which uses SMT solvers and computer algebra systems to analyze DDH-like assumptions. The tool takes as input a description of an assumption and either returns an algebraic attack or a concrete probability bound if the assumption is secure. The Generic Group Analyzer primarily works for non-interactive assumptions, in which the adversary can only call the oracles which perform the algebraic operations.

The Generic Group Model can also be used for proving the security of cryptographic constructions, such as signature schemes and algebraic MACs, against algebraic attacks. In this context, the adversary has access to oracles for performing signatures, verification, *etc.* The Generic Group Analyzer also provides support for such problems, but is inherently limited to oracles which do not take handles to group elements as inputs. This support can be used for analyzing simple interactive assumptions. Subsequent extensions of the Generic Group Analyzer overcome this limitation by providing support for oracles that take handles as inputs, and by allowing adversaries to make a bounded number of oracle queries [12]. Using this extension, Barthe et al. [12] synthesize (in the Type II-setting) structure-preserving signatures that are secure against adversaries that can make a bounded number of signing queries. Their approach is based on an algebraic characterization of security, using a vector space whose dimension increases by one for each query. Therefore, their approach is limited to a small number of queries, and an alternative approach must be used for proving security notions which do not impose a bound on the number of queries.

The first main contribution of this paper is to extend the Master Theorem to a general setting where adversaries can make arbitrarily many queries to oracles with group inputs, and where the winning conditions can be described using a rich language. As for simpler Master Theorems, our Master Theorem yields a sufficient condition for the security of cryptographic constructions. However, this simpler condition cannot be expressed in finite-dimensional linear algebra: informally, each adversarial query to an oracle taking group elements as inputs increases the dimension of the system to be analyzed, and therefore allowing arbitrarily many queries leads to a system that is not finite-dimensional. As a consequence, the algebraic approach of the Generic Group Analyzer cannot be used to analyze automatically sufficient conditions given by the Master Theorem.

The second main contribution of this paper is an automated method for proving the validity of these conditions, using a combination of methods from constraint solving, computer algebra, and symbolic cryptography. Building on

these two contributions, we implement an analyzer which subsumes the Generic Group Analyzer for interactive assumptions and is able to analyze many cryptographic constructions, including signatures and message authentication codes.

Technical Overview. In more detail, our contributions are as follows.

First, we define a language to express security experiments in the Generic Group Model where the adversary can make an unbounded number of queries to oracles; moreover, our model allows oracles to take group values as inputs. In addition, we define a rich language of winning conditions. We then establish a Master Theorem, which states that a generic algorithm is secure with respect to a security goal expressed using our language of winning conditions, if the constraint system extracted from the security experiment, given by the algorithm and the winning condition, has no computable solution. Informally, the notion of computable solution provides an algebraic counterpart to the notion of deducibility used in the symbolic (a.k.a. Dolev-Yao) approach to cryptography; more technically, this notion is based on an inductive definition of the adversary's knowledge throughout execution of the algorithm. From a broader perspective, our Master Theorem provides a novel light on the relationship between different cryptographic models, by showing a general relationship between the Generic Group Model and the symbolic model. Note that, for the sake of simplicity, we focus on group settings with bilinear pairings; however, we believe that our model and Master Theorem can naturally extend to the case of multilinear maps.

Second, we define an automated method for proving the absence of computable solutions of constraint systems. Our language of constraints supports algebraic expressions that are generally not considered by prior work on the symbolic model. Therefore, we cannot use previous constraint-solving methods developed for reasoning about cryptographic protocols in the symbolic model. Rather, we define a specialized method which combines general purpose algebraic computations and specialized steps. The algebraic computations are performed using Gröbner bases, whereas the specialized steps include simplifications related to big operators and case distinctions. The latter can be used to add new equations to constraint systems and thus to trigger new simplifications. Case distinctions are an essential ingredient for the success of our method: they yield compact proofs that follow the structure of pen-and-paper arguments found in the literature. Of course, the use of case distinctions is not new in automated deduction; it is at the core of Staalmarck's method, an empirically successful method for propositional logic. However, its use in our setting appears to be new.

Third, we implement our method and evaluate its effectiveness on a sizable set of case studies. Our tool uses off-the-shelf computer algebra systems to perform Gröbner bases computations. However, it draws its efficiency from a finely tuned heuristics for carrying case distinctions. We evaluate our tool on structure-preserving signatures, in all settings (Type I, Type II and Type III). Our tool is able to prove unbounded security of many structure-preserving signatures from the literature, as well as of the algebraic MACs from Chase, Meiklejohn and

Zaverucha [18], and of the short randomizable signatures from Pointcheval and Sanders [34]. Furthermore, it also proves unbounded security for most of the examples proved 2-time secure in [12] (these examples were generated automatically using synthesis techniques). Moreover, we also adapt the synthesis tool from [12] to generate structure-preserving signatures in the Type III setting and use our tool to prove security for more than a 100 such schemes.

Related Work. The Generic Group Model was introduced by Nechaev [33], Shoup [38] and Maurer [32], following distinct but equivalent approaches [29]. The original approach by Nechaev and Shoup lets the adversary access a randomly selected representation of group elements; in contrast, Maurer's approach requires the adversary to perform all algebraic operations via oracles, and uses handles as symbolic representations of group elements known to the adversary. We opt for the second approach, for its distinctively symbolic flavour. These works establish lower complexity bounds for the generic discrete logarithms and the generic hardness of Diffie-Hellman like assumptions. As for us, they use the Schwartz-Zippel Lemma for transforming their original problem into an algebraic one. This approach was extended by Boneh, Boyen and Goh [17]. First, their Generic Group Model focuses on bilinear groups. Second, they consider a general class of assumptions, and provide the first Master Theorem, which provides a systematic method for extracting algebraic conditions of security from assumptions. Their Master Theorem was subsequently extended in many directions. The most relevant works are those that involve the use of computer tools for verifying algebraic conditions. Notably, Freeman [23] verifies the hardness of two assumptions using Magma.

Shoup [38] and Schnorr and Jakobsson [36,37] were among the first to use the Generic Group Model for proving the security of crytographic constructions. Specifically, Shoup proves (generic) security of an identification scheme, whereas Schnorr and Jakobsson consider signed ElGamal encryption and blind discrete log signatures. More recently, the Generic Group Model has also become an important tool for analyzing the security of pairing-based cryptographic constructions. Chase, Meiklejohn and Zaverucha [18] propose a class of algebraic MACs and prove their generic security. Several authors use the Generic Group Model for proving the generic security of structure-preserving signatures [1]. Groth [26] proposes new fully-structure-preserving signatures [6] and proves their generic security. Similarly, Fuchsbauer, Hanser and Slamanig [25] define a structure-preserving signature on equivalence classes and prove its generic security. Furthermore, the Generic Group Model gives a convenient setting for establishing lower bounds on the complexity of structure-preserving signatures [2,4,5,12]. In a similar spirit, the Generic Group Model has been used for proving the correctness of translations of signature schemes from Type I to Type III [3,5,7].

It is also worth pointing to a recent examination of the efficiency of pairing-based implementations. Based on a practical evaluation of the efficiency of state-of-the-art implementations of pairings, Chatterjee and Menezes [19] argue that

Type III pairings are more efficient than their Type II counterparts, and should be favoured in implementations. Their observation justifies the need to transpose existing results and tools for the Type II setting to the Type III setting, and has motivated the application of our methods to the latter.

Several works have developed or used tools for reasoning about the Generic Group Model. As already mentioned, the Generic Group Analyzer [11] implements an automated method for analyzing assumptions. Moreover, a subsequent extension of the analyzer [12] supports the automated analysis of security of structure-preserving security against adversaries that make a bounded number of queries. In practice, the tool only terminates for small bounds on the number of queries. While these works are the most closely related to ours, there have been previous works that apply computer tools to the Generic Group Model. Barthe, Cederquist and Tarento [9,15] were the first to use formal verification tools for analyzing the security of hardness assumptions and cryptographic constructions in the Generic Group Model. Their work uses the Coq proof assistant, and provides no support for automation. Freeman [23] reports on using computer algebra systems to prove the validity of new hardness assumptions in the Generic Group Model. Beyond the Generic Group Model, there exist several tools for synthesizing constructions, such as encryption schemes, modes of operations, tweakable blockciphers, and structure-preserving signatures in the Type II setting [10,12,27,31], automated transformation of existing constructions, including signature schemes [3,7,8], and verification of security proofs [13,14,16]. In particular, [14] introduce AutoG&P, a highly automated framework for proving the security of pairing-based cryptographic primitives; the focus of [14] is on encryption schemes, but their methods are also applicable to signatures and MACs. AutoG&P and gga^{∞} are complementary in two different ways. First, gga^{∞} focuses on full automation in the Generic Group Model while AutoGP provides partial automation in the Standard model. Second, and more interestingly, some of our techniques for equational reasoning could be used to achieve more automation in AutoG&P, whereas it could be possible to use techniques from AutoG&P as a fallback solution when full automation fails in gga^{∞}.

2 Preliminaries

In this section, we give some background on bilinear groups and define the notation used throughout the paper.

2.1 Bilinear Groups

We consider bilinear groups $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t)$. For Type I, $\mathbb{G}_1 = \mathbb{G}_2$ and for Type II, there is an additional isomorphism $\Psi : \mathbb{G}_2 \to \mathbb{G}_1$. We use additive notation for all three groups and use P_1, P_2, P_t to denote their generators. For $a \in \mathbb{F}_p$, we use $[\![a]\!]_i$ to denote the implicit representation aP_i of a in $\mathbb{G}_i$ following [21].

2.2 Notation

We define $aS = \{as \mid s \in S\}$ and $SS' = \{ss' \mid s \in S \wedge s' \in S'\}$. For a set S, we write S^* to denote vectors of elements in S. We define $[n]$ as the range $\{1, \ldots, n\}$ for an arbitrary $n \in \mathbb{N}$. We use $\boldsymbol{v}$ to denote a vector and $\boldsymbol{v}_{(i)}$ to denote the i-th element. We assume given a set of *uniform variables* UVar, a set of *handle variables* $\mathsf{HVar} = \mathsf{HVar_1} \uplus \mathsf{HVar_2} \uplus \mathsf{HVar_t}$, a set of *parameter variables* PVar, and a set of *index variables* IVar. We use $ty(h) \in \{1, 2, \mathsf{t}\}$ to denote the type of a handle variable, i.e., $ty(h) = i$ iff $h \in \mathsf{HVar}_i$.

We use $R[\boldsymbol{X}^{\pm 1}]$ to denote the set of *Laurent polynomials* over the ring R with variables in $\boldsymbol{X}$. We also use the shorthand $R[\boldsymbol{Y}, \boldsymbol{X}^{\pm 1}]$ for $(R[\boldsymbol{Y}])[\boldsymbol{X}^{\pm 1}]$ to denote nested polynomial rings. We use a similar notation $Mon[\boldsymbol{X}^{\pm 1}, \boldsymbol{Y}]$ for *Laurent monomials*. We write $\deg_V(M)$ to denote the *degree* of V in the Laurent monomial M. We write $coeff_M(F)$ to denote the coefficient of the Laurent monomial M in the Laurent polynomial F.

For a term t possibly containing variables, we write $t[x \mapsto t']$ to denote the result of substituting all occurrences of the variable x in t with t'. A context C is a term with a distinguished variable $\square$ which denotes a hole that can be filled in by an arbitrary term. We assume the hole occurs exactly once in a context. We use $C[t]$ to denote the term obtained by plugging t into $C's$ hole.

3 Translating Security Experiments into Constraints

In this section, we first present a language to define security experiments in the Generic Group Model. Next, we define the language of winning constraints. Winning constraints are formulas that characterize if an adversary can win a security experiment. Finally, we present a translation procedure from security experiments to winning constraints.

3.1 Security Experiment Definition

We first present the language that we use to define security experiments. Afterwards, we define the corresponding games in the Generic Group Model and the symbolic group model (see [11]). We will exploit that the generic and symbolic games are indistinguishable and use the symbolic game to perform our analysis.

Definition 1 *(Security experiment). A* security experiment *is defined by a tuple* $SE = (t, ainp, odef, wcond)$ *where*

- *the* group type *is defined by* $t \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$,
- *the* adversary input *is defined by* $ainp = (\boldsymbol{X}, (\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_t}))$ *for*
  - global uniform variables $\boldsymbol{X} \in \mathsf{UVar}^*$ *and*
  - input polynomials $\boldsymbol{F_i} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}]^*$,
- *the* oracle *is defined by* $odef = (\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{R}, (\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_t}))$ *for*
  - arguments $\boldsymbol{a} \in \mathsf{PVar}^*$ *and* oracle handles $\boldsymbol{h} \in \mathsf{HVar}^*$,[1]

[1] Handle variables are typed, i.e., for all $j \in [|\boldsymbol{h}|]$, it holds that $ty(\boldsymbol{h}_{(j)}) \in \{1, 2, \mathsf{t}\}$.

  - oracle uniform variables $\boldsymbol{R} \in \mathsf{UVar}^*$, *and*
  - oracle polynomials $\boldsymbol{H_i} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, \boldsymbol{R}^{\pm 1}, \boldsymbol{a}, \boldsymbol{h}]^*$, *and*
- *the* winning condition *is defined by* $wcond = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{H}}, \boldsymbol{W}^{=}, \boldsymbol{W}^{\neq})$ *for*
  - winning arguments $\hat{\boldsymbol{a}} \in \mathsf{PVar}^*$ *and* winning handles $\hat{\boldsymbol{h}} \in \mathsf{HVar}^*$, *and*
  - winning (in)equalities $\boldsymbol{W}^{=}, \boldsymbol{W}^{\neq} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, \boldsymbol{R}^{\pm 1}, \boldsymbol{a}, \boldsymbol{h}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{h}}]^*$.

Intuitively, the *adversary input* represents the values given initially to the adversary. This usually includes the public parameters and the public keys. The *oracle* is defined by *arguments* and *oracle handles* that represent the oracle input; *uniform variables* that denote randomness sampled by the oracle; and *oracle polynomials* that denote the oracle response. Finally, the *winning condition* is defined by *winning arguments* that represent the forgery that the adversary must produce; and *winning (in)equalities* that characterize valid forgeries.

We define the corresponding generic group game $\mathsf{G}^{\mathsf{gen}}(SE)$ as follows:

1. Sample the vector $\boldsymbol{x} \in (\mathbb{F}_p^{\times})^{|\boldsymbol{X}|}$, compute the adversary inputs $[\![\boldsymbol{F_i}(\boldsymbol{x})]\!]_i \in \mathbb{G}_i^{|\boldsymbol{F_i}|}$ (for $i \in \{1, 2, \mathsf{t}\}$), and call the adversary $\mathcal{A}$ with the corresponding handles.
2. The adversary $\mathcal{A}$ can perform q_g queries to perform group operations (for group type t), an unbounded number of equality queries, and q queries to an oracle that implements *odef*. The oracle for *odef* takes scalars $\boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|}$ for $\boldsymbol{a}$ and a vector of handles to group elements $\boldsymbol{U}$ for $\boldsymbol{h}$. We use $\boldsymbol{u}$ to denote the discrete logarithms of $\boldsymbol{U}$, i.e., for all $j \in [|\boldsymbol{h}|]$, $[\![\boldsymbol{u}_{(j)}]\!]_i = \boldsymbol{U}_{(j)}$ where $i = ty(\boldsymbol{h}_{(j)})$. Then it samples $\boldsymbol{r} \in (\mathbb{F}_p^{\times})^{|\boldsymbol{R}|}$ and returns handles to $[\![\boldsymbol{H_i}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{r})]\!]_i \in \mathbb{G}_i^{|\boldsymbol{H_i}|}$. We use $\boldsymbol{v}^{(j)}$, $\boldsymbol{u}^{(j)}$, $\boldsymbol{r}^{(j)}$ to denote the corresponding values used in the j-th query.
3. The adversary $\mathcal{A}$ returns scalars $\hat{\boldsymbol{v}} \in \mathbb{F}_p^{|\hat{\boldsymbol{a}}|}$ for $\hat{\boldsymbol{a}}$ and handles to group elements $\hat{\boldsymbol{U}}$ for $\hat{\boldsymbol{h}}$. Again, we denote the discrete logarithms of $\hat{\boldsymbol{U}}$ with $\hat{\boldsymbol{u}}$. The adversary wins if for $\bowtie \in \{=, \neq\}$, $w \in \boldsymbol{W}^{\bowtie}$, and $j \in [q]$, it holds that $w(\boldsymbol{x}, \boldsymbol{r}^{(j)}, \boldsymbol{v}^{(j)}, \boldsymbol{u}^{(j)}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{u}}) \bowtie 0$.

Note that additional care must be taken to ensure that the oracles and winning conditions are efficiently computable using scalar multiplication, addition, application of isomorphisms, and application of bilinear maps. For example, it is possible to specify an oracle that takes a handle to an element $[\![v]\!]_{\mathsf{t}} \in \mathbb{G}_{\mathsf{t}}$ and returns $[\![v]\!]_1 \in \mathbb{G}_1$, which cannot be efficiently computed in most bilinear groups of interest.

The symbolic game $\mathsf{G}^{\mathsf{sym}}(SE)$ is defined similarly, but internally uses Laurent polynomials $f(\boldsymbol{X})$ instead of group elements $[\![f(\boldsymbol{x})]\!]_i$. It is completely deterministic since it uses formal variables $\boldsymbol{X}$ to represent the initially sampled values and indexed formal variables $\boldsymbol{R}^{(j)}$ to represent the values sampled in the oracle.

Formally, we define $\mathsf{G}^{\mathsf{sym}}(SE)$ as follows:

1. Store the polynomials $\boldsymbol{F_i}(\boldsymbol{X}) \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}]^{|\boldsymbol{F_i}|}$ in the list for the group $\mathbb{G}_i$ (for $i \in \{1, 2, \mathsf{t}\}$) and call the adversary $\mathcal{A}$ with the corresponding handles.

2. The oracles for group operations and equality checks provide the same interface as in the generic model, but perform all computations in the ring of Laurent polynomials. The oracle for *odef* takes (in the j-th query) scalars $\boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|}$ for $\boldsymbol{a}$ and handles to polynomials

$$\boldsymbol{u} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, (\boldsymbol{R}^{(1)})^{\pm 1}, \ldots, (\boldsymbol{R}^{(j-1)})^{\pm 1}]^{|\boldsymbol{h_i}|}$$

for $\boldsymbol{h}$. It returns handles to polynomials

$$\boldsymbol{H_i}(\boldsymbol{X}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{R^{(j)}}) \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, (\boldsymbol{R}^{(1)})^{\pm 1}, \ldots, (\boldsymbol{R}^{(j)})^{\pm 1}]^{|\boldsymbol{H_i}|}.$$

3. The adversary $\mathcal{A}$ returns scalars $\hat{\boldsymbol{v}} \in \mathbb{F}_p^{|\hat{\boldsymbol{a}}|}$ for $\hat{\boldsymbol{a}}$ and handles to polynomials

$$\hat{\boldsymbol{u}} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, (\boldsymbol{R}^{(1)})^{\pm 1}, \ldots, (\boldsymbol{R}^{(q)})^{\pm 1}]^{|\hat{\boldsymbol{h}}_i|}$$

for $\hat{\boldsymbol{h}}$. He wins if for $\bowtie \in \{=, \neq\}$, $w \in \boldsymbol{W}^{\bowtie}$, and $j \in [q]$, it holds that $w(\boldsymbol{X}, \boldsymbol{R}^{(j)}, \boldsymbol{v}^{(j)}, \boldsymbol{u}^{(j)}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{u}}) \bowtie 0$.

Setup $\mathcal{P}(1^\lambda)$: Return $PP = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{t}, e) \leftarrow \mathcal{G}(1^\lambda)$ where $\mathcal{G}$ is a polynomial time algorithm that on input 1^λ returns a description of a bilinear map in the Type III setting with groups of order p for a λ-bit prime p.

Key generation $\mathcal{K}(PP)$:
Choose $v, w \leftarrow \mathbb{F}_p^\times$ and compute $VK = (PP, V, W)$ and $SK = (PP, v, w)$ as

$$V \leftarrow [\![v]\!]_1 \text{ and } W \leftarrow [\![w]\!]_1 .$$

Signing $\mathcal{S}_{SK}(M)$:
For $M = [\![m]\!]_2 \in \mathbb{G}_2$ choose $r \leftarrow \mathbb{F}_p^\times$ and compute the signature (T_1, T_2, S) as

$$T_1 \leftarrow [\![r]\!]_1 , \; T_2 = [\![r]\!]_2 , \text{ and } S \leftarrow [\![m\, v + w + r^2]\!]_2 .$$

Verification $\mathcal{V}_{VK}(M, S)$:
Accept if and only if $T_1 \in \mathbb{G}_1$, $M, T_2, S \in \mathbb{G}_2$,

$$e([\![1]\!]_1 , S) = e(V, M) + e(W, [\![1]\!]_2) + e(T_1, T_2), \text{ and } e(T_1, [\![1]\!]_2) = e([\![1]\!]_1 , T_2).$$

Fig. 1. SPS-scheme from [19] in Type III setting.

Example 1. We can formalize the EUF-CMA security of the scheme in Fig. 1 using the security experiment $SE = (t, ainp, odef, wcond)$ defined as follows:

- the group type is $t = \mathbb{III}$
- the adversary input is $ainp = (\boldsymbol{X}, (\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_t}))$ where
  - $\boldsymbol{X} = (v, w)$ (for $v, w \in \mathsf{UVar}$), $\boldsymbol{F_1} = (1, v, w)$, $\boldsymbol{F_2} = (1)$, $\boldsymbol{F_t} = (1)$
- the *oracle* is $odef = (\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{R}, (\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_t}))$ where

- $\boldsymbol{a} = ()$, $\boldsymbol{h} = (m)$ (for $m \in \mathsf{HVar}_2$),
- $\boldsymbol{R} = (r)$ (for $r \in \mathsf{UVar}$)
- $\boldsymbol{H_1} = (r)$, $\boldsymbol{H_2} = (r,\ mv + w + r^2)$, $\boldsymbol{H_t} = ()$

– the winning condition is $wcond = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{h}}, \boldsymbol{W}^{=}, \boldsymbol{W}^{\neq})$ where

- $\hat{\boldsymbol{a}} = ()$, $\hat{\boldsymbol{h}} = (\hat{m}, \hat{t}_1, \hat{t}_2, \hat{s})$ and (for $\hat{t}_1 \in \mathsf{HVar_1}$ $\hat{m}, \hat{t}_2, \hat{s} \in \mathsf{HVar}_2$),
- $\boldsymbol{W}^{=} = (\hat{s} - \hat{m}v - w - \hat{t}_1\hat{t}_2,\ \hat{t}_1 - \hat{t}_2)$, $\boldsymbol{W}^{\neq} = (\hat{m} - m^{(j)})$ ■

3.2 Winning Constraints

We first define the language of winning constraints, a class of formulas that can be used to characterize if an adversary can win the symbolic game $\mathsf{G}^{\mathsf{sym}}(SE)$. Then we define the set of solutions of a winning constraint and present a set of simplication rules that preserve the set of solutions.

$\mathcal{C} ::= \exists i \notin K.\mathcal{C} \mid \mathcal{C}$	constraint
$\mathcal{C}' ::= \mathcal{C}' \wedge \mathcal{C}' \mid \forall k \notin K.\mathcal{C}' \mid \mathcal{E} = 0 \mid \mathcal{E} \neq 0$	non-existential constraint
$\mathcal{E} ::= \mathcal{E} + \mathcal{E} \mid \mathcal{E} * \mathcal{E} \mid -\mathcal{E} \mid \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})$	expression
$\mid \sum_{k \notin K} \mathcal{E} \mid \mathcal{R} \mid \mathcal{R}^{-1} \mid \mathcal{P} \mid \mathcal{V} \mid 1 \mid 0$	
$\mathcal{M} ::= \mathcal{M} * \mathcal{M} \mid \mathcal{R} \mid \mathcal{R}^{-1} \mid 1$	monomial over uniform variables
$\mathcal{R} ::= R_{[k]} \mid R$	(indexed) uniform variable ($R \in \mathsf{UVar}$)
$\mathcal{P} ::= \rho_{[k]} \mid \rho$	(indexed) parameter ($\rho \in \mathsf{PVar}$)
$\mathcal{V} ::= Y_{[k]}$	indexed handle variable ($Y \in \mathsf{HVar}$)

Fig. 2. Grammar for winning constraints (for $k \in \mathsf{IVar}, K \subset \mathsf{IVar}$). For every $\mathsf{Coeff}(\mathcal{E})$, $\mathcal{E}$ does not contain the symbol Coeff.

Definition 2 *(Winning constraints). The language of winning constraints is defined by the grammar given in Fig. 2. We distinguish between* bound index variables *and* free index variables *depending on whether they are bound by $\forall/\Sigma$. We write $ivars(\mathcal{C})$ to denote the free index variables in the constraint $\mathcal{C}$.*

Intuitively, atomic constraints $\mathcal{E} = 0$ represent polynomial equalities. In the quantifications $\forall k \notin K$ and $\sum_{k \notin K}$, the index variable k ranges over all elements in $[q]$ except for the valuations of the index variables in K. Uniform variables $R/R_{[k]}$ are treated like formal variables, parameters $\rho/\rho_{[k]}$ can be instantiated with integers, handle variables $Y_{[k]}$ can be instantiated with Laurent polynomials over uniform variables, and the arithmetic operations are interpreted in the ring

$$eval_s(c) = \begin{cases} eval_{s_{k,K,1}}(c') \vee \ldots \vee eval_{s_{k,K,q-|K|}}(c') & \text{for } c = \exists k \notin K.\, c' \\ eval_{s_{k,K,1}}(c') \wedge \ldots \wedge eval_{s_{k,K,q-|K|}}(c') & \text{for } c = \forall k \notin K.\, c' \\ eval_s(c_1) \otimes eval_s(c_2) & \text{for } c = c_1 \otimes c_2 \\ eval_{s_{k,K,1}}(c') + \ldots + eval_{s_{k,K,q-|K|}}(c') & \text{for } c = \sum_{k \notin K} c' \\ \delta(\rho) & \text{for } c = \rho \\ \chi(\rho, \sigma(i)) & \text{for } c = \rho_{[i]} \\ \xi(Y, \sigma(i)) & \text{for } c = Y_{[i]} \\ R^e & \text{for } c = R^e, e \in \{+1, -1\} \\ R^e_{[\sigma(i)]} & \text{for } c = R^e_{[i]}, e \in \{+1, -1\} \\ coeff_{\sigma(\mathcal{M})}(eval_s(\mathcal{E})) & \text{for } c = \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) \\ c & \text{for } c \in \{0, 1\} \end{cases}$$

Fig. 3. Definition of the evaluation function $eval_s$ for $s = (p, q, \sigma, \delta, \chi, \xi)$, where $R \in \mathsf{UVar}$, $\otimes \in \{=, \neq, \wedge, *, +\}$ are interpreted as the corresponding boolean operations/arithmetic operations in the ring of Laurent polynomials over $\mathbb{F}_p$ and $s_{k,K,i}$ defined as follows. Let $\{v_1, \ldots v_{q-|K|}\} = [q] \setminus \sigma(K)$, then $s_{k,K,i} = (p, q, \sigma', \delta, \chi, \xi)$ where $\sigma' = \sigma[k \mapsto v_i]$ for $i \in \{1, \ldots, q - |K|\}$.

of Laurent polynomials over $\mathbb{F}_p$ for a prime p. An expression $\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})$ represents the coefficient of the monomial $\mathcal{M}$ in the expression $\mathcal{E}$ after the parameters and handle variables in $\mathcal{E}$ are instantiated. The resulting Laurent polynomial after instantiation contains only (indexed) uniform variables. Formally, the set of solutions of a winning constraint is defined as follows.

Definition 3 *(Solutions of winning constraints). A* structure $s = (p, q, \sigma, \delta, \chi, \xi)$ *for a prime number p, a natural number q, a valuation $\sigma : \mathsf{IVar} \to [q]$ for (free) index variables, valuations $\delta : \mathsf{PVar} \to \mathbb{F}_p$ and $\chi : \mathsf{PVar} \times [q] \to \mathbb{F}_p$ for the parameters, and a valuation $\xi : \mathsf{HVar} \times [q] \to \mathbb{F}_p[\mathsf{UVar}^{\pm 1}, \mathsf{UVar}^{\pm 1}_{[1]}, \ldots, \mathsf{UVar}^{\pm 1}_{[q]}]$ for the handle variables is a* solution *for a winning constraint $\mathcal{C}$ if $eval_s(\mathcal{C}) = true$ for the function eval defined in Fig. 3.*

3.3 Translation from Security Experiments to Winning Constraints

We define the translation function to convert a security experiment definition into winning constraints. The translation is sound and complete with respect to a certain class of solutions. Roughly, this means that there is an efficient attacker[2] on the security experiment in the Generic Group Model with non-negligible winning probability iff there is a solution for the translated winning constraints where handle variables are instantiated with "computable" Laurent polynomials.

To simplify the presentation, we assume that for all security experiments in Type II, it holds that $\boldsymbol{F_2} \subseteq \boldsymbol{F_1}$ and $\boldsymbol{H_2} \subseteq \boldsymbol{H_1}$ which allows us to ignore the

[2] More precisely, an attacker that performs a polynomial number of queries q_g and q.

isomorphism Ψ. Similarly, we assume for Type I that $\boldsymbol{F_1} = \boldsymbol{F_2}$ and $\boldsymbol{H_1} = \boldsymbol{H_2}$ which allows us to ignore that $\mathbb{G}_1 = \mathbb{G}_2$.

First, note that $\boldsymbol{W}^{\bowtie} \subset \mathbb{Z}[\boldsymbol{X}^{\pm 1}, \boldsymbol{R}^{\pm 1}, \boldsymbol{a}, \boldsymbol{h}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{h}}]$ where $\boldsymbol{X}, \boldsymbol{R} \in \mathsf{UVar}^*$, $\boldsymbol{a}, \hat{\boldsymbol{a}} \in \mathsf{PVar}^*$, and $\boldsymbol{h}, \hat{\boldsymbol{h}} \in \mathsf{HVar}^*$. For an index variable $j \in \mathsf{IVar}$, we write $\boldsymbol{R}_{[j]}$ to denote the vector $(\boldsymbol{R}_{(1)[j]}, \ldots, \boldsymbol{R}_{(|\boldsymbol{R}|)[j]})$ of indexed uniform variables. Similarly, we write $\boldsymbol{a}_{[j]}$ and $\boldsymbol{h}_{[j]}$. For our translation, we instantiante each winning handle variable $\hat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar_1} \cup \mathsf{HVar_2}$ with a linear combination of polynomials in the adversary input and in the oracle output. Formally, we define the vector $\boldsymbol{E}$ of expressions as follows. For $u \in [|\hat{\boldsymbol{h}}|]$ such that $\hat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar_1}$ and $l = |\boldsymbol{H_1}|$, we define

$$\begin{aligned}\boldsymbol{E}_{(u)} = {} & \rho^{(1,u,1)} \boldsymbol{F_1}_{(1)}(\boldsymbol{X}) + \ldots + \rho^{(1,u,|\boldsymbol{F_1}|)} \boldsymbol{F_1}_{(|\boldsymbol{F_1}|)}(\boldsymbol{X}) + \\ & \sum_k \tau_{[k]}^{(1,u,1)} \boldsymbol{H_1}_{(1)}(\boldsymbol{X}, \boldsymbol{R}_{[k]}, \boldsymbol{a}_{[k]}, \boldsymbol{h}_{[k]}) + \ldots + \\ & \sum_k \tau_{[k]}^{(1,u,l)} \boldsymbol{H_1}_{(l)}(\boldsymbol{X}, \boldsymbol{R}_{[k]}, \boldsymbol{a}_{[k]}, \boldsymbol{h}_{[k]})\end{aligned}$$

where $\rho^{(1,u,n)}$ and $\tau^{(1,u,n)}$ are distinct fresh parameter variables. For $u \in [|\hat{\boldsymbol{h}}|]$ such that $\hat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar_2}$, we define $\boldsymbol{E}_{(u)}$ analogously. For $u \in [|\hat{\boldsymbol{h}}|]$ such that $\hat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar_t}$, we define $\boldsymbol{E}_{(u)}$ analogously additionally taking products of polynomials from $\mathbb{G}_1$ and $\mathbb{G}_2$ into account. We define the winning constraint derived from SE as

$$toConstr(SE) = \bigwedge_{w \in \boldsymbol{W}^{\bowtie}} \forall j. \left(w(\boldsymbol{X}, \boldsymbol{R}_{[j]}, \boldsymbol{a}_{[j]}, \boldsymbol{h}_{[j]}, \hat{\boldsymbol{a}}, \boldsymbol{E}) \bowtie 0\right).$$

A priori, the notion of solution for winning constraints does not restrict the set of Laurent polynomials that can be used to instantiate the handle variables in $\boldsymbol{h}_{[j]}$. Since we are only interested in solutions where the instantiations of handle variables are computable, we now define the notion of constrained solution.

Definition 4 *(Constrained solutions of winning constraints). A solution is* constrained *by sequences of sets $\{K_j^{(i)}\}_{j \in \mathbb{N}}$ of Laurent polynomials (for $i \in \{1, 2, \mathsf{t}\}$) if for all $i \in \{1, 2, \mathsf{t}\}$, $Y \in \mathsf{HVar}_i$, and $j \in [q]$, it holds that $\xi(Y, j) \in K_j^{(i)}$.*

Since we are interested in solutions constrained by computable Laurent polynomials, we next define the sequences of computable polynomials. We use $\langle S \rangle$ to denote the vector space over $\mathbb{F}_p$ generated by S.

Definition 5 *(Computable polynomials). The sequences of* computable polynomials *for a security experiment*

$$SE = (t, \boldsymbol{X}, (\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_t})), (\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{R}, (\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_t})), wcond)$$

are defined as follows:

$$\begin{aligned}\mathcal{K}_0^{SE,(i)} &= \langle toSet(\boldsymbol{F_i}) \rangle && \text{for } i \in \{1, 2\} \\ \mathcal{K}_0^{SE,(\mathsf{t})} &= \langle toSet(\boldsymbol{F_t}) \cup (\mathcal{K}_0^{SE,(1)} * \mathcal{K}_0^{SE,(2)}) \rangle\end{aligned}$$

$$\begin{aligned}
\mathcal{K}_{j+1}^{SE,(i)} &= \langle \mathcal{K}_j^{SE,(i)} \cup && \text{for } j \geq 0, i \in \{1,2\} \\
&\quad \{H(\boldsymbol{X}, \boldsymbol{v}, \boldsymbol{E}, \boldsymbol{R}^{(j+1)}) \mid H \in \boldsymbol{H}_i \wedge \\
&\quad\quad \boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|} \wedge |\boldsymbol{E}| = |\boldsymbol{h}| \wedge \boldsymbol{E}_{(u)} \in \mathcal{K}_j^{SE,(ty(\boldsymbol{h}_{(u)}))}\}\rangle \\
\mathcal{K}_{j+1}^{SE,(\mathsf{t})} &= \langle \mathcal{K}_j^{SE,(\mathsf{t})} \cup (\mathcal{K}_{j+1}^{SE,(1)} * \mathcal{K}_{j+1}^{SE,(2)}) \cup && \text{for } j \geq 0 \\
&\quad \{H(\boldsymbol{X}, \boldsymbol{v}, \boldsymbol{E}, \boldsymbol{R}^{(j+1)}) \mid H \in \boldsymbol{H}_{\mathsf{t}} \wedge \\
&\quad\quad \boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|} \wedge |\boldsymbol{E}| = |\boldsymbol{h}| \wedge \boldsymbol{E}_{(u)} \in \mathcal{K}_j^{SE,(ty(\boldsymbol{h}_{(u)}))}\}\rangle
\end{aligned}$$

The definition is always valid for Type III*. For Types* I *and* II*, it is valid under the previously stated assumptions on* $\boldsymbol{F_i}$ *and* $\boldsymbol{H_i}$*. We say a solution* s *is an* SE-computable solution *if it is constrained by* $(\mathcal{K}_j^{SE,(i)})_{j,i}$*.*

Theorem 1 *(Soundness and Completeness of Translation). Let* $p \approx 2^\lambda$ *and* q_g, q *polynomial in* λ*. Then the winning probability in the generic group game* $\mathsf{G}^{\mathsf{gen}}(SE)$ *with a group of order* p *is negligible in* λ *for all adversaries that perform at most* q_g *(resp.* q*) queries iff there is no* SE*-computable solution for* $toConstr(SE)$*.*

Proof (Sketch). For all concrete values of q_g, q, and SE we can use the master theorem for interactive assumptions from [11] (more precisely, the extended version for handles from [22]) to obtain an algebraic criterion that is equivalent to the security of the construction. By unfolding the definitions of *toConstr* and *eval*, we can verify that the criterion is true for all bounds on the number of oracle-queries iff there is no SE-computable solution for $toConstr(SE)$. □

Example 2. The translation of the security experiment for the example in Fig. 1 to winning constraints is

$$\hat{S} - \hat{M} * V - W - \hat{T}_1 * \hat{T}_2 = 0 \quad \wedge \quad \hat{T}_1 - \hat{T}_2 = 0 \quad \wedge \quad \forall k.\ \hat{M} - M_{[k]} \neq 0$$

where $V, W, R \in \mathsf{UVar}$ and $M \in \mathsf{HVar}_2$, $\mu, \mu', \mu'', \rho, \rho', \rho'', \rho''', \tau, \tau', \tau'', \gamma, \gamma', \gamma'' \in \mathsf{PVar}$, and $\hat{M}, \hat{S}_1, \hat{S}_2, \hat{S}_3$ are defined as

$$\begin{aligned}
\hat{M} &= \mu + \sum_k \mu'_{[k]} * R_{[k]} + \sum_k \mu''_{[k]} * (M_{[k]} * V + W + R_{[k]}^2), \\
\hat{T}_1 &= \rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W, \\
\hat{T}_2 &= \tau + \sum_k \tau'_{[k]} * R_{[k]} + \sum_k \tau''_{[k]} * (M_{[k]} * V + W + R_{[k]}^2), \text{ and} \\
\hat{S} &= \gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + W + R_{[k]}^2).
\end{aligned}$$

We first outline the sequence of computable monomials for $\mathbb{G}_1$:

$$\begin{aligned}
\mathcal{K}_0^{SE,(2)} &= \langle 1, V, W \rangle \\
\mathcal{K}_1^{SE,(2)} &= \langle \mathcal{K}_0^{SE,(2)} \cup \{R_{[1]}\} \rangle \\
\mathcal{K}_2^{SE,(2)} &= \langle \mathcal{K}_1^{SE,(2)} \cup \{R_{[2]}\} \rangle \\
&\dots
\end{aligned}$$

For $\mathbb{G}_2$, the sequence looks as follows:

$$\begin{aligned}
\mathcal{K}_0^{SE,(2)} &= \langle 1 \rangle \\
\mathcal{K}_1^{SE,(2)} &= \langle \mathcal{K}_0^{SE,(2)} \cup \{1, R_{[1]}, \overbrace{V + W + R_{[1]}^2}^{:=f_1}\} \rangle \\
\mathcal{K}_2^{SE,(2)} &= \langle \mathcal{K}_1^{SE,(2)} \cup \{R_{[2]}, R_{[1]} * V + W + R_{[2]}^2, f_1 * V + W + R_{[2]}^2\} \rangle \\
&\dots
\end{aligned}$$

For $\mathbb{G}_t$, only the first line of the definition (computable earlier or product of computable in $\mathbb{G}_1$ and computable in $\mathbb{G}_2$) is non-empty. ■

4 Constraint Solving

In this section, we define an algorithm that takes a winning constraint and tries to derive a contradiction thereby showing that the winning constraint has no solution. Our algorithm uses constraint solving rules to perform a complete search for solutions using simplification rules and case distinctions. We first give the rules and then describe a strategy to apply the rules in Sect. 5. We begin by describing a set of simplification rules for constraints that exploit logical equivalences to bring a constraint into a simplified form. Next, we describe a set of rules for introducing and simplifying Coeff constraints. Then, we describe our rules for performing case distinctions followed by describing a procedure for equational simplification based on Gröbner Basis techniques. We conclude by giving a worked out example.

4.1 Constraint Solving Rules and Soundness

We use the notation $\mathcal{C} \rightsquigarrow_{SE} \mathcal{C}_1 \vee \ldots \vee \mathcal{C}_k$ to denote the constraint solving rule that "simplifies" the constraint $\mathcal{C}$ into the disjunction of constraints $\mathcal{C}_1, \ldots, \mathcal{C}_k$. The constraint solving rule might depend on the security experiment SE. Our rules are sound in the following sense: If there exists an SE-solution s for $\mathcal{C}$, then there is an $i \in \{1, \ldots, k\}$ such that there exists an SE-solution s' for $\mathcal{C}_i$. The solution s' is usually very similar to s, but might, for example, perform an additional query with trivial parameters. We use $\mathcal{C} \rightsquigarrow_{SE} \bot$ to denote that $\mathcal{C}$ can be simplified to the empty disjunction, which is equivalent to false.

We say a constraint $\mathcal{C}$ is contradictory if there is either a rule $\mathcal{C} \rightsquigarrow_{SE} \bot$ or there is a rule $\mathcal{C} \rightsquigarrow_{SE} \mathcal{C}_1 \vee \ldots \vee \mathcal{C}_k$ such that for all $i \in \{1, \ldots, k\}$, the constraint $\mathcal{C}_i$ is contradictory. Since all rules are sound, we obtain that if $\mathcal{C}$ is contradictory, then $\mathcal{C}$ has no solution.

4.2 Simplification Rules

To exploit the equivalence $e = e'$ given in Fig. 5, we define a corresponding constraint solving rule $C[e] \rightsquigarrow_{SE} C[e']$ for each of them. The rules up to and including the equivalences for Coeff can be used to bring every winning constraint into simplified form (see Fig. 4). Additionally, we assume given rules for the axioms of commutative rings with respect to $0, 1, *$ and $+$.

The remaining rules are useful to enable the application of other rules. The first remaining set of rules allows to swap binders, which might be required before applying rules that expect a certain binder to be in outermost position. To preserve the well-formedness of constraints, we adapt the index exception sets K as shown below. The second remaining set of rules allows us to add exceptions to binders. This might also benefit the applicability of other rules.

4.3 Introducing and Simplifying Coeff Constraints

In this section, we describe how to introduce and simplify constraints that involve Coeff expressions. To define our constraint solving rules, we define three functions that filter variables in monomials.

The functions

- $umon : Mon[\mathsf{UVar}^{\pm 1}, \mathsf{HVar}, \mathsf{PVar}] \to Mon[\mathsf{UVar}^{\pm 1}]$,
- $hmon : Mon[\mathsf{UVar}^{\pm 1}, \mathsf{HVar}, \mathsf{PVar}] \to Mon[\mathsf{HVar}]$, and
- $pmon : Mon[\mathsf{UVar}^{\pm 1}, \mathsf{HVar}, \mathsf{PVar}] \to Mon[\mathsf{PVar}]$.

keep the exponents for the desired type of variables and set the exponents of all other variables to zero.

$\mathcal{C}_{simp}$	$::= \exists k \notin K. \mathcal{C}_{simp} \mid \mathcal{C}^{\wedge}$	existential quantification
$\mathcal{C}^{\wedge}$	$::= \mathcal{C}^{\forall} \wedge \ldots \wedge \mathcal{C}^{\forall}$	conjunction
$\mathcal{C}^{\forall}$	$::= \forall k \notin K. \mathcal{C}^{\forall} \mid \mathcal{C}^{eq}$	universal quantification
$\mathcal{C}^{eq}$	$::= \mathcal{E}^{+} = 0 \mid \mathcal{E}^{+} \neq 0$	(in)equality
$\mathcal{E}^{+}$	$::= \mathcal{E}^{\Sigma} + \ldots + \mathcal{E}^{\Sigma} \mid 0$	sum
$\mathcal{E}^{\Sigma}$	$::= \sum_{k \notin K} \mathcal{E}^{\Sigma} \mid -\mathcal{E}^{*} \mid \mathcal{E}^{*} \mid \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}^{*})$	symbolic sum
$\mathcal{M}$	$::= \mathcal{M} * \mathcal{M} \mid R^{\pm 1} \mid R^{\pm 1}_{[k]} \mid 1$	monomial over uniform variables
$\mathcal{E}^{*}$	$::= \mathcal{E}^{pv} * \ldots * \mathcal{E}^{pv} \mid 1$	monomials
$\mathcal{E}^{pv}$	$::= \rho_{[k]} \mid \rho \mid R^{\pm 1} \mid R^{\pm 1}_{[k]} \mid Y_{[k]}$	parameter/variable

Fig. 4. Grammar for simplified winning constraints where $\rho \in \mathsf{PVar}, R \in \mathsf{UVar}, Y \in \mathsf{HVar}, k \in \mathsf{IVar}$. Conjunctions, sums, and products cannot by empty, but they can have a single argument. All bound variables must occur in the body. A monomial never contains a uniform variable and its inverse and never contains 1 unless it is equal to 1.

$$(\forall k \notin K. \mathcal{C}_1 \wedge \mathcal{C}_2) = (\forall k \notin K. \mathcal{C}_1) \wedge (\forall k \notin K. \mathcal{C}_2) \quad \text{(equiv-1)}$$

$$(\forall k \notin K. \mathcal{C}_1) = \mathcal{C}_1 \qquad \text{if } k \notin \mathit{ivars}(\mathcal{C}_1) \quad \text{(equiv-2)}$$

$$\mathcal{E}_1 * \mathcal{E}_2 = \mathcal{E}_2 * \mathcal{E}_1 \quad \text{(equiv-3)}$$

$$-(\mathcal{E}_1 + \mathcal{E}_2) = (-\mathcal{E}_1) + (-\mathcal{E}_2) \quad \text{(equiv-4)}$$

$$\sum_{k \notin K} (\mathcal{E}_1 + \mathcal{E}_2) = (\sum_{k \notin K} \mathcal{E}_1) + (\sum_{k \notin K} \mathcal{E}_2) \quad \text{(equiv-5)}$$

$$(\sum_{k \notin K} \mathcal{E}_1) * \mathcal{E}_2 = (\sum_{k \notin K} \mathcal{E}_1 * \mathcal{E}_2) \quad \text{(equiv-6)}$$

$$-(\sum_{k \notin K} \mathcal{E}) = (\sum_{k \notin K} -\mathcal{E}) \quad \text{(equiv-7)}$$

$$((-\mathcal{E}_1) * \mathcal{E}_2) = -(\mathcal{E}_1 * \mathcal{E}_2) \quad \text{(equiv-8)}$$

$$-(-\mathcal{E}) = \mathcal{E} \quad \text{(equiv-9)}$$

$$\mathcal{R} * \mathcal{R}^{-1} = 1 \quad \text{(equiv-10)}$$

$$\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}_1 + \mathcal{E}_2) = \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}_1) + \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}_2) \quad \text{(equiv-11)}$$

$$\mathsf{Coeff}_{\mathcal{M}}(\sum_{k \notin K} \mathcal{E}) = \sum_{k \notin K} \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) \qquad \text{if } \mathit{ivars}(\mathcal{M}) \subseteq K \quad \text{(equiv-12)}$$

$$\mathsf{Coeff}_{\mathcal{M}}(-\mathcal{E}) = -\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) \quad \text{(equiv-13)}$$

$$\exists k_1 \notin K_1. \exists k_2 \notin K_2. \mathcal{C} = \exists k_2 \notin K_2'. \exists k_1 \notin K_1'. \mathcal{C} \quad \text{(swap-1)}$$

$$\forall k_1 \notin K_1. \forall k_2 \notin K_2. \mathcal{C} = \forall k_2 \notin K_2'. \forall k_1 \notin K_1'. \mathcal{C} \quad \text{(swap-2)}$$

$$\sum_{k_1 \notin K_1} \sum_{k_2 \notin K_2} \mathcal{E} = \sum_{k_2 \notin K_2'.} \sum_{k_1 \notin K_1'} \mathcal{E} \quad \text{(swap-3)}$$

$$\forall k \notin K. \mathcal{C} = (\forall k \notin K \cup \{k^*\}. \mathcal{C}) \wedge \mathcal{C}[k \mapsto k^*] \qquad \text{if } k^* \notin K \quad \text{(split-1)}$$

$$C[\sum_{k \notin K} \mathcal{E}] = C[(\sum_{k \notin K \cup \{k^*\}} \mathcal{E}) + \mathcal{E}[k \mapsto k^*]] \qquad \text{where C defines } k^* \neq k' \text{ forall } k' \in K \quad \text{(split-2)}$$

Fig. 5. Equivalences for simplifying constraints where K_2' is defined as $K_2 \setminus \{k_1\}$ and K_1' is defined as $K_1 \cup \{k_2\}$ if $k_1 \in K_2$ and K_1 otherwise.

$$C[\mathcal{E} = 0] \rightsquigarrow_{SE} C[\mathcal{E} = 0 \wedge (\forall i_1 \notin K_1, \ldots, i_l \notin K_l.\, \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) = 0)] \quad \text{(coeff-1)}$$
$$\text{if } \{i_1, \ldots, i_l\} \cap \mathit{ivars}(\mathcal{E}) = \emptyset \text{ and } \mathcal{E} \text{ does not contain } \mathsf{Coeff}$$

$$C[\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})] \rightsquigarrow_{SE} C[\mathit{pmon}(\mathcal{E})] \quad \text{if } \mathit{hmon}(\mathcal{E}) = 1 \text{ and } \mathcal{M} = \mathit{umon}(\mathcal{E}) \quad \text{(coeff-2)}$$

$$C[\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})] \rightsquigarrow_{SE} C[0] \quad \text{if } \mathsf{contMon}_{\mathcal{M}/\mathit{umon}(\mathcal{E})}(\mathit{hmon}(\mathcal{E})) \rightsquigarrow_{SE} \bot \quad \text{(coeff-3)}$$
$$\text{and } C \text{ assures } \mathit{ivars}(\mathcal{M}) \cap \mathit{ivars}(\mathcal{E}) = \emptyset$$

Fig. 6. Rules for introducing and simplifying Coeff expressions

The constraint solving rules are given in Fig. 6. The first rule exploits that if a polynomial is equal to zero, then when interpreting the polynomial as a polynomial over uniform variables, the coefficients for all monomials must be zero. The remaining two rules allow to simplify Coeff expressions. The first rule deals with the case where $\mathcal{E}$ does not contain any handle variables and $\mathcal{M}$ is equal to the monomial over uniform variables contained in $\mathcal{E}$. The second rule deals with the case where it is possible to prove that there is no (SE-computable) instantiation of the handle variables in $\mathcal{E}$ such that the resulting Laurent polynomial contains the monomial $\mathcal{M}$. The rule makes uses the contMon constraint. We will present the rules for showing that such a constraint is contradictory in the next section.

Example 3. Consider the constraint Γ such that

$$\Gamma = (\sum_j \rho_{[j]} R_{[j]} = 0) \wedge \Gamma'$$

We can simplify the constraint as follows:

$$\begin{aligned}
\Gamma &\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}(\sum_j \rho_{[j]} R_{[j]}) = 0 && \text{[coeff-1]}\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}((\sum_{j \notin \{i\}} \rho_{[j]} R_{[j]}) + \rho_{[i]} R_{[i]}) = 0 && \text{[split-2]}\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}(\sum_{j \notin \{i\}} \rho_{[j]} R_{[j]}) + \mathsf{Coeff}_{R_{[i]}}(\rho_{[i]} R_{[i]}) = 0 && \text{[equiv-11]}\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}(\sum_{j \notin \{i\}} \rho_{[j]} R_{[j]}) + \rho_{[i]} = 0 && \text{[coeff-2]}\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, (\sum_{j \notin \{i\}} \mathsf{Coeff}_{R_{[i]}}(\rho_{[j]} R_{[j]})) + \rho_{[i]} = 0 && \text{[equiv-12]}\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, (\sum_{j \notin \{i\}} 0) + \rho_{[i]} = 0 && \text{[coeff-3]}\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \rho_{[i]} = 0 && \text{[equiv-ring]}
\end{aligned}$$

For the step using [coeff-3], we exploit that $\mathsf{contMon}_{R_{[i]}/R_{[j]}}(1) \rightsquigarrow_{SE} \bot$ and that $j \notin \{i\}$ ensures that these index variables will never be instantiated with the same value in the given context. We will give the required rules in the next section. Then, our Gröbner-Basis based simplification algorithm will replace $\rho_{[j]}$ by 0 in Γ for arbitrary index variables j. ■

Proving Coeff to be zero for all *SE* solutions. In this section, we describe a method to check if $\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})$ can be simplified to 0, i.e., for all SE-computable solutions $s = (p, q, \sigma, \delta, \chi, \xi)$, it holds that $\mathit{coeff}_{\sigma(\mathcal{M})}(\mathit{eval}_s(\mathcal{E})) = 0$. As in previous sections, we describe our approach for Type III, but stress that it can be adapted to Type I and Type II, e.g., by transforming the security experiment to make the isomorphisms redundant. We assume that the oracle definitions are efficiently computable and only return handles to elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. Furthermore, we assume that the winning condition only uses handles to elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. This covers most cryptographic constructions of interest (including all SPS schemes). In this case, we never have to deal with handle variables from HVar_t and for $i \in \{1, 2\}$, the polynomials $\boldsymbol{H}_i$ defining the oracle return values contain only handle variables from HVar_i. We distinguish three cases for $\mathsf{contMon}_{\mathcal{M}}(\mathcal{E})$: (i) $\deg(\mathcal{E}) = 0$, (ii) $\deg(\mathcal{E}) = 1$, and (iii) $\deg(\mathcal{E}) > 1$.

Case (i): We use the rule

$$\mathsf{contMon}_{\mathcal{M}}(1) \rightsquigarrow_{SE} \bot \quad \text{if } \mathcal{M} \neq 1.$$

Here, we require that distinct index variables must be instantiated with distinct values, which is ensured by the side condition of the Coeff-(3) rule.

Case (ii): We have $\mathcal{E} = Y_{[j]}$ for $Y \in \mathsf{HVar}_i$, $j \in \mathsf{IVar}$, and $i \in \{1, 2\}$. We must prove that the monomial $\mathcal{M}$ is not computable in i before query j, i.e., it is impossible (in the symbolic group model) to obtain a handle h for $\mathbb{G}_i$ that points to a polynomial F with $m \in \mathit{mons}(F)$ before the j-th oracle query. We perform a proof by contradiction that covers all cases on how a given monomial $\mathcal{M}$ can be computed. We write $\mathsf{canMult}_{i,\{j_1,\ldots,j_n\}}(m)$ if it is possible to perform the multiplication of a given monomial with m using oracle queries with query-indices distinct from $\{j_1, \ldots, j_n\}$. For example, if we have an oracle that returns a handle to $Y * R_{[j]} + W$ in $\mathbb{G}_1$ (where $Y \in \mathsf{HVar}_1, R, W \in \mathsf{UVar}$), then $\mathsf{canMult}_{1,\{j_1\}}(R_{[j_2]} * R_{[j_3]})$ is true since we can call the oracle for indices j_2 and j_3 to perform a multiplication with $R_{[j_2]}$ and $R_{[j_3]}$. In contrast, $\mathsf{canMult}_{1,\{j_1\}}(R_{[j_1]} * R_{[j_2]} * R_{[j_3]})$ is false because we cannot multiply with $R_{[j_1]}$ if using the oracle for query index j_1 is forbidden. To formalize this reasoning, we define a set of rules to reduce a constraint $\mathsf{contMon}_m(Y_{[j]})$ to a disjunction of constraints $\mathsf{canMult}_{i,J}(m)$ such that $\mathit{ivars}(m) = \emptyset$.

We define the set $\mathcal{SM}_i^{SE}$ of *start monomials for a security experiment SE and group index i* as $\mathit{mons}(\boldsymbol{F}_i) \cup (\mathit{mons}(\boldsymbol{H}_i) \cap \mathit{Mon}[\mathsf{UVar}^{\pm 1}])$ where the $\boldsymbol{H}_i$ are considered as polynomials over handle and uniform variables. We define the set $\mathcal{TM}_i^{SE}$ of *transformation monomials for a security experiment SE and a group index i* as $\{m \mid Y * m \in \mathit{mons}(\boldsymbol{H}_i) \wedge Y \in \mathsf{HVar}_i\} \subseteq \mathit{Mon}[\mathsf{UVar}^{\pm 1}]$. For both sets, we partition the previously defined sets into $\mathcal{SM}_i^{SE} = \mathcal{SM}_{i,glob}^{SE} \uplus \mathcal{SM}_{i,orcl}^{SE}$ and

$\mathcal{TM}_i^{SE} = \mathcal{TM}_{i,glob}^{SE} \uplus \mathcal{TM}_{i,orcl}^{SE}$ where the *glob*-sets contain all monomials that contain only global uniform variables and the *orcl*-sets contain all monomials that contain at least one oracle uniform variable. For monomials m, we write $m[j]$ to denote the monomial where all oracle uniform variables Y are replaced with their indexed versions $Y_{[j]}$. We also use the same notation for sets of monomials.

$\mathsf{contMon}_{\tilde{m}}(Y_{[j]}) \rightsquigarrow_{SE}$ [contMon-1]

$\mathsf{canMult}_{i,\{j\}}(\tilde{m}/m_1) \vee \ldots \vee \mathsf{canMult}_{i,\{j\}}(\tilde{m}/m_l) \vee$
$\mathsf{canMult}_{i,\{j,j_1\}}(\tilde{m}/\hat{m}_1[j_1]) \vee \ldots \vee \mathsf{canMult}_{i,\{j,j_1\}}(\tilde{m}/\hat{m}_l[j_1]) \vee$
$\ldots \vee$
$\mathsf{canMult}_{i,\{j,j_n\}}(\tilde{m}/\hat{m}_1[j_n]) \vee \ldots \vee \mathsf{canMult}_{i,\{j,j_n\}}(\tilde{m}/\hat{m}_l[j_n])$

if $Y \in \mathsf{HVar}_i$, $\{m_1, \ldots, m_l\} = \mathcal{SM}_{i,glob}^{SE}$,
$\{\hat{m}_1, \ldots, \hat{m}_l\} = \mathcal{SM}_{i,orcl}^{SE}$, and $\{j_1, \ldots, j_n\} = ivars(\tilde{m}) \setminus \{j\}$.

$\mathsf{canMult}_{i,J}(\tilde{m}) \rightsquigarrow_{SE}$ [contMon-2]

$\mathsf{canMult}_{i,J\cup\{j\}}(\tilde{m}/m_1[j]) \vee \ldots \vee \mathsf{canMult}_{i,J\cup\{j\}}(\tilde{m}/m_l[j])$

if $\{m_1, \ldots, m_l\} = \mathcal{TM}_{i,orcl}^{SE}$ and $j \in ivars(\tilde{m}) \setminus J$.

$\mathsf{canMult}_{i,J}(\tilde{m}) \rightsquigarrow_{SE} \bot$ [contMon-3]

if $J \cap ivars(\tilde{m}) \neq \emptyset$

Fig. 7. Rules for dealing with contMon. We use m/m' to denote the corresponding reduced Laurent monomial

We can now define the rules given in Fig. 7. The first rule captures that to compute the monomial $\tilde{m}$ in i before query j, the adversary must start with a monomial m' (in $m_1, \ldots, m_l, \hat{m}_1[j_1], \ldots$) and then use oracle queries to achieve an indirect multiplication of m' by $\tilde{m}/m'$. Here, the monomials m_i are either monomials included in the adversary input or monomials included in the oracle return values that do not depend on handles and do not contain oracle uniform variables. The monomials $\hat{m}_i[j_u]$ are monomials included in the oracle return values that do not depend on handles and that contain oracle uniform variables. The set of forbidden query indices for the indirect multiplication takes into account that j can never be used and that j_u cannot be used if a monomial with index j_u is used as the start monomial.

The second rule is applicable whenever $\tilde{m}$ contains an indexed uniform variable $R_{[j]}$ such that $j \notin J$. In this case, the j-th query must be used to perform an indirect multiplication that cancels out $R_{[j]}$ and we perform a case distinction on all monomial multiplications containing oracle uniform variables that can be performed by the oracle. For all cases where this step does not cancel out *all*

variables indexed with j, we can use the third rule that formalizes the following fact: If the j-th query is forbidden, there is no way to cancel out a uniform variable with index j.

It is not hard to see that we can reduce all constraints to $\mathsf{canMult}_{i,J}(\tilde{m})$ such that $ivars(\tilde{m}) = \emptyset$: If $ivars(\tilde{m})$ non-empty, then either there is a $j \in ivars(\tilde{m}) \cap J$ and we can conclude with the last rule or we can apply the second rule and add an index $j \in ivars(\tilde{m})$ to J. To check if a constraint $\mathsf{canMult}_{i,J}(\tilde{m})$ with $ivars(\tilde{m}) = \emptyset$ is unsatisfiable, we translate the constraint into a system of linear equations that formalizes the following idea. Let $\{m_1, \ldots, m_l\} = \mathcal{TM}^{SE}_{i,glob}$, then all indirect multiplications that do not introduce indexed uniform variables are of the form

$$m_1^{\delta_1} * \ldots * m_l^{\delta_l}$$

for $\delta_i \in \mathbb{N}$. This corresponds to using the i-th transformation δ_i times to achieve a multiplication with $m_i^{\delta_i}$. To check if there exist $\delta_1, \ldots, \delta_l \in \mathbb{N}$ such that

$$\tilde{m} = m_1^{\delta_1} * \ldots * m_l^{\delta_l}$$

we check if the linear system of equations

$$\begin{aligned} &\deg_{V_1}(\tilde{m}) = \deg_{V_1}(m_1) * \delta_1 + \ldots + \deg_{V_1}(m_l) * \delta_l \\ &\ldots \\ &\deg_{V_n}(\tilde{m}) = \deg_{V_n}(m_1) * \delta_1 + \ldots + \deg_{V_n}(m_l) * \delta_l \end{aligned}$$

has a solution over $\mathbb{N}$ where $\{V_1, \ldots, V_n\}$ is the set of uniform variables that occur in $\tilde{m}, m_1, \ldots, m_l$.

Case (iii): The last case can be handled by generalizing the previous case. We sketch how to achieve this, the full description will be included in the full version of this paper. We have $\mathcal{E} = (Y_1)_{[j_1]} * \ldots * (Y_n)_{[j_n]}$ for $Y_u \in \mathsf{HVar}_{i_u}$, $j_u \in \mathsf{IVar}$, and $i_u \in \{1, 2\}$. To extend the method from Case (ii), we use adapted set of start monomials and transformation monomials that take cancellations between these values for the different handles into account. For example, the set of transformation monomials is the product of transformation monomial sets for $j_1, \ldots, j_n$ also allowing any set to be replaced by $\{1\}$.

Example 4. We will show that $\mathsf{contMon}_{R_{[i]}/V}(M_{[k]})$ is contradictory for the security experiment SE defined in Example 1. Note that $M_{[k]} \in \mathsf{HVar}_2$ and the monomial sets for this group are:

$$\begin{aligned} \mathcal{SM}^{SE}_{2,glob} &= \{1, W\} & \mathcal{SM}^{SE}_{2,orcl} &= \{R, R^2\} \\ \mathcal{TM}^{SE}_{2,glob} &= \{V\} & \mathcal{TM}^{SE}_{2,orcl} &= \emptyset \end{aligned}$$

By applying the first rule in Fig. 7 we have:

$$\begin{aligned} &\mathsf{contMon}_{R_{[i]}/V}(M_{[k]}) \rightsquigarrow_{SE} \\ &\quad \mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}) \vee \mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}W^{-1}) \vee && \text{(div. by 1 and } W) \\ &\quad \mathsf{canMult}_{2,\{k,i\}}(V^{-1}) \vee \mathsf{canMult}_{2,\{k,i\}}(V^{-1}R_{[i]}^{-1}) && \text{(div. by } R_{[i]} \text{ and } R_{[i]}^2) \end{aligned}$$

Now, since $\mathcal{TM}^{SE}_{2,orcl} = \emptyset$, the second rule in Fig. 7 gives us:

$$\mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}) \rightsquigarrow_{SE} \bot$$
$$\mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}W^{-1}) \rightsquigarrow_{SE} \bot$$

Additionally,

$$\mathsf{canMult}_{2,\{k,i\}}(V^{-1}R^{-1}_{[i]}) \rightsquigarrow_{SE} \bot$$

because $\{k,i\} \cap ivars(V^{-1}R^{-1}_{[i]}) \neq \emptyset$. Our problem has been reduced to compute

$$\mathsf{canMult}_{2,\{k,i\}}(V^{-1})$$

so we define the system of equations:

$$\deg_V(V^{-1}) = \deg_V(V) * \delta_1$$

where $\delta_1 \in \mathbb{N}$. The equation is $-1 = 1 * \delta_1$ and it reduces to $\bot$. This analysis proves that $\mathsf{contMon}_{R_{[i]}/V}(M_{[k]}) \rightsquigarrow_{SE} \bot$, i.e., the handle variable $M_{[k]}$ cannot contain the monomial $R_{[i]}/V$.

4.4 Case Distinctions and Contradictions

The rules for case distinctions and contradictions are given in Fig. 8. The first rule is applicable whenever we can express the left-hand-side of an equality with 0 as a product of the two factors $\mathcal{E}_1$ and $\mathcal{E}_2$. Since we reason about elements of an integral domain, we can conclude that at least one of the factors must be equal to 0. The second rule formalizes that if $\mathcal{C}'$ is true for some i, then it is either true for some $i \neq j$ or it is true for $i = j$. The third rule formalizes that for all expressions $\mathcal{E}$, the expression is either equal to 0 or not. We only apply this rule with an $\mathcal{E}$ that already occurs as a subterm of C. In most cases $\mathcal{E} = \rho$ for $\rho \in \mathsf{PVar}$. The final case distinction rule deals with indexed parameter variables $\rho_{[i]}$. Either $\rho_{[i]}$ is equal to zero for all indices not in K or there is an index j not in K such that $\rho_{[j]}$ is not zero. The rule uses Δ to denote all existential bindings in the constraint.

The two contradiction rules are straightforward. The first rule states that a non-zero constant c is not equal to zero. We keep track of applications of this rule to obtain a lower bound on the prime p for which our proof is valid. The second rule just formalizes that zero is always equal to itself.

4.5 Gröbner Basis Simplification

Before applying the Gröbner Basis simplification, we ensure that all $\forall$-quantifiers use the same binders Δ and that all index exception sets are maximal for Δ. This might require renaming of variables, extending the index exception sets, and introducing unused variables. For the $\sum$-binders $\hat{\Delta}_u$, we assume for all u, v that (i) $\hat{\Delta}_u = \hat{\Delta}_v$, (ii) $\hat{\Delta}_u$ is a prefix of $\hat{\Delta}_v$, or (iii) vice versa.

$$C[\mathcal{E}_1 * \mathcal{E}_2 = 0] \rightsquigarrow_{SE} C[\mathcal{E}_1 = 0] \vee C[\mathcal{E}_2 = 0] \qquad \text{[dist-1]}$$

$$C[\exists i \notin K.\, \mathcal{C}'] \rightsquigarrow_{SE} \begin{array}{l} C[\exists i \notin K \cup \{j\}.\, \mathcal{C}'] \\ \vee\, C[\mathcal{C}'[i \mapsto j]] \end{array} \quad \text{if } j \notin K \qquad \text{[dist-2]}$$

$$C[\mathcal{C}'] \rightsquigarrow_{SE} \begin{array}{l} C[\mathcal{C}' \wedge \mathcal{E} = 0] \\ \vee\, C[\mathcal{C}' \wedge \mathcal{E} \neq 0] \end{array} \quad \text{where } \mathcal{E} \text{ arbitrary} \qquad \text{[dist-3]}$$

$$\exists \Delta.\, \mathcal{C}' \rightsquigarrow_{SE} \begin{array}{l} \exists \Delta.\, (\forall i \notin K.\, \rho_{[i]} = 0) \wedge \mathcal{C}' \\ \vee\, \exists \Delta, j \notin K.\, \rho_{[j]} \neq 0 \wedge \mathcal{C}' \end{array} \quad \begin{array}{l} \text{where } K \text{ arbitrary} \\ \text{and } j \notin \mathit{ivars}(\Delta) \\ \cup \mathit{ivars}(\mathcal{C}') \end{array} \qquad \text{[dist-4]}$$

$$C[c = 0] \rightsquigarrow_{SE} \bot \quad \text{if } c \in \mathbb{Z} \setminus \{0\} \qquad \text{[false-1]}$$

$$C[0 \neq 0] \rightsquigarrow_{SE} \bot \qquad \text{[false-2]}$$

Fig. 8. Rules for performing case distinctions and contradictions.

The resulting constraint system can be rearranged to have the following form

$$\exists \nabla.\, (\forall \Delta.\, \mathcal{E}_1 = 0) \wedge \ldots \wedge (\forall \Delta.\, \mathcal{E}_l = 0) \wedge$$
$$(\forall \Delta.\, \hat{\mathcal{E}}_1 \bowtie_1 0) \wedge \ldots \wedge (\forall \Delta.\, \hat{\mathcal{E}}_{\hat{l}} \bowtie_{\hat{l}} 0)$$

where the $\mathcal{E}_u$ are expressions that do not contain handle variables, uniform variables, or Coeff expressions, which we call parameter equality polynomials. The $\hat{\mathcal{E}}_u$ denote the remaining expressions. We want to move all the $\mathcal{E}_u$ under a single quantifier for simplification. To take renamings of the bound variables into account, we ensure beforehand that for all $\mathcal{E}_u$ and all permutations of the $\forall$-bound variables, the resulting expression is already included. For example, given

$$\forall j_1, j_2 \notin \{j_1\}.\, \rho_{[j_1]} * \rho'_{[j_2]} = 0 \wedge \forall j_1, j_2 \notin \{j_1\}.\, \rho_{[j_2]} * \rho'_{[j_1]} - \alpha = 0$$

it is usually useful to add at least the permutation

$$\forall j_1, j_2 \notin \{j_1\}.\, \rho_{[j_1]} * \rho'_{[j_2]} - \alpha = 0$$

before moving everything under a common quantifier since this yields the shared monomial $\rho_{[j_1]} * \rho'_{[j_2]}$. After moving the parameter equality polynomials under the same quantifier, we get:

$$\exists \nabla.\, (\forall \Delta.\, \mathcal{E}_1 = 0 \wedge \ldots \wedge \mathcal{E}_l = 0) \wedge$$
$$(\forall \Delta.\, \hat{\mathcal{E}}_1 \bowtie_1 0) \wedge \ldots \wedge (\forall \Delta.\, \hat{\mathcal{E}}_{\hat{l}} \bowtie_{\hat{l}} 0)$$

Now, we move non-indexed parameters in monomials out of the $\sum$-binder and consistently replace non-bound parameters and $\sum$-expressions with variables X_v. We call the corresponding mapping σ and use g_u to denote polynomial resulting from $\mathcal{E}_u$. We can revert this abstraction process by applying σ, i.e.,

$\sigma(g_u) = \mathcal{E}_u$. Next, we compute the Gröbner Basis (over $\mathbb{Z}$) of the ideal $\langle g_1, \ldots, g_l \rangle$ which we denote with $I = \langle g'_1, \ldots, g'_{l'} \rangle$. By the properties of the Gröbner Basis, we know that

$$(g_1 = 0 \wedge \ldots \wedge g_l = 0) \Leftrightarrow (g'_1 = 0 \wedge \ldots \wedge g'_{l'} = 0)$$

and hence

$$(\mathcal{E}_1 = 0 \wedge \ldots \wedge \mathcal{E}_l = 0) \Leftrightarrow (\mathcal{E}'_1 = 0 \wedge \ldots \wedge \mathcal{E}'_{l'} = 0)$$

for $\mathcal{E}'_u = \sigma(g_u)$ which we exploit to simplify the parameter equality polynomials. For computing the Gröbner Basis, we use a monomial order that prefers to eliminate abstracted $\sum$ expressions. Next, we use the Gröbner Basis to simplify the expressions $\forall \Delta. \hat{\mathcal{E}}_u \bowtie_u 0$. If $\hat{\mathcal{E}}_u$ uses all variables in Δ, we use an extension σ' of σ to abstract $\hat{\mathcal{E}}_u$ to the polynomial f and define f' as the result of reducing f modulo the Gröbner Basis I. As before, we define the simplified $\hat{\mathcal{E}}'_u$ as $\sigma'(f')$. Often, it is very useful to also simplify below $\sum$-binders. We use an example to illustrate how this works.

Example 5. Assume $\nabla = j_1$, $\Delta = j_2 \notin \{j_1\}$, $I = \langle X_1 * X_2 \rangle$, $\sigma = \{X_1 \mapsto \rho_{[j_1]}, X_1 \mapsto \rho'_{[j_2]}\}$, and

$$\mathcal{E}_1 = (\sum_{j_3 \notin \{j_1\}} \rho_{[j_1]} * \rho'_{[j_3]} = 0).$$

Then we use $\forall j_2 \notin \{j_1\}. \rho_{[j_1]} * \rho'_{[j_2]} = 0$ to rewrite $\rho_{[j_1]} * \rho'_{[j_3]}$ to 0 below $\sum_{j_3 \notin \{j_1\}}$ by instantiating j_2 with j_3 (both have the same exception j_1). ■

4.6 Example: Proof of EUF-CMA for SPS

In this section show how our constraint solving rules can be used to prove (unbounded) EUF-CMA security of the signature scheme in Fig. 1. The winning constraints for the associated security experiment *SE* are already given in Example 2. To prove EUF-CMA security in the Generic Group Model, we must show that the following constraint has no *SE*-computable solution

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}) \\
&- ((\tau + \sum_k \tau'_{[k]} * R_{[k]} + \sum_k \tau''_{[k]} * (M_{[k]} * V + W + R^2_{[k]})) \\
&\quad * (\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W) + \hat{M} * V + W) = 0 \qquad (1) \\
\wedge\quad &\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W \\
&- (\tau + \sum_k \tau'_{[k]} * R_{[k]} + \sum_k \tau''_{[k]} * (M_{[k]} * V + W + R^2_{[k]})) = 0 \qquad (2) \\
\wedge\quad &\forall k. \hat{M} - M_{[k]} \neq 0 \qquad (3)
\end{aligned}$$

where $\hat{M}$ is defined as

$$\hat{M} = \mu + \sum_k \mu'_{[k]} * R_{[k]} + \sum_k \mu''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}).$$

Instead of immediately simplifying everything using the equivalences in Fig. 5, we first apply the rule [coeff-1] where $\mathcal{M} = R^2_{[i]}$ and $\mathcal{E}$ is the Eq. (2). After simplifying the resulting Coeff expressions (see Example 4), we get the new equation $\forall i. -\tau''_{[i]} = 0$. Our Gröbner Basis simplification replaces every occurrence of τ''_i by 0. This results in the following new constraint:

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}) \\
&- ((\tau + \sum_k \tau'_{[k]} * R_{[k]}) * (\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W) \\
&\quad + \hat{M} * V + W) = 0 && (1)\\
\wedge\quad &\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W - (\tau + \sum_k \tau'_{[k]} * R_{[k]}) = 0 && (2)\\
\wedge\quad &\forall k. \hat{M} - M_{[k]} \neq 0 && (3)
\end{aligned}$$

Now, we can apply the rule [coeff-1] where $\mathcal{E}$ is the left hand side of Eq. (2) and for different monomials $\mathcal{M}$, we obtain the following new equations:

$$\begin{aligned}
&\rho - \tau = 0 && \text{for } \mathcal{M} = 1\\
&\forall k. \rho'_{[k]} - \tau'_{[k]} = 0 && \text{for } \mathcal{M} = R_{[k]}\\
&\rho'' = 0 && \text{for } \mathcal{M} = V\\
&\rho''' = 0 && \text{for } \mathcal{M} = W
\end{aligned}$$

After this, we basically got rid of Eq. (2) and our Gröbner Basis simplification yields:

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + R^2_{[k]} + W) \\
&- (\tau^2 + (2 \sum_k \tau * \tau'_{[k]} * R_{[k]}) + \sum_{k,k' \notin \{k\}} \tau'_{[k]} * \tau'_{[k']} * R_{[k]} * R_{[k']} \\
&\quad + \sum_k \tau'^2_{[k]} * R^2_{[k]} + \hat{M} * V + W) = 0 && (1)\\
\wedge\quad &\forall k. \hat{M} - M_{[k]} \neq 0 && (2)
\end{aligned}$$

We now apply the rule [coeff-1] where $\mathcal{E}$ is expression in Eq. (1) obtaining the following new equations:

$$\begin{aligned}
\wedge\quad &\sum_k \gamma''_{[k]} - 1 = 0 && \text{for } \mathcal{M} = W && (3)\\
\wedge\quad &\forall k. \gamma''_{[k]} - \tau'^2_{[k]} = 0 && \text{for } \mathcal{M} = R^2_{[k]} && (4)\\
\wedge\quad &\forall k. \forall k' \notin \{k\}. 2 * \tau'_{[k]} * \tau'_{[k']} = 0 && \text{for } \mathcal{M} = R_{[k]} R_{[k']} && (5)
\end{aligned}$$

Then, we apply the rule [dist-4] with $K = \emptyset$ to perform a case distinction on the parameter τ':

$$\begin{aligned} & \forall k.\, \tau'_{[k]} = 0 \wedge \Gamma && \text{(case 1)} \\ \vee \quad & \exists k^*.\, \tau'_{[k^*]} \neq 0 \wedge \Gamma && \text{(case 2)} \end{aligned}$$

Here, Γ represents the conjunction of our previous five equations. In case 1, the Gröbner Basis simplification results in the system

$$\begin{aligned} & \gamma + \sum_k \gamma'_{[k]} * R_{[k]} - \tau^2 - \hat{M} * V - W = 0 && (1) \\ \wedge \quad & \forall k.\, \hat{M} - M_{[k]} \neq 0 && (2) \\ \wedge \quad & -1 = 0 && (3) \end{aligned}$$

which simplifies to $\bot$ after applying rule [false-1] to Eq. (3).
In case 2, Gröbner Basis simplification yields:

$$\begin{aligned} & \exists k^*. \\ & \quad \gamma + \sum_k \gamma'_{[k]} * R_{[k]} + M_{[k^*]} * V - \tau^2 - 2\tau R_{[k^*]} - \hat{M} * V && (1) \\ \wedge \quad & \forall k.\, \hat{M} - M_{[k]} \neq 0 && (2) \end{aligned}$$

We apply the rule [coeff-1] where $\mathcal{E}$ is the left hand side of Eq. (1) for different monomials as $\mathcal{M}$, obtaining:

$$\begin{aligned} & \gamma - \tau^2 = 0 && \text{for } \mathcal{M} = 1 \\ & \forall k \notin \{k^*\}.\, \gamma'_{[k]} = 0 && \text{for } \mathcal{M} = R_{[k]} \\ & \gamma'_{[k^*]} - 2\tau = 0 && \text{for } \mathcal{M} = R_{[k^*]} \end{aligned}$$

After simplifying the system, we obtain:

$$\begin{aligned} & M_{[k^*]} * V - \hat{M} * V = 0 && (1) \\ \wedge \quad & \forall k.\, \hat{M} - M_{[k]} \neq 0 && (2) \end{aligned}$$

Applying the rule [dist-1] to Eq. (1) we obtain two cases:

$$\begin{array}{lll} \exists k^*. & & \exists k^*. \\ \quad V = 0 & \bigvee & \quad M_{[k^*]} - \hat{M} = 0 \\ \wedge \quad \forall k.\, \hat{M} - M_{[k]} \neq 0 & & \wedge \quad \forall k.\, \hat{M} - M_{[k]} \neq 0 \\ \text{(case 2.1)} & & \text{(case 2.2)} \end{array}$$

In case 2.1, after applying [coeff-1] for $\mathcal{M} = V$ to the first equation and simplifying, we obtain the equation $1 = 0$ that reduces to $\bot$ according to rule [false-1].

Finally, in case 2.2 we apply the rule [split-2] and we get the system:

$$\begin{aligned} & M_{[k^*]} - \hat{M} = 0 \\ \wedge \quad & \forall k \notin \{k^*\}. \hat{M} - M_{[k]} \neq 0 \\ \wedge \quad & \hat{M} - M_{[k^*]} \neq 0 \end{aligned}$$

Our Gröbner Basis simplification will reduce it to,

$$0 \neq 0 \wedge (\forall k \notin \{k^*\}. \hat{M} - M_{[k]} \neq 0)$$

which reduces to $\bot$ according to rule [false-2].

5 Implementation and Case Studies

We have implemented the described algorithm in the gga$^\infty$ tool[3] and have evaluated its effectiveness and performance on cryptographic constructions from the literature (presented in Table 1) and automatically synthesized schemes (presented in Table 2). The source code is written in OCaml and uses the computer algebra system SAGE [40] for Gröbner Basis computations and the SMT solver Z3 [20] for checking the satisfiability of linear equations over the natural numbers. Although the code reproduces the algorithm as it is described in this paper, it also implements some optimizations and additional rules to derive contradictions, that will be further explained in the full version of this paper.

The tool takes an input file such as the one shown in Fig. 9 and performs a proof search using our constraint solving rules guided by a heuristic. If the search is successful, the tool returns a representation of the proof tree. To ensure termination, we establish a timeout of 1000 s.

```
group_setting 3.

sample V,W.
input [V,W] in G1.

oracle o1(M:G2) =
  sample R;
  return [ R ] in G1,
         [ R, M*V + R^2 + W] in G2.

win (wM:G2, wT1:G1, wT2:G2, wS:G2) =
 ( (forall i: wM <> M_i) /\ wT1 = wT2 /\ wS = V*wM + wT1*wT2 + W ).
```

Fig. 9. Input file for the Type III re-randomizable SPS scheme from Fig. 1

[3] Source code and case studies at http://generic-group-analyzer.github.io/.

5.1 Case Studies

We analyze the security of cryptographic constructions from the literature and collect the results in Table 1. The first five entries do not require support for oracles that take handles and are therefore also in the scope of the tool presented in [11]. For the first four entries, both the tool from [11] and gga$^{\infty}$ prove unbounded security. For the fifth example, gga$^{\infty}$ succeeds, whereas the tool from [11] fails to find a proof.

The remaining examples are all outside the scope of the tool from [11]. First, we analyze the Message Authentication Codes proposed in [18]. They propose two MACs (instead of public key signatures) as the basis for their anonymous credential system. One of them is proven secure in the Generic Group Model and the other under the decisional Diffie-Hellman (DDH) assumption. Our tool confirms the first proof and finds a proof in the Generic Group Model for the second construction[4].

We also prove security for a number of structure-preserving signature schemes. First, we analyze the scheme proposed in [2] for bilinear groups of Type III.

Then, we analyze the re-randomizable scheme from [4] for Type II and Type III. Next, we prove sEUF-CMAsecurity of the unified SPS signature scheme proposed in [5], which is secure in all three settings. We also prove EUF-CMA security of its re-randomizable version (randomization tokens are given to the adversary). Later, we analyze the translation of the scheme for Type III proposed in [19]. We also consider the Type II scheme from [12].

Finally, we analyze two instances of fully structure-preserving signature schemes proposed in [26].

Table 1. Case studies (last column denotes time for fully automated proof).

Reference	Scheme	Property	Time
Lysyanskaya et al. '99 [30]	LRSW assumption	Valid	2s
Abe et al. '11 [5]	One-time SPS in Type I	OT-EUF-CMA	1s
Pointcheval et al. '15 [34]	Assumption 1	Valid	1s
"	Assumption 2	Valid	1s
"	Multi-message sign. scheme ($r = 3$)	EUF-CMA	1s
Chase et al. '13 [18]	MAC_{GGM} (messages length ≤ 3)	UF-CMVA	1s
"	MAC_{DDH} (messages length ≤ 3)	UF-CMVA	3s
Abe et al. '11 [2]	SPS scheme, messages in $\mathbb{G}_1 \times \mathbb{G}_2$	sEUF-CMA	22s
Abe et al. '14 [4]	Re-random. SPS for msg. in $\mathbb{G}_2$	EUF-CMA	6s
Abe et al. '14 [5]	Unified SPS scheme	sEUF-CMA	5s
"	Unified SPS scheme (with tokens)	EUF-CMA	11s
Chatterjee et al. '15 [19]	Type III randomizable SPS	EUF-CMA	3s
Barthe et al. '15 [12]	Re-randomizable SPS in Type III	EUF-CMA	6s
Groth '15 [26]	Fully comb. $\text{SPS}_{b=0}$ $(m, n = 1)$	EUF-CMA	8s
"	Fully comb. $\text{SPS}_{b=1}$ $(m, n = 1)$	sEUF-CMA	8s

[4] This is of course implied by the pen-and-paper proof under the DDH assumption.

Table 2. Synthesis results for SPS schemes in Type Ⅱ and Type Ⅲ with $r, v, w \xleftarrow{\$} \mathbb{Z}_p$, verification keys $V = g_1^v, W = g_1^w \in \mathbb{G}_1$, message $M = g_2^m \in \mathbb{G}_2$ and signatures $S_1 = g_1^{s_1} \in \mathbb{G}_1$, $S_2 = g_2^{s_2}, S_3 = g_2^{s_3} \in \mathbb{G}_2$.

Search Space			Results	
	Verification equations	First signature elements	2-secure	∞-secure
Ⅱ	$s_3 = f(r, v, w, m)$	$S_2 = [\![r]\!]_2$,	1	1
	$s_3 s_2 = f(r, v, w, m)$	$S_2 = [\![r]\!]_2$,	12	9
	$s_3(s_2 - w) = f(r, v, w, m)$	$S_2 = [\![r + w]\!]_2$,	14	8
Ⅲ	$s_1 = s_2 \wedge \quad s_3 = f(r, v, w, m)$	$S_1 = [\![r]\!]_1$, $S_2 = [\![r]\!]_2$	2	2
	$s_1 = s_2 \wedge \ s_1 s_3 = f(r, v, w, m)$	$S_1 = [\![r]\!]_1$, $S_2 = [\![r]\!]_2$	117	75
	$s_1 s_2 = 1 \ \wedge \ s_1 s_3 = f(r, v, w, m)$	$S_1 = [\![r]\!]_1$, $S_2 = [\![r^{-1}]\!]_2$	39	22
			185	117

To evaluate our tool on a wider range of examples, we also make use of the synthesis tool for structure-preserving signature schemes presented in [12]. We take the existing results for Type Ⅱ from [12] and use our tool to analyze (unbounded) EUF-CMA-security for all schemes where the tool from [12] succeeds to prove 2-EUF-CMA security. We also extend the synthesis tool to generate new schemes in Type Ⅲ and apply our tool to those schemes that can be proven 2-EUF-CMA secure with the tool from [12]. The results for both Type Ⅱ and Type Ⅲ are summarized in Table 2. We classify the schemes in different groups, depending on the shape of the verification equations (first column). The column 2-*secure* represents the number of schemes of each group that are proven 2-EUF-CMA secure using the tool from [12], while the column ∞-*secure* represents the number of schemes of each group that are proven EUF-CMA secure using our tool (for all bounds that are polynomial in the security parameter).

Acknowledgements. This work is supported in part by ONR grant N00014-12-1-0914, Madrid regional project S2009TIC-1465 PROMETIDOS, and Spanish national projects TIN2009-14599 DESAFIOS 10, and TIN2012-39391-C04-01 Strongsoft. The research of Schmidt has received funds from the European Commissions Seventh Framework Programme Marie Curie Cofund Action AMAROUT II (grant no. 291803).

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
2. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (2011)

3. Abe, M., Groth, J., Ohkubo, M., Tango, T.: Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 241–260. Springer, Heidelberg (2014)
4. Abe, M., Groth, J., Ohkubo, M., Tibouchi, M.: Structure-preserving signatures from type II pairings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 390–407. Springer, Heidelberg (2014)
5. Abe, M., Groth, J., Ohkubo, M., Tibouchi, M.: Unified, minimal and selectively randomizable structure-preserving signatures. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 688–712. Springer, Heidelberg (2014)
6. Abe, M., Kohlweiss, M., Ohkubo, M., Tibouchi, M.: Fully structure-preserving signatures and shrinking commitments. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 35–65. Springer, Heidelberg (2015)
7. Akinyele, J.A., Garman, C., Hohenberger, S.: Automating fast and secure translations from type-I to type-III pairing schemes. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015, pp. 1370–1381. ACM, New York (2015)
8. Akinyele, J.A., Green, M., Hohenberger, S.: Using SMT solvers to automate design tasks for encryption, signature schemes. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) 20th Conference on Computer and Communications Security, ACM CCS 2013, 4–8 November 2013, Berlin, Germany, pp. 399–410. ACM Press (2011)
9. Barthe, G., Cederquist, J., Tarento, S.: A machine-checked formalization of the generic model and the random oracle model. In: Basin, D., Rusinowitch, M. (eds.) IJCAR 2004. LNCS (LNAI), vol. 3097, pp. 385–399. Springer, Heidelberg (2004)
10. Barthe, G., Crespo, J.M., Grégoire, B., Kunz, C., Lakhnech, Y., Schmidt, B., Zanella Béguelin, S.: Fully automated analysis of padding-based encryption in the computational model. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) 20th Conference on Computer and Communications Security, ACM CCS 2013, 4–8 November 2013, Berlin, Germany, pp. 1247–1260. ACM Press (2011)
11. Barthe, G., Fagerholm, E., Fiore, D., Mitchell, J., Scedrov, A., Schmidt, B.: Automated analysis of cryptographic assumptions in generic group models. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 95–112. Springer, Heidelberg (2014)
12. Barthe, G., Fagerholm, E., Fiore, D., Scedrov, A., Schmidt, B., Tibouchi, M.: Strongly-optimal structure preserving signatures from type II pairings: synthesis and lower bounds. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 355–376. Springer, Heidelberg (2015)
13. Barthe, G., Grégoire, B., Heraud, S., Béguelin, S.Z.: Computer-aided security proofs for the working cryptographer. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 71–90. Springer, Heidelberg (2011)
14. Barthe, G., Grégoire, B., Schmidt, B.: Automated proofs of pairing-based cryptography. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 12–16 October 2015, Denver, CO, USA, pp. 1156–1168 (2015)
15. Barthe, G., Tarento, S.: A machine-checked formalization of the random oracle model. In: Filliâtre, J.-C., Paulin-Mohring, C., Werner, B. (eds.) TYPES 2004. LNCS, vol. 3839, pp. 33–49. Springer, Heidelberg (2006)
16. Blanchet, B.: A computationally sound mechanized prover for security protocols. In: IEEE Symposium on Security and Privacy, 21–24 May 2006, Berkeley, California, USA, pp. 140–154. IEEE Computer Society Press (2006)

17. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
18. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs, keyed-verification anonymous credentials. In: Ahn, G.-J., Yung, M., Li, N. (eds.) 21st Conference on Computer and Communications Security, ACM CCS 2014, 3–7 November 2014, Scottsdale, AZ, USA, pp. 1205–1216. ACM Press (2011)
19. Chatterjee, S., Menezes, A.: Type 2 structure-preserving signature schemes revisited. In: Iwata, T., et al. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 286–310. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_13
20. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
21. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013)
22. Fagerholm, E.: Automated analysis in generic groups. Ph.D. thesis, University of Pennsylvania (2015)
23. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
24. Fuchsbauer, G.: Breaking existential unforgeability of a signature scheme from asiacrypt 2014. Cryptology ePrint Archive, Report 2014/892 (2014). http://eprint.iacr.org/2014/892
25. Fuchsbauer, G., Hanser, C., Slamanig, D.: EUF-CMA-secure structure-preserving signatures on equivalence classes. Cryptology ePrint Archive, Report 2014/944 (2014). http://eprint.iacr.org/2014/944
26. Groth, J.: Efficient fully structure-preserving signatures for large messages. In: Iwata, T., et al. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 239–259. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_11
27. Hoang, V.T., Katz, J., Malozemoff, A.J.: Automated analysis and synthesis of authenticated encryption schemes. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 12–16 October 2015, Denver, CO, USA, pp. 84–95 (2015)
28. Hwang, J.Y., Lee, D.H., Yung, M.: Universal forgery of the identity-based sequential aggregate signature scheme. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) 4th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, 10–12 March 2009, Sydney, Australia, pp. 157–160. ACM Press (2011)
29. Jager, T., Schwenk, J.: On the equivalence of generic group models. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 200–209. Springer, Heidelberg (2008)
30. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H., Adams, C. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
31. Malozemoff, A.J., Katz, J., Green, M.D.: Automated analysis and synthesis of block-cipher modes of operation. In: IEEE 27th Computer Security Foundations Symposium, CSF 2014, 19–22 July 2014, Vienna, Austria, pp. 140–152 (2014)
32. Maurer, U.M.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)

33. Nechaev, V.: Complexity of a determinate algorithm for the discrete logarithm. Math. Notes **55**(2), 165–172 (1994)
34. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Heidelberg (2016)
35. Rupp, A., Leander, G., Bangerter, E., Dent, A.W., Sadeghi, A.-R.: Sufficient conditions for intractability over black-box groups: generic lower bounds for generalized DL and DH problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 489–505. Springer, Heidelberg (2008)
36. Schnorr, C.-P.: Security of blind discrete log signatures against interactive attacks. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 2001. LNCS, vol. 2229, pp. 1–12. Springer, Heidelberg (2001)
37. Schnorr, C.-P., Jakobsson, M.: Security of signed ElGamal encryption. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 73–89. Springer, Heidelberg (2000)
38. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
39. Szydlo, M.: A note on chosen-basis decisional Diffie-Hellman assumptions. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 166–170. Springer, Heidelberg (2006)
40. The Sage Developers. Sage Mathematics Software (Version 6.8) (2015). http://www.sagemath.org