Automated applications of Cryptographic Assumptions

Charlie Jacomme
supervised by Gilles Barthes and Benedikt Schmidt, IMDEA software
and Hubert Comon-Lundh, LSV

1. Introduction

General context. The last decades have seen the development of the internet to a point where we use it in our everyday lifes for operations that can be critical like handling our bank account or buying items. With this kind of critical operations, we need a secure way to execute them. For example, if you buy something online, nobody should be able to obtain your credit card number, and if your phone has a GPS chip, you do not want someone to be able to steal your location.

We see here emerging two important notions of security : confidentiality, which means that an eavesdropper cannot recover the content of a message, and authentication, which means that the sender of a message is who it is supposed to be.

This lead to the design of symmetric encryption schemes, where the different parties have a previously shared secret and they use this secret to encrypt and decrypt some messages. Only someone with the corresponding secret can decrypt an encrypted message, thus achieving confidentiality, provided they can find a way to share the secret. Among such schemes we can cite AES or DES.

An encryption scheme can be run on a single computer, but authentication is a property involving programs running on different locations. This kind of distributed programs is called a protocol. For example, a key exchange protocol runs on two different computers and allows the computers to share a new secret value. An example of such a protocol is the NSL protocol, which is actually an interesting example because it was first proposed in 1978 [19] but an attack was found in 1995 and it was then corrected [17]. We see here that designing a protocol can be an error prone process and a protocol that seems to be secure might be inherently flawed.

A more recent example is the TLS protocol, a widely used authentication protocol. Its first version was published in 1999, and in the past 17 years many different attacks have been discovered, the protocol being patched consequently over the years. NSL and TLS are only two examples among many others. The consequence is clear : protocols and schemes can look secure at first glance but neither intuition nor experience can be trusted on this. In order to have truly secure protocols, we need some way to prove it formally.

Proven security. At this stage the problem is : given a protocol and a property, can we prove that the protocol satisfies the property ? It may look like a simple question of model checking but in classical model checking we usually ensure the safety of a program, i.e that any run of the program will verify some property. In our problem, the program should verify some property, even when it is running in parallel with any attacker : there is a universal quantifier over the attacker.

Besides the universal quantification, we need also to precise the range of the quantification : what is an attacker ? As we cannot provide any guarantee on what happens over the network, we must consider the worst case scenario and consider that the attacker has full control over it, i.e. can intercept and forge any message. Furthermore, we also need to define the computation capabilities of the attacker : he might have access to some super computer, or maybe a quantum calculator, or maybe it is just a basic laptop. If we prove something with a too weak attacker, the proof is useless in practice, but if we consider an all powerful attacker, we will never be able to prove any security.

To propose a solution to this question, two main models have been developed and used in the past, relying on two different assumptions. The first one relies on the Dolev-Yao assumption [10] and is called the symbolic

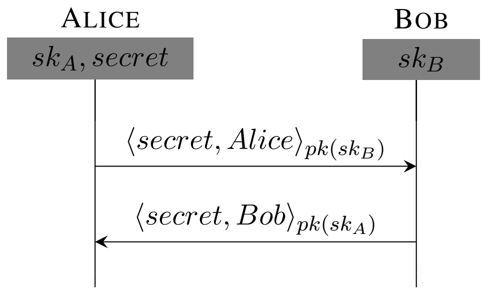
ALICE
BOB
$sk_A, secret$
sk_B
$\langle secret, Alice\rangle_{pk(sk_B)}$
$\langle secret, Bob\rangle_{pk(sk_A)}$

Figure 1. Basic handsake protocol

setting. In this model, the idea is to describe precisely what the attacker can do using an inference system. In the second one, the computational model, the attacker is any probabilistic polynomial time Turing machine (PPT).

The symbolic model. To develop a bit, in the symbolic setting the assumption is basically that the encryption schemes are unbreakable. Therefore, an attacker can decrypt a cyphertext only if he has the corresponding secret key. We can then model the messages with terms, where the function symbols capture the cryptographic primitives. A message could for example be $enc(''hello'', secret\_key)$, and "hello" can be obtained from this term only if one also knows the secret key. Of course, this model is not well suited for the verification of encryption schemes themselves. It is however relevant to protocol verification. Let us for example consider a simple hand shake protocol described in figure 1. We use here an asymmetric encryption scheme where anyone can encrypt with some public-key but only the person with the corresponding secret key can decrypt the message. It can be represented by the term $\langle ''hello'' \rangle_{pk(secret\_key)}$. Intuitively, it acts has a locker that anyone can lock but which can only be opened with a key. Using this encryption scheme we show in figure 1 a basic handshake protocol where Alice sends some secret to Bob using its public key, and Bob replies to Alice as an acknowledgement.

In the formal model, as the encryption scheme is assumed to be perfect and as the attacker does not know Alice's secret key nor Bob's, we can prove that if the protocol has been completed, then Alice and Bob share a common secret, which is unknown to the attacker.

This is a toy example where it is quite simple to see what is happening and one can formally prove the confidentiality of the secret, but in general, in the symbolic model it is impossible to decide if a protocol does satisfy some property. It is however not an issue in many practical cases and automation of security analysis in this model has been well-developed for years and several tools [18, 14, 8] have proven their usefulness.

However, if this formal model allows to find specific flaws in the construction of protocols, it does not capture many of the real world attacks because we assumed the encryption to be perfect. For example, the encryption scheme might ensure confidentiality of the message but not the integrity, thus allowing the attacker to modify the message inside the encryption. Let us consider once more the protocol of figure 1, but this time we assume that the encryption of a pair is the pair of the encryptions : $\langle secret, Alice\rangle_{pk(sk_B)} = (\langle secret\rangle_{pk(sk_B)}, \langle Alice\rangle_{pk(sk_B)})$. Then, there is an attack on the protocol, which allows an attacker to obtain the secret, as shown in figure 2. Intuitively, Charlie can intercept the message from Alice to Bob and insert his name inside it, so Bob will reencrypt the message with Charlie's public key, thus allowing Charlie to obtain the secret and finally forge the answer to Alice.

This example highlights the limitations of the symbolic model : a protocol found secure under the Dolev Yao assumption might be broken in the real world if the implementation of the primitives has an unexpected weakness.

The computational model. To fix the above problem, this second model considers that an attacker is a PPT. We can then consider the security of a scheme against such attackers and if, in some protocol the encryption scheme

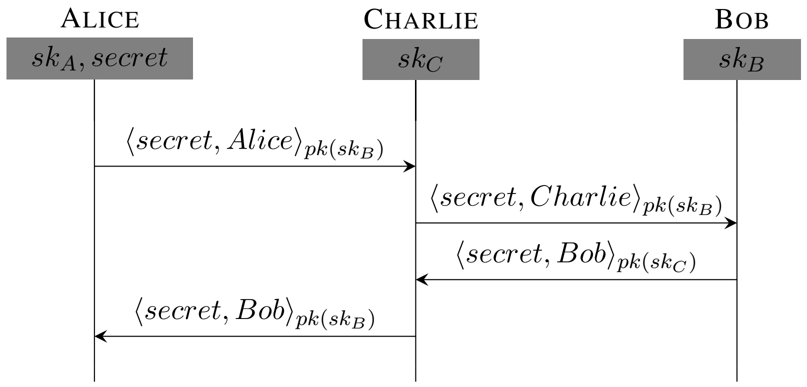
ALICE
CHARLIE
BOB
$sk_A, secret$
sk_C
sk_B
$\langle secret, Alice\rangle_{pk(sk_B)}$
$\langle secret, Charlie\rangle_{pk(sk_B)}$
$\langle secret, Bob\rangle_{pk(sk_C)}$
$\langle secret, Bob\rangle_{pk(sk_B)}$

Figure 2. Attack on the key exchange

does not preserve the integrity, there will be an attacker who can mount the attack. Therefore, the previous attack, which was ignored in the symbolic model, would be captured in this model.

A direct consequence of considering a probabilistic attacker is that if we want to know if some specific value is secret, there is always an attacker who can, with some luck, pick at random a key, get the correct value and break the secrecy. To avoid this problem, we just try to prove that, with high probability, no attacker can break the secrecy. Moreover, the guarantee provided by a security scheme will always depend on the size of the secret key. If a secret key is made of only 1 bit, a polynomial attacker will always be able to break it and obtain the secret just by trying the two possibilities. Therefore, we need to consider probabilities that depend on the size of some security parameters, classically the secret key, and we then want to prove that some scheme or protocol guarantees the property with an asymptotically high probability.

Proving such a result is difficult and the classical proof method is by reduction : we assume that some problem is difficult and then prove the desired result. A classical assumption is for example the Decisional Diffie-Helmann problem, where we consider that given three group elements g^a, g^b, g^c, the complexity of deciding if $c = ab$ is high. We then assume that no PPT attacker can solve this problem with a non negligible probability. Given a scheme and a property, we then reason ad absurdo and assume that we have an attacker, who breaks the property and try to construct a new attacker using this attacker, who solves DDH.

Proof method in the computational model. The example of the OAEP scheme [21] shows that security proofs in the computational model are error prone. Therefore, it is wise to design formally verifiable proofs.

In order to formalize the framework, in which such proofs should be carried, the concept of cryptographic games was introduced [22, 5]. The core idea is to consider the interactions between the attacker and the protocol or scheme as a game and then perform game transformations until breaking a hardness assumption. Designing a formal framework for cryptographic game transformations was the starting point of many projects [3, 6, 23, 12].

Proof automation. Once computer verified proof were available, the complexity of the proof search exposed the need for automation. Computer-aided analysis has been developed a lot in the past years [4, 1, 7, 20]. Cryptoverif captures a small set of game transformations and thus allows for a lot of automation. However, its drawbacks is that in many cases we cannot obtain a proof, and we do not know if it is because there is an atack or because we need some other game transformation. Another stand point was taken with CertiCrypt : it relies on the Coq prover and can then capture any valid game transformation, but it is a lot more difficult to automate. This difficulty yield the design of Easycrypt and Zoocrypt which both tried to achieve automation for some specific kind of schemes or protocols. These tools finally lead to AutoG&P[2], where a new logic is introduced, which captures a set of possible valid reductions between games, thus allowing to partially explore automatically the possible proof trees, and enable the automation of a large class of schemes and protocols.

However, even if AutoG&P provides a complete framework for obtaining fully verified proofs, the automation part still relies heavily on some heuristics. The main issue is that each game transformation can be applied in an unbounded number of ways. It is then unclear what might be the correct proof search given a game. And even if we consider only one application of one rule, some of its conditions are sometimes so specific that we cannot decide if it is possible or not to apply it to a given game, leaving us once again using a heuristic. To say it in a nutshell, AutoG&P is currently using a heuristic to decide which rule to try to apply, and another heuristic will say if it is possible or not to apply this rule. The heuristics are fine tuned and allow to obtain automated or partially automated proofs for some complicated schemes, but replacing some of these heuristics by complete procedures seems to be necessary to progress further into the automation.

Current issues. One of the main issue correspond to the automated application of cryptographic assumptions. As we explain later, deciding whether or not a cryptographic assumption is applicable to a given a game requires to solve symbolic deducibility problems (can an attacker deduce a message given a set of known messages ?). This is actually quite surprising, while trying to automate proofs in the computational model, we suddenly fall back on a symbolic model problem studied many times before. However, the framework of AutoG&P is very general, capturing for instance protocols requiring a bilinear map, and the existing results on symbolic deducibility are not sufficient for AutoG&P. Indeed, we are here dealing with an equational theory, which allows to perform operations in a multivariate polynomial ring, given that we may have some extra information about some variables, typically that some of them are different from 0.

Contributions. We mainly present in this report new decision results, allowing to improve the automation of game transformations.

1) Our first contribution is to give a decision algorithm for deducibility in the AutoG&P setting, an axiomatized Diffie-Hellman equational theory containing a multi-linear map. This requires solving deducibility constraint in an extended theory of modular exponentiation for which no procedure was known before. We then use this algorithm to decide if an assumption can directly be applied to a given game. This has been implemented and experimented in AutoG&P.
2) We extend the previous decision algorithm to a fragment of game transformation rules : in addition to directly applying an assumption, we can also create in some cases a sequence of rules, which allows to apply the assumption. To do so we introduce a new problem, "modulo deducibility", which requires the attacker to be able to compute a message that has the same probabilistic distribution as the secret.
3) Our third contribution is to provide a criterion to detect when it is not possible to find a sequence of rules leading to an assumption application. This allows us, when we are not in the fragment of the previous algorithm, to decide whether or not we should try to find heuristically such a sequence of rules.
4) Finally we provide with an undecidability result, showing that there will always be a need for heuristics in the search for game transformations.

Plan. In section 2, we present the framework of AutoG&P and the general problem of applying a cryptographic assumption and how it relates to deducibility. In section 3 we prove the decidability of deducibility and give some details about the implementation in section 4. In section 5 we provide with a decision procedure for a restricted modulo deducibility and give in section 6 an example of its use on a cryptographic problem. Finally, in section 7 we provide a criterion used to detect when it will be impossible to apply an assumption to a given game even with many transformations, and in section 8 we conclude with an undecidability result showing that it is useless to look for some decision procedure.

$x : string$	variable
$g_i : \mathbb{G}_i$	generator of $\mathbb{G}_i$
$\_ * \_ : \mathbb{G}_i \times \mathbb{G}_i \mapsto \mathbb{G}_i, \_/\_ : \mathbb{G}_i \times \mathbb{G}_i \mapsto \mathbb{G}_i$	multiplication and division in $\mathbb{G}_i$
$(\_)^{(\_)} : \mathbb{G}_i \times \mathbb{F}_q \mapsto \mathbb{G}_i$	exponentiation in $\mathbb{G}_i$
$0 : \mathbb{F}_q, \_ + \_ : \mathbb{F}_q \times \mathbb{F}_q \mapsto \mathbb{F}_q, -(\_) : \mathbb{F}_q \mapsto \mathbb{F}_q$	additive group operations for $\mathbb{F}_q$
$1 : \mathbb{F}_q, \_ * \_ : \mathbb{F}_q \times \mathbb{F}_q \mapsto \mathbb{F}_q, (\_)^{-1} : \mathbb{F}_q \mapsto \mathbb{F}_q$	multiplicative group operations for $\mathbb{F}_q$
$\hat{e}(\_, ..., \_) : \mathbb{G}_1 \times ... \times \mathbb{G}_k \mapsto \mathbb{G}_t$	multi-linear map to $\mathbb{G}_t$

Figure 3. Signature of the equational theory $\mathcal{E}$

2. The AutoG&P framework

The goal of AutoG&P is to enable modeling and transforming many different types of games. The syntax must capture all the possible mathematical constructs that could be used in a scheme or protocol, such as group operations, bilinear map, logical operations on bitstring... To be able to manipulate such mathematical constructions and model these operations, we formally define the type of messages and values with a generic type : expressions.

Expressions. Expressions are built over a Diffie-Hellman equationnal theory $\mathcal{E}$ extended with a k-multilinear map whose signature is presented in figure 3. The source groups G_i are all finite groups of order $q \in \mathbb{N}$. A context C is an expression with a hole that can be filled by an other expression. We classically define an equivalence relation on expressions based on satisfaction in first-order logic. The expressions e and e' are equivalent modulo $\mathcal{E}$, written $e =_{\mathcal{E}} e'$, if $\mathcal{E} \models e = e'$. Here, $\mathcal{E}$ denotes the axioms for our signature consisting of the field axioms for $\mathbb{F}_q$, the (multilinear) group axioms for $\mathbb{G}_i$, and the usual axioms for the equality. We consider inversion in $\mathbb{F}_q$, as underspecified, i.e., 0^{-1} is some arbitrary fixed value in $\mathbb{F}_q$, and we can only simplify $x * x^{-1}$ to 1 if $x \neq 0$ holds. We use $\Gamma \models e =_{\mathcal{E}} e'$ to denote $(\Gamma, \mathcal{E}) \models e = e'$, i.e., $\mathcal{E}$ is extended with additional axioms Γ. We assume the set of axioms Γ consists of (in)equalities on expressions. We write $\mathcal{E} - X$ with X a function symbol to denote the equational theory generated by the equations of $\mathcal{E}$, which do not contains the symbols in X. Finally, we will use $\mathbb{K}$ to denote any field.

Example 1. We can have expressions of the form $e_1 = g_1^2$,$e_2 = \hat{e}(e_1, g_2^3)$ and $e_3 = g_t^5 \times e_2$. We have a direct matching between the equational theory and the classical group operations in mathematics, and here for example we have $e_3 = g_t^5 \times \hat{e}(g_1^2, g_2^3) = g_t^5 \times \hat{e}(g_1, g_2)^{2+3} = g_t^5 \times g_t^5 = g_t^{10}$ following from the characterization of a bilinear map.

For an environment $\sigma : string \mapsto \{0,1\}^*$, we can define the natural semantic of an expression, where $[\![expr]\!]_\sigma$ is the bitstring corresponding to the mathematical evaluation of the expression where we replace every variable with the corresponding value given by σ.

Now that we have defined the type of the objects, we can show the syntax of game. We take here a different approach from the AutoG&P syntax and choose to only consider a restricted fragment of games along with a new syntax. This is only for ease of presentation and not a limitation, we do not wish to reintroduce the full AutoG&P syntax, most of which would be irrelevant for the problems we consider. We thus define a new type to define games, game expressions, which corresponds to expressions extended with equality testing, conditional, random sampling and attacker call. An attacker call can either be a variable of type PPT, or an explicitly given attacker, i.e a closed game expression which can take inputs parameters. An explicitly given attacker can still make calls to some other PPT attackers, and we will call such an attacker a simulator. We write $a : D$ to denote the sampling of a in D.

Definition 2. A game expression is of the form:

$$\begin{aligned}
gexpr \;&:= expr\\
&\mid (expr_1 = expr_2) : bool\\
&\mid \text{if } gexpr_1 : bool \text{ then } gexpr_2 \text{ else } gexpr_3\\
&\mid a_1 : D_1, \dots a_m : D_m . gexpr\\
&\mid \mathcal{B}(gexpr_1, ..., gexpr_n)\\
\mathcal{B} \;&:= \mathcal{A}_k\\
&\mid \lambda e_1 \dots \lambda e_n . gexpr
\end{aligned}$$

In order to formally reason about games, we present the corresponding semantic, which correspond to the evaluation of a game expression in an environment σ. The environment will map all the variable to a bitstring, and will map an attacker symbol $\mathcal{A}_i$ to a PTT A_i and its random tape ρ_i. We give below the semantic of a game expression, assuming that σ gives a value for all the variables appearing in $gexpr$.

$$\begin{aligned}
[\![expr_1 = expr_2]\!]_\sigma &= \begin{cases} 1 \text{ if } ([\![expr_1]\!]_\sigma = [\![expr_2]\!]_\sigma \\ 0 \text{ or else} \end{cases}\\
[\![\text{if } gexpr_1 \text{ then } gexpr_2 \text{ else } gexpr_3]\!]_\sigma &= \begin{cases} [\![gexpr_2]\!]_\sigma \text{ if } [\![gexpr_1]\!]_\sigma \\ [\![gexpr_3]\!]_\sigma \text{ or else} \end{cases}\\
[\![\mathcal{A}_k(gexpr_1, ..., gexpr_n)]\!]_\sigma &= A_k([\![gexpr_1]\!]_\sigma, ..., [\![gexpr_n]\!]_\sigma) \text{ if } \sigma(\mathcal{A}_k) = A_k\\
[\![a_1 : D_1, \dots a_m : D_m . gexpr]\!]_\sigma &= [\![gexpr]\!]_\sigma \text{ if } \sigma(a_k) \in D_k\\
[\![\lambda e_1 \dots \lambda e_n . gexpr(gexp_1, ..., gexp_n)]\!]_\sigma &= [\![gexpr[^{e_1}/_{gexp_1}, ...,^{e_n}/_{gexp_n}]]\!]_\sigma \text{ if } gexpr \text{ closed}
\end{aligned}$$

We can use the syntactic sugar "let $x =$ _ in", because as we fix the randomness using σ, every call to the same attacker will return the same value. Then, we can have sequential calls to different attacker either by using nested calls or using a let binding. Finally, we can simply write a cryptographic game with a game expression, the semantic of the game expression matching the execution of the game.

We say that two games G1 and G2 are equivalent, denoted $G1 \simeq G2$ if they have the same, up to negligible, of giving the same output. More formally :

$$a_1 : D_1, ..., a_n : D_n . gexpr \simeq a'_1 : D'_1, ..., a'_m : D'_m . gexpr'$$

$$\Leftrightarrow$$

$$\forall (A_i), |Prob(\sigma, \sigma(a_k) \in D_k, rho_i : U \| [\![gexpr]\!]_\sigma = 0)$$

$$-Prob(\sigma, \sigma(a_k) \in D_k, rho_i : U \| [\![gexpr']\!]_\sigma = 0)| \text{ is negligible}$$

A cryptographic assumption is then given by simply assuming that two specific games are equivalent.

Example 3. The following equivalence expresses the so-called DDH assumption :

$$(a : \mathbb{F}_q, b : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^{ab})) \simeq (a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q . \mathcal{A}(g^a, g^b, g^c)$$

Here, we say that there is no attacker who will be able to return 0 more often if given g^{ab} rather than g^c. We say that no attacker can distinguish between (g^a, g^b, g^{ab}) and (g^a, g^b, g^c)

With this syntax, the (incomplete) AutoG&P logic then allows us to prove equivalences on games by describing valid game transformations. We only present a simplified formalism in figure 4 of a subset of the full logic, which can be found in [2], with some precondition omitted or simplified for readability.

We use $G[e]_p$ to denote the expression in G at position p and we write $G\{\mathcal{A} \mapsto B(\mathcal{A})\}$ to denote the game G in which we replace the attacker symbol $\mathcal{A}$ by a simulator B.. The SYM, DEQ and TRANS rules capture the fact that the $\simeq$ relation is symmetric, transitive and reflexive. Then, the SWAP rule states that the order of some independent commands in the game does not matter, the ADD rule allows one to add some sampling or binding to a game and SUBST allows to replace expressions by some other equal expression. There are two more

Probability judgment:

$$\textsc{Sym}\ \frac{G' \simeq G}{G \simeq G'} \qquad \textsc{Deq}\ \frac{}{G \simeq G} \qquad \textsc{Trans}(G')\ \frac{G \simeq G' \quad G' \simeq G''}{G \simeq G''}$$

Program transformation:

$$\textsc{Subst}(p,e)\ \frac{G[e]_p \simeq G'}{G[e']_p \simeq G'}\ \boxed{e =_{\mathcal{E}} e'}$$

Random sampling:

$$\textsc{Rnd}(x,e)\ \frac{G[{}^x/_e] \simeq G'}{G \simeq G'}\ \boxed{\text{x:var and e:expr have the same probabilistic distribution}}$$

Reduction:

$$\textsc{Abstract}(B)\ \frac{G \simeq G'}{G\{\mathcal{A} \mapsto B(\mathcal{A})\} \simeq G'\{\mathcal{A} \mapsto B(\mathcal{A})\}}\ \boxed{\text{B is a valid simulator for G and G'}}$$

Figure 4. Simplified fragment of the AutoG&P logic

$$H_1 = (a : \mathbb{F}_q, b : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^{ab})) \simeq H_2 = (a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c)$$

Figure 5. The DDH assumption

complex rules, RND allow for instance to replace a uniformly distributed variable by some uniformly distributed expression.

Finally, the last rules corresponds to the application of an assumption. Let us consider once again the DDH assumption in figure 5.

Applying this assumptions to a game means that if in the game there is a call to an attacker with expressions of the form (g^x, g^y, g^{xy}) with x and y random variables, then you can replace xy by a fresh random variable.

Example 4. Let us assume the DDH assumption of figure 5, we want to prove the 3-PDDH assumption :

$$G_1 = (a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{abc})) \simeq G_2 = (a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^d)$$

We obtain the following proof tree :

$$\textsc{Trans}(G')\ \frac{\textsc{Abstract}(B)\ \dfrac{DDH\ \dfrac{}{H_1 \simeq H_2}}{G_1 \simeq G'} \qquad \textsc{Rnd}(d, \dfrac{d}{c})\ \dfrac{Deq\ \dfrac{}{G_2 \simeq G_2}}{G' \simeq G_2}}{(G_1 \simeq G_2)}$$

with

$$G' = (a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q, d : \mathbb{F}_q.\mathcal{A}(g^a, g^b, g^c, g^{cd})$$

$$B = \lambda(A), \lambda e_1, e_2, e_3, d : \mathbb{F}_q, A(e_1, e_2, g^d, e_3^d)$$

The difficult part of the proof is to find the valid simulator for the Abstract rule. Given B, it is easy to check if the application is correct :

$$\begin{aligned}
H_1\{\mathcal{A} \mapsto B(\mathcal{A})\} &= a : \mathbb{F}_q, b : \mathbb{F}_q.(\lambda(A), \lambda e_1, e_2, e_3, d : \mathbb{F}_q, A(e_1, e_2, g^d, e_3^d))(\mathcal{A})(g^a, g^b, g^{ab}) \\
&= a : \mathbb{F}_q, b : \mathbb{F}_q.(\lambda e_1, e_2, e_3, d : \mathbb{F}_q, \mathcal{A}(e_1, e_2, g^d, e_3^d))(g^a, g^b, g^{ab}) \\
&= a : \mathbb{F}_q, b : \mathbb{F}_q.(d : \mathbb{F}_q, \mathcal{A}(g^a, g^b, g^d, g^{abd})) \\
&= G_1
\end{aligned}$$

$$\begin{aligned}
H_2\{\mathcal{A} \mapsto B(\mathcal{A})\} &= a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q.(\lambda(A), \lambda e_1, e_2, e_3, d : \mathbb{F}_q, A(e_1, e_2, g^d, e_3^d))(\mathcal{A})(g^a, g^b, g^c) \\
&= a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q.(\lambda e_1, e_2, e_3, d : \mathbb{F}_q, \mathcal{A}(e_1, e_2, g^d, e_3^d))(g^a, g^b, g^c) \\
&= a : \mathbb{F}_q, b : \mathbb{F}_q, c : \mathbb{F}_q.(d : \mathbb{F}_q, \mathcal{A}(g^a, g^b, g^d, g^{cd})) \\
&= G'
\end{aligned}$$

However, the converse is not true, because we need to check alpha renaming and deducibility in order to know if there is a simulator.

Given a game G, one can apply the DDH assumption by using the rule ABSTRACT, which requires to find a simulator B such that $G = H_1\{\mathcal{A} \mapsto B(A)\}$. Basically, the idea is to abstract away some part of the game G by saying that those operations can be made by an attacker that we must give explicitly. The simulator is then a valid simulator if it corresponds to an actual attacker : it cannot guess any secret value and every expression that it contains must be computed from its input parameters. It then introduces the classical problem of deducibility in the symbolic setting. Given a set of terms, can an attacker deduce another term ? More formally, we write $\Gamma \models e \vdash_{\mathcal{E}} e'$ if there exists a context C such that $\Gamma \models C[e] =_{\mathcal{E}} e'$. We see here that we have a sufficient condition in order to have a valid simulator. Indeed, if every expression in the simulator can be deduced from its inputs we then have contexts that allows us to construct the expressions. Therefore, we can construct the simulator using these contexts.

3. Unrestricted decidability of axiomatized deducibility constraints

Existing work concerning equational theories with Diffie-Hellman exponentiation always have some restriction. For instance, in [9] they only consider products in the exponents and in [11] they only considers polynomials with maximum degree of 1 (linear expressions). Here, we try to capture the most general possible instance of this problem, without any restriction on the terms known to the attacker and allowing the equational theory to be extended with axioms that are either equalities or inequalities between terms.

Adding inequalities is an important generalization, because in order to deduce the variable b from the set $\{a * b, a\}$, we need the assumption $a \neq 0$. In our case, when x is in $\mathbb{F}_q \setminus 0$ we insert the assumption $x \neq 0$ in the context.

Let X be a set of public names sampled in $\mathbb{F}_q$, Y be a set of private names sampled in $\mathbb{F}_q$, $f_1, ...f_k, h \in \mathbb{K}[X, Y]$ be a set of polynomials over both public and secret names and Γ be a coherent set of axioms.

Our deducibility problem is then to decide if :

$$\Gamma \models X, g_{i_1}^{f_1}, ..., g_{i_k}^{f_k} \vdash_{\mathcal{E}} g_t^h$$

We solve this problem in two steps, first by reducing to terms that are only in the target group, and then solving the problem using Groebner basis techniques. 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.

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 [15]. Their constructive proof can be used to obtain the following proposition.

Proposition 5. *For any sets X and Y, polynomials $f_1, ...f_n, h \in \mathbb{K}[X, Y]$ and groups elements $g_{i_1}^{f_1}, ..., g_{i_n}^{f_n}$,if we denote $(g_t^{e_i}) = \{\hat{e}(g_{i_j}, g_{i_k}) | 1 \leq j \leq k \leq n, g_{i_j} \in \mathbb{G}_1, g_{i_k} \in \mathbb{G}_2\} \cup \{\hat{e}(g_{i_j}, 1) | 1 \leq j \leq n, g_{i_j} \in \mathbb{G}_1, \} \cup \{\hat{e}(1, g_{i_j}) | 1 \leq j \leq n, g_{i_j} \in \mathbb{G}_2, \}$, 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.

Reduction to polynomials. The problem at hand is now reduced to solving a non linear multi-variate polynomial equation, as shown by the following proposition :

Lemma 6. *For any sets X and Y, polynomials $w_1, ...w_N, h \in \mathbb{K}[X, Y]$*

$$\Gamma \models X, g_t^{w_1}, ..., g_t^{w_N} \vdash_{\mathcal{E}} g_t^h \Leftrightarrow \exists (e_i, g_i) \in \mathbb{K}[X], (\forall i, \Gamma \models g_i \neq 0) \wedge \sum_i e_i \times \frac{w_i}{g_i} = h$$

The idea is that the attacker can only put a group element to a power that he fully knows in $\mathbb{K}$, so in the exponent he can only multiply by polynomials in $\mathbb{K}[X]$. The attacker can also divide by a polynomial if and only if the axioms implies that it is different from 0.

The first remark is 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 [16]). 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 7. *For any sets X and Y, polynomials $w_1, ...w_N, h \in \mathbb{K}[X, Y]$*

$$\Gamma \models X, g_t^{w_1}, ..., g_t^{w_N} \vdash_{\mathcal{E}} g_t^h \Leftrightarrow \exists (e_i) \in \mathbb{K}[X], (k_g) \in \mathbb{N}, \sum_i e_i \times w_i = h \prod_{g \in Gs} g^{k_g}$$

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 8. 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 9.

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

We are now going to prove that the previous membership problem is decidable. The Buchberger algorithm allows us to compute a Groebner basis of any free $\mathbb{K}[X]$-module [13] and then decide the membership problem. A Groebner basis is a generating set of a multivariate polynomial module such that when given an ordering on the monomials, the consecutive multivariate division following the ordering of a polynomial by the Groebner Basis elements yelds a unique remainder. Moreover, this remainder is 0 if and only if the polynomial belongs to the module, so we can use this to solve the membership problem.

Example 10. With $X = \{a, b\}$ and $Y = \{c\}$, let us consider the free $\mathbb{K}[X]$-module M generated by the following Groebner Basis :

$$GB = \{g_1 = c^2, g_2 = abc, g_3 = b^2\}$$

For any element of $\mathbb{K}[X, Y]$, we have that the multivariate division by GB yields a unique remainder. Let us consider $h = b^2c^2 + a^2bc + ab^2c + a^5b^3$:

$$\begin{aligned} h|c^2 &\rightarrow r_1 = a^2bc + ab^2c + a^5b^3 \\ r_1|abc &\rightarrow r_2 = a^5b^3 \\ r_2|b^2 &\rightarrow 0 \end{aligned}$$

We obtain that $h \in M$ with :

$$h = b^2g_1 + (a + b)g_2 + a^5bg_3$$

We then just have to be able to express the saturation of M as such a module, which is done in the following lemma.

Lemma 11. *For any sets X and Y, polynomials $f_1, ...f_n, h \in \mathbb{K}[X, Y]$, $g \in \mathbb{K}[X]$,let $M = \{\sum_i e_i \times w_i | e_i \in \mathbb{K}[X]\}$. Then, with t as a fresh variable :*

$$M :_{\mathbb{K}[X,Y]} g^\infty = \langle (f_i) \cup ((gt - 1)Y^{\vec{j}})_{\vec{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]} \cap \mathbb{K}[X, Y]$$

Thanks to the previous lemma, we finally get the decidability result :

Theorem 1. *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 3 that can be solved using lemma 4 by computing the $\langle (f_i) \cup ((gt - 1)Y^{\vec{j}})_{\vec{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]}$, keep only the elements of the base that are independent of t and then check 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 Groebner basis allows us to find generators of the corresponding syzygies (Theorem 15.10 of [13]), which actually captures all the possible distinguishers of a frame.

4. Implementation and example

In order to confirm our theoretical results, we decided to implement the Groebner Basis computation and replace in AutoG&P the heuristic for deducibility by the new decision procedure.

Many implementations of Groebner basis computations can be found online, but all of them are only usable for polynomial ideal, we thus implemented a version of the Buchberger algorithm for $\mathbb{K}[X]$-module and plugged it in AutoG&P.

Our main concern was the complexity of the final algorithm, because the saturation squares up the number of inputs terms and the Groebner Basis can be at worst a double exponential. However, we need to keep in mind that we are dealing with instances of our problem that come from protocols so they are relatively small instances. This was actually confirmed when all the examples of AutoG&P ran without any loss of speed with the new decision procedure rather than the heuristic.

Let us consider the run of the procedure on a toy example (which was not covered by the heuristic). We have here a bilinear map $\hat{e} : G_1 \times G_1 \mapsto G_t$, and the deducibility problem is :

$$\emptyset \models g_1^{c+d}, g_1^d \vdash_{\mathcal{E}} g_t^{c^2+c\times d+d^2}$$

The first step is the saturation (Proposition 5), where we basically compute all the possible application of the map, with for example $\hat{e}(g_1, g_1^d)$ or $\hat{e}(g_1^{c+d}, g_1^d)$. We obtain :

$$\emptyset \models g_t^{c+d}, g_t^d, g_t^{d^2}, g_t^{(c+d)^2}, g_t^{(c+d)\times d} \vdash_{\mathcal{E}-\hat{e}} g_t^{c^2+c\times d+d^2}$$

We will call $x_0 = c + d, ..., x_4 = (c + d) \times d$ the known terms. Here, we are in the case where we do not have any axioms, so we just have to compute the Groebner basis of the (x_i) and try to reduce the secret (Lemma 7 with $G = \emptyset$).

To compute the Groebner Basis, we will consider the lexical ordering on the degrees of the monomials, with the variable d leading. For example : $d^2 \times c > d \times c^5$, because $(2, 1) >_{lex} (1, 5)$. Then, the idea behind the Buchberger algorithm is to start with a set of polynomials and consider every combination of two polynomials, which allows to introduce a new leading monomial. Then, we add those new polynomial and repeat the process until a fixed point is reached. We can see that the max degree of new polynomials introduced is strictly decreasing, hence the termination.

Example 12. For example, if we consider the pair $(d, d + c)$, we have that $d - (d + c) = c$, so c will be part of the Groebner Basis. When starting with the set (x_i), we finally get the following Groebner Basis :

$$\{x_0, ..., x_4\} \xrightarrow{x_0 \leftarrow x_1 - x_0} \{c, d, d^2, (c + d)^2, d(c + d)\} \xrightarrow{x_4 \leftarrow x_4 - x_2} \{c, d, d^2, (c + d)^2, dc\}$$
$$\xrightarrow{x_3 \leftarrow x_3 - 2x_4} \{c, d, d^2, c^2 + d^2, dc\} \xrightarrow{x_3 \leftarrow x_3 - x_2} \{c, d, d^2, c^2, dc\}$$

Finally, we can try the successive multivariate division of $d^2 + c \times d + c^2$. The division will start by trying to remove the leading monom, so the first step will be :

$$d^2 + c \times d + c^2 - x_2 = c \times d + c^2$$

Then, we continue by dividing the remainder, and we obtain :

$$c \times d + c^2 - x_4 = c^2 = x_3$$

This means that our polynomial is deducible, and from the Groebner Basis we obtain the context :

$$d^2 + c \times d + c^2 = x_2 + x_4 + x_3$$

Finally, reintroducing the bilinear map and going back to the original (x_i), we obtain a valid solution of the deducibility problem :

$$g_t^{c^2+c\times d+d^2} = \hat{e}(g_1^d, g_1^d) \times \hat{e}(g_1^{c+d}, g_1^d)^{-1} \times \hat{e}(g_1^{c+d}, g_1^{c+d})$$

5. Decidabilty of the modulo deducibility

Now, we extend the previous section, trying to decide the system that includes both RND and ABSTRACT, where RND allows to replace a term with another one that defines the same distribution.

Looking for any equivalently distributed set of terms is difficult. Therefore we consider a restricted application, in which the term u, defining the same distributions as t, is obtained from t by applying an invertible context which only depends on random variables.

Following this idea, we now introduce the following definitions :

Definition 13. Given a set of random variables $X = (x_i)$ and of variables Y, we define $Inv(X \cup Y, x_i)$ as the set of invertible contexts that only depend on variables in X that are independent from x_i.

Given this set of contexts, we can now define the modulo deducibility problem :

Definition 14. Given a set of terms $t_1, ..., t_n, h_1, ..h_m$ ranging over random variables $X = (x_i)$ and variables $Y = (y_i)$, we say that $t_1, ..., t_n \vdash_{\mathcal{E}}^* h_1, ..., h_m$ if and only if :

$$\exists (C_i) \in Inv(X \cup Y, x_i), t_1, ..., t_n \vdash_{\mathcal{E}} (h_1, ..., h_m)[^{C_i(x_i)}/_{x_i}]$$

Restriction. Solving the previous problem in general is quite complex, because if for instance some h contains random variables squared, we then also square the context. Thus capturing all the possible sets of terms is difficult. We therefore consider a restricted instance of the previous problem, where the degree max of the random variable in h is 1, i.e h is of the form $h = px + q$ with p and q polynomials, and where we only consider context of the form $C[x] = ux + g$ with u and g polynomials over variables independent from x.

With this hypothesis the modulo deducibility problem can then be expressed as a mathematical problem. Formally, we are given X and Y two sets of variables, and a set of random variables $T = \{t_1, ..., t_m\} \subset X \cup Y$. Let us call $\mathcal{I}_x$ the set of variables that are independent from x. Then, given a set of polynomials $f_1, ..., f_n, q \in \mathbb{K}[X, Y]$ and some polynomials $p_1, ...p_m \in \mathbb{K}[X \cup Y \setminus T]$, we are trying to decide if :

$$g^{f_1}, ..., g^{f_n} \vdash_{\mathcal{E}}^{*} g^{\sum_k p_k \times t_k + q}$$

Reducing to a polynomial problem. Using the same techniques as in the previous lemmas, we obtain the following reduction into a mathematical problem

Lemma 15. *For any set X, Y and $T = \{t_1, ..., t_m\} \subset X \cup Y$, given a set of polynomials $f_1, ..., f_n, q \in \mathbb{K}[X, Y]$ and $p_1, ...p_m \in \mathbb{K}[X \cup Y \setminus T]$, we have :*

$$g^{f_1}, ..., g^{f_n} \vdash_{\mathcal{E}}^{*} g^{\sum_k p_k \times t_k + q}$$

$$\Leftrightarrow$$

$$\exists e_i \in \mathbb{K}[X], u_t, g_t, v_t \in \mathbb{K}[\mathcal{I}_t], \sum e_i f_i = \sum_k p_k (\frac{u_{t_k} \times t_k + g_{t_k}}{v_{t_k}}) + q$$

Given an instance of this problem, we consider the sets :

$$M = \{\sum e_i f_i | e_i \in \mathbb{K}[X], \}$$

$$N_{t_k} = \{p_{t_k} u \times t_k + p_{t_k} g | u \in \mathbb{K}(\mathcal{I}_{t_k})^{*} g \in \mathbb{K}(\mathcal{I}_{t_k})\}$$

With these notation, we obtain the following result :

Lemma 16.

$$\exists e_i \in \mathbb{K}[X], u_t, v_t \in \mathbb{K}[\mathcal{I}_t] \setminus 0, g_t \in \mathbb{K}[\mathcal{I}_t], \sum e_i f_i = \sum_k p_k (\frac{u_{t_k} \times t_k + g_{t_k}}{v_{t_k}}) + q$$

$$\Leftrightarrow$$

$$M \cap \langle (N_t), q \rangle_{\mathbb{K} \setminus 0} \neq \emptyset$$

We are now faced with linear spans of modules, so we need to generalize the Groebner basis techniques :

Lemma 17. *The union of the Groebner basis of free disjoint Y_i-modules S_i is a Groebner basis of the linear span of the Y_i-module.*

We can now have the decidability result for one secret but we need to consider the case where we have to deduce a tuple. Indeed, to apply the assumption, all the terms in the simulator must be deducible at the same time, i.e with the same bijection applied to the random variables. With the previous result, we use some encoding technique to bind the bijection to be identical for every secret.

Lemma 18. *For any set X, Y and $T = \{t_1, ..., t_m\} \subset X \cup Y$, given a set of polynomials $f_1, ..., f_n, q \in \mathbb{K}[X, Y]$ and for $1 \leq j \leq l$, $q_j, p_{j,1}, ...p_{j,m} \in \mathbb{K}[X \cup Y \setminus T]$, if we set $\nu_1, ..., \nu_m$ to be fresh variable, we have that :*

$$\exists e_{j,i} \in \mathbb{K}[X, \nu_j], u_t, g_t, v_t \in \mathbb{K}[\mathcal{I}_t], \sum e_{j,i} f_i = \sum_k (\sum_j \nu_j \times p_{j,k}) (\frac{u_{t_k} \times t_k + g_{t_k}}{v_{t_k}}) + (\sum_j \nu_j \times q_j)$$

$$\Leftrightarrow$$

$$\exists e_{j,i} \in \mathbb{K}[X], u_t, g_t, v_t \in \mathbb{K}[\mathcal{I}_t], \begin{cases} \sum e_{1,i} f_i = \sum_k p_{1,k}(\frac{u_{t_k} \times t_k + g_{t_k}}{v_{t_k}}) + q_1 \\ ... \\ \sum e_{l,i} f_i = \sum_k p_{l,k}(\frac{u_{t_k} \times t_k + g_{t_k}}{v_{t_k}}) + q_l \end{cases}$$

Intuitively, the idea is to try to deduce the secret $\nu_1 h_1 + ... + \nu_l h_l$, we then get only one bijection for every random variable and thanks to the fresh variables ν we can directly split the solution to obtain one for every secret h_l.

We then finally get the intended decidability result :

Proposition 19. *For any set X, Y and $T = \{t_1, ..., t_m\} \subset X \cup Y$, given a set of polynomials $f_1, ..., f_n, q \in \mathbb{K}[X, Y]$ and for $1 \leq j \leq l$ $q_j, p_{j,1}, ...p_{j,m} \in \mathbb{K}[X \cup Y \setminus T]$*

$$X, g^{f_1}, ..., g^{f_n} \vdash^*_{\mathcal{E}} (g^{\sum_k p_{j,k} \times t_k + q_j})_{1 \leq j \leq l} \textit{ is decidable}$$

Proof. Using lemma 18 we first reduce the problem to the one where we only have one secret. Then, using lemma 15 and 16 , we reduce the problem to deciding the emptiness of the intersection of the K[X]-module M and the linear span of the $\mathbb{K}(\mathcal{I}_t])[1]$-modules N_t which we can solve with lemma 17. □

6. Example

In this section, we come back to AutoG&P and provide with an example of proofs.

We are given a game G that performs sampling of variables (x_i) at its start, and we want to apply to it an assumption B that samples the variables (y_i) and call an adversary with the terms (t_i). We propose the algorithm in figure 6 to obtain useful applications of the assumption on G.

```
For each mapping $\sigma$ from $(x_i)$ to $(y_i)$:
  Applying the mapping to G
  Put all samplings of the $(y_i)$ at the start of G.
  Let $(h_i)$ be the terms appearing in G
  Let $(u_i)$ be the random variables in G
  and $(U_i)$ their respective independant variables.
  For each subset $(u'_i)$ of $(u_i)$:
    If $(h_i)$ are all linear w.r.t $(u'_i)$:
      For each subset $(t'_i)$ of $(t_i)$:
        If $(t'_i) \vdash^*_{\mathcal{E}} (h_i)$:
          Let G′ be the game obtained after applying B.
          If G′ $\neq$ G, return G′
        Endif
      Endfor
    Endif
  Endfor
Endfor
```

Figure 6. Heuristic for assumption application

Intuitively, given a game and an assumption, there can be many different mapping from the variables of the assumptions to the variables of the game, so we are going to try all possible cases. Moreover, we may not need to apply a context to all the random variables, and some terms may not be linear for one subset of random variables but linear for another, so we also try them all. Once we have a mapping and random variables such that we obtain a valid modulo deducibility problem, we can try to solve it. However, some application of B might be useless. For instance, applying the DDH assumption on a simulator that does not use its third components is useless. Therefore, we are going to try to solve the modulo deducibility for any subset of the public terms, in order to force the simulator to use the important terms. Finally, we just check if the application of the assumption is useful before returning the new game.

6.1. DLIN to BDDH

We will now show an example of the algorithm running on a basic example. We consider the assumption :

$$s_1, s_2, r_1, r_2 : \mathbb{F}_q.\mathcal{A}(g, g^{s_1}, g^{s_2}, g^{s_1 r_1}, g^{s_2 r_2}, g^{r_1+r_2}) \simeq s_1, s_2, r_1, r_2, r : \mathbb{F}_q.\mathcal{A}(g, g^{s_1}, g^{s_2}, g^{s_1 r_1}, g^{s_2 r_2}, g^{r})$$

Intuitively, the DLIN assumption allows us to replace $r_1 + r_2$ by a fresh random variables in a game.

We want to apply it to the BDDH game:

$$a, b, c : \mathbb{F}_q.\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc})$$

The first step is to choose a mapping from the variables of the game to the variables of the assumption. We consider here the mapping $a \mapsto s_1, b \mapsto s_2, c \mapsto r_1$. We then need to check if there is a valid simulator B such that

$$B(g^{s_1}, g^{s_2}, g^{s_1 r_1}, g^{s_2 r_2}, g^{r_1+r_2}) = \mathcal{A}(g^{s_1}, g^{s_2}, g^{r_1}, g_t^{s_1 s_2 r_1})$$

We thus obtain the following modulo deducibility problem :

$$g^{s_1}, g^{s_2}, g^{s_1 r_1}, g^{s_2 r_2}, g^{r_1+r_2} \vdash_{\mathcal{E}}^{*} g^{s_1}, g^{s_2}, g^{r_1}, g_t^{s_1 s_2 r_1}$$

Solving the problem using our algorithms yields the RND application $r_1 \mapsto r_1 + r_2$, which is correct as

$$\hat{e}(g^{s_2}, g^{s_1 r_1})\hat{e}(g^{s_1}, g^{s_2 r_2}) = g_t^{s_1 s_2 (r_1+r_2)} = g_t^{s_1 s_2 r_1}[^{r_1+r_2}/_{r_1}]$$

Finally, by using first RND, then ABSTRACT and then the axiom corresponding to DLIN, we prove the statement corresponding to the BDDH assumption:

$$s_1, s_2, r_1 : \mathbb{F}_q.\mathcal{A}(g, g^{s_1}, g^{s_2}, g^{r_1}, g^{s_2 r_2 r_1}) \simeq s_1, s_2, r_1, r_2 : \mathbb{F}_q.\mathcal{A}(g, g^{s_1}, g^{s_2}, g^{r_1}, g^{r_2})$$

We here automatically proved that the BDDH assumption is a consequence of the DLIN.

7. Necessary criterion for the application of an assumption

We presented in the previous section an algorithm which is only complete for some instance of the inference rules. Therefore, we sometimes need to try to find heuristically a sequence of transformations which will finally allows us to apply the assumption. This search can be time consuming, and it can be useful to detect quickly when there is no hope to find such a sequence. We will therefore try to identify a way to detect if it is impossible to apply an assumption to a game.

Given a game G containing attacker calls $A_1(g^{t_{11}}, ..., g^{t_{1n_1}}), ..., A_m(g^{t_{m_1}}, ..., g^{t_{mn_m}})$, we denote $ExA(G) = \{t_{i_j} | 1 \leq i \leq m, 1 \leq j \leq n_i\}$. Less formally, ExA(G) represents all the arguments that are given to the attackers called in G. We recall that the syzygy of a set of terms is the set of relations between those terms, and we denote it Syz. Formally, given a ring S:

$$Syz_S(f_1, ..., f_n) = \{(e_i)_{1 \leq i \leq n} \in S | \sum e_i \times f_i = 0\}$$

With those notations, we define an interesting reduction.

Definition 20. Given a game G with exponential terms over S, we say that $G \approx G'$ is an interesting reduction if and only if :

$$Syz_S(ExA(G')) \subsetneq Syz_S(ExA(G))$$

The intuition behind this definition is that when we do a reduction, we want to reduce the set of possible knowledge of the attacker. A direct way to do this is to reduce the different relations that exist between the terms, and this corresponds exactly to reducing the syzigies of the terms. At first, just considering the relations between terms and not the terms themselves might sound strange, but for example we may have a game with an attacker call $A(\gamma_1, ..., \gamma_n)$ and an assumption which allows us to replace γ_n by a fresh random variable. Then, if γ_n was previously complicated we may think that we won something, but actually we won something only if γ_n was related to some other γ. In the other case, from the point of view of the attacker, γ_n was already equivalent to a random variable, and thus the reduction did not allow us to make any progress.

From this definition, we then want to try to identify a criterion allowing us to detect efficiently in what cases it is impossible to do an interesting reduction by applying a specific assumption.

We consider that we want to apply the hypothesis $G_1 \approx G_2$ to some game G.

Let us assume we do have an interesting application of G_1 onto G. Then, we have B such that :

$$Syz_S(ExA(G_2\{\mathcal{A} \mapsto B(\mathcal{A})\})) \subsetneq Syz_S(ExA(G_1\{\mathcal{A} \mapsto B(\mathcal{A})\}))(1)$$
$$G \approx G_1\{\mathcal{A} \mapsto B(\mathcal{A})\}(2)$$

We are now going to focus on the case where the terms are only group elements, i.e we can obtain an algebraic criterion.

7.1. Sub case : Polynomial assumption

We are given as set of variables X, a set of variable Y and polynomials $P_1, ..., P_n, c \in \mathbb{K}[\bar{X}]$,$Q_1, ..., Q_m \in \mathbb{K}[\bar{U}]$. We then consider :

$$\text{assumption : } (X : \mathbb{F}_q.A_1(g^{P_1}, ..., g^{P_n})) \simeq (X : \mathbb{F}_q.); A_2(g^{P_1}, ..., g^{P_{n-1}}, g^c))$$

$$\text{game : } G = Y : \mathbb{F}_q.A(g^{Q_1}, ..., g^{Q_m})$$

We have an interesting application of the assumptions on G only if :

$$\exists e_{i_j} \in \mathbb{K}[\bar{Z}], \exists R \in Syz_{\mathbb{K}[\bar{U}]}((Q_i)_{1 \leq i \leq m})$$
$$\begin{cases} R((\sum_j e_{i_j} P_j)_{1 \leq i \leq m}) \\ \neg R((\sum_{j \leq n-1} e_{i_j} P_j + e_{i_n} c)_{1 \leq i \leq m}) \end{cases}$$

Intuitively, the relation R corresponds to the relation extracted from equation (1) which must be satisfied on one side and not the other in order to have the strict inclusion.

$R((\sum_j e_{i_j} P_j)_{1 \leq i \leq m})$ can be written under the form $\exists e_i \in \mathbb{K}[\bar{Z}], 0 = \sum e_k R'_k$ where every R'_k is a product of P_i. This is interesting because we then become independent from the contexts.

If we do have an interesting application, then

$$Syz((R'_k))_{\mathbb{K}[\bar{Z}]} \setminus Syz_{\mathbb{K}[\bar{Z}]}((R'_k)[^c/_{P_n}]]) \neq \emptyset$$

We obtain here a way to detect if it is possible or not to apply an assumption to a game. This is an algebraic criterion that is applicable only to game with only polynomials term, but we can also extract from other general games a sub case on which this criterion may yield result.

7.2. Examples

Many classical assumptions only consider polynomials in exponent, with for example :

- DDH: a, b, ab
- k-lin: $x_1, ..., x_k, r_1 x_1, ..., r_k x_k, (r_1 + ... + r_k)$
- BDDH: $x, y, z, xy, xz, yz, x^2, y^2, z^2, xyz$
- k-PDDH (Party) : $x_1, ..., x_k, (x_1 ... x_k)$
- k-EDDH (Exp) : x, x^k

We will use our criterion to see if there is a possible reduction between 3-PDDH and DDH

3-PDDH on DDH. We want to apply the 3-PDDH assumption on DDH, so we have : $P_1 = a, P_2 = b, P_3 = c, P_4 = abc, Q_1 = a, Q_2 = b, Q_4 = ab$ Then, there is only one $\exists R \in Syz_{\mathbb{K}[a,b]}((Q_i)_{1 \leq i \leq m})$, with $R : Q_3 - Q_1 Q_2 = 0$. Thus, R is captured by

$$\exists e_i, f_i, g_i \in \mathbb{K}[\bar{Z}], (\sum e_i P_i) - (\sum f_i P_i)(\sum g_i P_i) = 0$$

Once we devellop the relation, we obtain that :

$$(R'_k) = (P_1, P_2, P_3, P_4) \cup P_i P_j | 1 \leq i \leq j \leq 4$$

So we consider the two sets :

$$Syz_{\mathbb{K}[\bar{Z}]}(a, b, c, abc, a^2, b^2, c^2, (abc)^2, ...)$$
$$Syz_{\mathbb{K}[\bar{Z}]}(a, b, c, d, a^2, b^2, c^2, d^2, ...)$$

If we do the actual computation using the Buchberger algorithm which allows to compute the syzygies, we see that the difference is empty. More intuitively, with a degree argument we can see that replacing abc by d does not introduce a new relation, because we only replace unrelated terms of degree 4 by terms of degree two with a fresh variable, and thus unrelated to others.

In conclusion, there is no sequence of rule of AutoG&P such that 3-PDDH can be applied to DDH in an interesting way. This does not mean that in general DDH is not a consequence of 3-PDDH because AutoG&P is not complete, but it means that it is useless to try to find a proof of DDH using 3-PDDH in the AutoG&P logic.

We thus obtained a criterion that can be used to decide whether or not we should try to prove heuristically a statement when it does not fall in the complete fragment of our algorithm for the application of an assumption.

8. Undecidability of the AutoG&P logic

We presented previously two ways to guide the proof search in AutoG&P. There is still a lot of work that can be done in order to improve these heuristics, but this also raises the question : can we find a decision procedure ? The theorem bellow provides with a negative answer :

Theorem 2. *There is no decision procedure for the AutoG&P logic.*

The full encoding of the halting problem into this problem can be found in appendix B.

9. Conclusion

Using Groebner Basis and symbolic methods techniques, we gave in this report two new ways to improve the proof search in the AutoG&P framework. A natural next step is to keep improving these heuristics as much as we can. However, we need to remember that this model is incomplete, and even in this incomplete model there is no decision procedure for the proof search. These two facts may point in an another direction : to enable automation in the computational model, we might need a completely different approach. New models might be developed in the future, but it is a difficult path.

References

[1] G. Barthe et al. *Formal Certification of Code-Based Cryptographic Proofs*. Cryptology ePrint Archive, Report 2007/314. http://eprint.iacr.org/2007/314. 2007.

[2] Gilles Barthe, Benjamin Grégoire, and Benedikt Schmidt. "Automated Proofs of Pairing-Based Cryptography". In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: ACM, 2015, pp. 1156–1168. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813697. URL: http://doi.acm.org/10.1145/2810103.2813697.

[3] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella-Béguelin. "Formal Certification of Code-Based Cryptographic Proofs". In: *36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009*. ACM, 2009, pp. 90–101. URL: http://dx.doi.org/10.1145/1480881.1480894.

[4] Gilles Barthe et al. "Computer-Aided Security Proofs for the Working Cryptographer". In: 2011, pp. 71–90.

[5] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *Advances in Cryptology - EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006. Proceedings*. Ed. by Serge Vaudenay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 409–426. ISBN: 978-3-540-34547-3. DOI: 10.1007/11761679_25. URL: http://dx.doi.org/10.1007/11761679_25.

[6] Bruno Blanchet. "A Computationally Sound Automatic Prover for Cryptographic Protocols". In: *Workshop on the link between formal and computational models*. Paris, France, June 2005.

[7] Bruno Blanchet. *A Computationally Sound Mechanized Prover for Security Protocols*. Cryptology ePrint Archive, Report 2005/401. http://eprint.iacr.org/2005/401. 2005.

[8] Bruno Blanchet. "Automatic Verification of Security Protocols in the Symbolic Model: the Verifier ProVerif". In: *Foundations of Security Analysis and Design VII, FOSAD Tutorial Lectures*. Ed. by Alessandro Aldini, Javier Lopez, and Fabio Martinelli. Vol. 8604. 2014, pp. 54–87.

[9] Yannick Chevalier et al. "FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science: 23rd Conference, Mumbai, India, December 15-17, 2003. Proceedings". In: ed. by Paritosh K. Pandya and Jaikumar Radhakrishnan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. Chap. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents, pp. 124–135. ISBN: 978-3-540-24597-1. DOI: 10.1007/978-3-540-24597-1_11. URL: http://dx.doi.org/10.1007/978-3-540-24597-1_11.

[10] D. Dolev and A. C. Yao. "On the Security of Public Key Protocols". In: *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*. SFCS '81. Washington, DC, USA: IEEE Computer Society, 1981, pp. 350–357. DOI: 10.1109/SFCS.1981.32. URL: http://dx.doi.org/10.1109/SFCS.1981.32.

[11] 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*. 2014, pp. 217–231. DOI: 10.1109/CSF.2014.23. URL: http://dx.doi.org/10.1109/CSF.2014.23.

[12] Nancy Durgin, John Mitchell, and Dusko Pavlovic. "A Compositional Logic for Proving Security Properties of Protocols". In: *J. Comput. Secur.* 11.4 (July 2003), pp. 677–721. ISSN: 0926-227X. URL: http://dl.acm.org/citation.cfm?id=959088.959095.

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

[14] Steve Kremer and Robert Künnemann. "Automated Analysis of Security Protocols with Global State". In: *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. 2014, pp. 163–178. DOI: 10.1109/SP.2014.18. URL: http://dx.doi.org/10.1109/SP.2014.18.

[15] Steve Kremer, Antoine Mercier, and Ralf Treinen. "Reducing Equational Theories for the Decision of Static Equivalence". In: *Journal of Automated Reasoning* 48.2 (2012), pp. 197–217. DOI: 10.1007/s10817-010-9203-0. URL: https://hal.inria.fr/inria-00636797.

[16] A.K. Lenstra. “Factoring multivariate polynomials over finite fields”. In: *Journal of Computer and System Sciences* 30.2 (1985), pp. 235 –248. ISSN: 0022-0000. DOI: http://dx.doi.org/10.1016/0022-0000(85)90016-9. URL: http://www.sciencedirect.com/science/article/pii/0022000085900169.

[17] Gavin Lowe. “An attack on the Needham-Schroeder public-key authentication protocol”. In: *Information Processing Letters* 56.3 (1995), pp. 131 –133. ISSN: 0020-0190. DOI: http://dx.doi.org/10.1016/0020-0190(95)00144-2. URL: http://www.sciencedirect.com/science/article/pii/0020019095001442.

[18] Simon Meier et al. “The TAMARIN Prover for the Symbolic Analysis of Security Protocols”. In: *Proceedings of the 25th International Conference on Computer Aided Verification*. CAV13. Saint Petersburg, Russia: Springer-Verlag, 2013, pp. 696–701. ISBN: 978-3-642-39798-1. DOI: 10.1007/978-3-642-39799-8_48. URL: http://dx.doi.org/10.1007/978-3-642-39799-8_48.

[19] Roger M. Needham and Michael D. Schroeder. “Using Encryption for Authentication in Large Networks of Computers”. In: *Commun. ACM* 21.12 (Dec. 1978), pp. 993–999. ISSN: 0001-0782. DOI: 10.1145/359657.359659. URL: http://doi.acm.org/10.1145/359657.359659.

[20] Adam Petcher and Greg Morrisett. “The Foundational Cryptography Framework”. In: *Principles of Security and Trust - 4th International Conference, POST*. Ed. by Riccardo Focardi and Andrew C. Myers. Vol. 9036. Lecture Notes in Computer Science. Springer, 2015, pp. 53–72.

[21] Victor Shoup. “OAEP Reconsidered”. In: *Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 239–259. ISBN: 978-3-540-44647-7. DOI: 10.1007/3-540-44647-8_15. URL: http://dx.doi.org/10.1007/3-540-44647-8_15.

[22] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Report 2004/332. 2004.

[23] Nikhil Swamy et al. “Secure distributed programming with value-dependent types”. In: *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming*. Ed. by Manuel M. T. Chakravarty, Zhenjiang Hu, and Olivier Danvy. ACM, 2011, pp. 266–278. ISBN: 978-1-4503-0865-6. DOI: 10.1145/2034773.2034811. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=150012.

Appendix

1. Proof of section 3 and 5

Lemma 9.

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

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 $\sum_i e_i \times w_i = h \prod_{g \in Gs} g^{k_g}$. So, with $K = max(k_g)$, we just have to multiply both side 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 :

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

□

Lemma 11. *For any sets X and Y, polynomials $f_1, ...f_n, h \in \mathbb{K}[X,Y]$, $g \in \mathbb{K}[X]$,let $M = \{\sum_i e_i \times w_i | e_i \in \mathbb{K}[X]\}$. Then, with t as a fresh variable :*

$$M :_{\mathbb{K}[X,Y]} g^\infty = \langle (f_i) \cup ((gt-1)Y^{\vec{j}})_{\vec{j} \in \{deg_Y(f_i)\}} \rangle_{\mathbb{K}[X,t]} \cap \mathbb{K}[X,Y]$$

Proof. $\subset$
Let there be $v \in M :_{\mathbb{K}[X,Y]} g^\infty = \{x \in \mathbb{K}[X,Y] | \exists k \in \mathbb{N}, g^k \times x \in M\}$. 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, $g^k t^k v \in MK[X,t]$, 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 $(\vec{j_i}) \subset \{deg_Y(f_i)\}$ such that

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

Finally, we have

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

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

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

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

$$v = \sum_i e_i(X, \sum e_i f_i = \sum_k p_k(\frac{u_{t_k} \times t_k + g_{t_k}}{v_{t_k}}) frac1g) f_i$$

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 then finally have that :

$$g^k \times v = \sum_i e''_i f_i \in M$$

Which means that :

$$v \in M :_{\mathbb{K}[X,Y]} g^\infty$$

□

2. Proof of 8

We are given a Turing machine $(Q, \Gamma, B, \Sigma, q_0, \delta, F)$ and an initial configuration (w_0, q_0, w_1). For every symbol in Γ and in Q, we consider the homonym variable picked at random in the field $\mathbb{K}$. Then, we define an encoding of word over $\Gamma \cup Q$ using tuples:

$$\forall w \in (\Gamma \cup Q)^*, \hat{w} = \begin{cases} a \text{ } \textit{if } w = a \in (\Gamma \cup Q) \\ (\hat{t}, a) \text{ } \textit{if } w = a.t \end{cases}$$

We define $\bar{w} = \hat{w_m}$ with w_m being the mirror of w. For a set of symbol $T = (t_1, ..., t_n)$, we define $Pick(T)$ such as the set of all possible sequence commands such that we at least sample all the t :

$$Gen = \{a : D\}^*$$

$$Pick(T) = \{G | G \in Gen \wedge \forall t \in T, (t : \mathbb{F}_q) \in G\}$$

We add to the equationnal theory a free private function symbol f and for any configuration (w'_0, q', w'_1), we define a corresponding class of game :

$$Conf(w'_0, q', w'_1) = RND^*(\{Init; e = \mathcal{A}(\Gamma \cup Q); Finish;; | e =_{\mathcal{E}} f(\hat{w'_0}, q', \bar{w'_1}) \wedge Init \in Pick(\Gamma \cup Q), Finish \in Gen\})$$

Lemma 21. *For any configuration (w'_0, q', w'_1), $Conf(w'_0, q', w'_1)$ is stable by SUBST and RND.*

Proof. Let there be G = $Init; f(\hat{w'_0}, q', \bar{w'_1}) \leftarrow \mathcal{A}(); Finish;; \in Conf(w'_0, q', w'_1)$

- SUBST The class of game is defined modulo the equational theory, so substituting terms is not a problem. We also remark, that as f is a free function symbol, only trivial substitution are possible.
- RND Trivial by the definition.

□

Lemma 22. *For any configurations $(w'_0, q', w'_1) \neq (w''_0, q'', w''_1)$,*

$$Conf(w'_0, q', w'_1) \cap Conf(w''_0, q'', w''_1) = \emptyset$$

.

Proof. Let us call $\mathcal{S}((C_t)_{t \in T}) = \{{}^{C'_t(t)}/_t | t \in T\}$ the set of substitution defined by a list of context. If we have an element in both classes, then we have a term e and two invertible contexts C_t,C'_t for all $t \in \Gamma \cup Q$ such that :

$$f(\hat{w'_0}, q', \bar{w'_1})[\mathcal{S}((C_t))] =_{\mathcal{E}} e =_{\mathcal{E}} f(\hat{w''_0}, q'', \bar{w''_1})[\mathcal{S}((C'_t))]$$

By applying the inverse substitution of each of the C_t on the equation, we obtain new invertible contexts C_t'' such that :

$$f(\hat{w_0'}, q', \bar{w_1'}) =_{\mathcal{E}} f(\hat{w_0''}, q'', \bar{w_1''})[\mathcal{S}((C_t''))]$$

f is a free function symbol, so we have $\hat{w_0'} =_{\mathcal{E}} \hat{w_0''}[\mathcal{S}((C_t''))] \wedge q' =_{\mathcal{E}} q''[\mathcal{S}((C_t''))] \wedge \bar{w_1'} =_{\mathcal{E}} \bar{w_1''}[\mathcal{S}((C_t''))]$. On the left hand side, we only have tuple of symbols, so the only possible contexts are the identity, and then the configurations must be equal which is a contradiction. □

By adding the correct assumptions, we will show that the halting problem is equivalent to finding a proof for a security experiment. The core assumption is that two consecutive configurations must be equivalent. So, for any transition in δ of the form $(q, a) \mapsto (q', a', \rightarrow)$ and any letter b, we add the assumption :

$$T : \mathbb{F}_q.f((left, a), q, (b, right)) = \mathcal{A}(f(w_0, q_0, w_1)) \simeq T : \mathbb{F}_q.f(((left, a'), b), q', right)) = \mathcal{A}_{O_1}(f(w_0, q_0, w_1))$$

O_1 is the oracle which contains the full delta table and on any input $f(w_0', q_0', w_1')$ gives back the next configuration. We also add similar assumptions for the other types of transitions.

Proposition 23. *Let there be (B, q', B) a final configuration.*

$$(w_0, q, w_1) \rightarrow^* (B, q', B) \Leftrightarrow \exists C_1 \in Conf(w_0, q, w_1), C_2 \in Conf(B, q', B).C_1 \simeq C_2$$

Proof. $\Rightarrow$

Let us assume that $(w_0, q, w_1) \rightarrow^\delta (w_0', q', w_1')$ in one transition. Then, we take the simplest representative of each classes where Init only contains the necessary samplings and Finish is empty and considering the assumption corresponding to δ, we obtain the result by applying ABSTRACT,SYM,ABSTRACT and using the assumption. Then, if we have a sequence of transition, we use TRANS to introduce the next step of the sequence, the right premise being discharged by the previous proof and the left premise is discharged by interacting with the next transition, until we obtain the last transition and use DEQ.

$\Leftarrow$

We have two Representative C_1 and C_2 and a proof tree of the statement. We want to prove that the probability is 0, so the proof We consider the proof modulo the classes of configuration, i.e we remove all the SUBST and RND operations and we obtain a proof on the classes of games such that the proof only contains SYM,DEQ,TRANS,ABSTRACT and axioms. This reduction is possible thanks to the two lemmas.

If we consider all the leafs of the proof tree, they are either DEQ or axioms. If we do not have axioms in the proof a statement of the form $C_1 \simeq C_2$ with $C_1 \in Conf(w_0, q, w_1), C_2 \in Conf(w_0', q', w_1')$ is computationally false and as the core logic is correct we cannot prove it. So we must use axioms to prove this, and if we can do it with only one axiom, then we have that $(w_0, q, w_1) \rightarrow^\delta (w_0', q', w_1')$.

By induction on the size of the proof, if we must use several axioms, we consider the TRANS rule closest to the root. This TRANS rule must be used to introduce a game which is inside a configuration class or else the proof is impossible, and then by induction on the two premises we obtain two sequences $(w_0, q, w_1) \rightarrow^* (w_0', q', w_1')$ and $(w_0', q', w_1') \rightarrow^* (B, q', B)$, so finally we do have : $(w_0, q, w_1) \rightarrow^* (B, q', B)$

□