Computationally Sound Analysis of Protocols using Bilinear Pairings *

Steve Kremer[1] Laurent Mazaré[2]

[1]LSV, ENS Cachan & CNRS & INRIA kremer@lsv.ens-cachan.fr
[2]LexiFi SAS, laurent.mazare@polytechnique.org

Abstract

In this paper, we introduce a symbolic model to analyse protocols that use a bilinear pairing between two cyclic groups. This model consists in an extension of the Abadi-Rogaway logic and we prove that the logic is still computationally sound: symbolic indistinguishability implies computational indistinguishability provided that the Bilinear Decisional Diffie-Hellman assumption holds and that the encryption scheme is IND-CPA secure. We illustrate our results on classical protocols using bilinear pairing like Joux tripartite Diffie-Hellman protocol or the TAK-2 and TAK-3 protocols. We also investigate the security of a newly designed variant of the Burmester-Desmedt protocol using bilinear pairings. More precisely, we show for each of these protocols that the generated key is indistinguishable from a random element.

Keywords: Security, Formal Methods, Dolev-Yao Model, Computational Soundness, Bilinear Pairing

1 Introduction

Recently bilinear pairings such as Weil pairing or Tate pairing on elliptic and hyperelliptic curves have been used to build several cryptographic protocols. One of the first practical pairing-based protocols has been designed by Joux in [29] where a key exchange protocol based on pairing is proposed. This protocol allows three participants to build a shared secret key in a single round. However this protocol was only designed to be secure in the passive setting and is subject to man-in-the-middle attacks. Several key exchange protocols that extend this original protocol were developed, either to ensure some form of authentication [6] or to extend it to a group setting [10]. Pairings were also used

*Work partly supported by the ARA SSIA Formacrypt, the ARA SeSur AVOTÉ and the ARTIST2 Network of Excellence.

as a robust building block for other cryptographic primitives such as identity based encryption schemes or signature schemes [21].

Our contributions. In this paper, we propose an extension of the symbolic model from Dolev and Yao [20] for protocols using bilinear pairing and symmetric encryption. To the best of our knowledge, this is the first time pairings are considered in a Dolev-Yao like model. Moreover we prove that our symbolic model is sound in the computational setting: if there are no attacks in the symbolic setting, then attacks in the computational setting have only a negligible probability of success. This is done by extending the Abadi-Rogaway logic from [3] to symbolic terms using pairings. In particular, we need to adapt formal indistinguishability and keep track of linear relations between polynomials which an adversary might use to distinguish terms. The notion of key cycles needs also to be extended in a non-trivial way, again keeping track of linear relations. We use classical cryptographic assumptions from the standard model to prove soundness: the symmetric encryption scheme has to satisfy indistinguishability against chosen-plaintext attacks ($\mathsf{IND\text{-}CPA}$) and the bilinear mapping has to satisfy the bilinear decisional Diffie-Hellman assumption (BDDH). The proof also relies on a technical result of independent interest, which states that BDDH implies an extended version of BDDH similar to recent results on DDH [13]. Under these assumptions, our soundness result can be used to prove computational security of protocols such as Joux tripartite Diffie-Hellman protocol [29] or the TAK-2 and TAK-3 protocols from Al-Riyami and Paterson [6]. By computational security we mean that the generated key is indistinguishable from a random element. To illustrate the scope of our result we also design a new pairing based variant of the Burmester-Desmedt [14] protocol and prove its security in the passive setting.

We stick to the passive setting of [3]. This setting is restrictive compared to results for active adversaries. However this restriction can be partially removed. As shown by Katz and Yung [30], it is possible to automatically transform a (key agreement) protocol that is secure in the passive setting into a protocol that is secure in the active setting. Hence a protocol that is provably secure against active adversaries can be designed using the following methodology: *(i)* design a protocol and prove that it is secure against a passive, symbolic adversary; *(ii)* use the soundness result of this paper to conclude that this protocol is secure against passive adversaries in the computational setting; *(iii)* apply the Katz and Yung compiler to generate a protocol that is secure against active adversaries in the computational setting.

Related work. This result follows the line of a recent trend in bridging the gap which separates the symbolic and computational views of cryptography. This work started with [3, 2] where only passive adversaries are considered.

Further work focused on extending this result by considering the active setting and by adding cryptographic primitives. The active setting has been explored through a rich and generic framework by Backes et al. in [7] and subsequent papers. Micciancio and Warinschi later proposed another soundness result for the active case in [34]. They consider a less general framework but in their model automatic verification of protocols in the symbolic model is possible through existing tools. This model was later extended in [18, 28] in order to remove some of the original limitations and to consider digital signatures. The work of Canetti and Herzog [15] shows that symbolic proofs obtained by the tool ProVerif imply universal composable security for a restricted class of key exchange protocols.

In the passive setting, numerous cryptographic primitives have been studied. Baudet et al. [11] consider exclusive or and ciphers. Low entropy passwords which are subject to guessing attacks are studied in [1]. Garcia and van Rossum [22] prove soundness of symbolic hashes by using probabilistic hash functions. In [4] a stronger variant of semantic security is used to allow symmetric encryption schemes in the presence of key cycles. Adão et al. [5] allow symmetric encryption which leaks partial information about the length and the key. Laud and Corin [31] did consider composed keys. There have also been results on completeness of symbolic models [33, 26, 5, 11]. However we are not aware of any computational soundness result involving pairing-based protocols.

Variants of the classical Diffie-Hellman assumption are used to characterize the security of bilinear pairings [29]. Hence the concept and difficulties of considering pairings are close to those introduced by considering Diffie-Hellman exponentiation. But computational soundness for this primitive has only been considered in a few works. In [24, 19, 35, 36], results for protocols based on Diffie-Hellman exponentiation are given for the computational protocol composition logic. Herzog presents in [25] an abstract model for Diffie-Hellman key exchange protocols; however in this work the abstract model is very different from classical Dolev-Yao models for modular exponentiation [16] as the adversary is extended with the capability of applying arbitrary polynomial time functions. Bana et al. discuss some of the difficulties to obtain computational soundness for Diffie-Hellman exponentiation in [8]. More recently, Bresson et al. [13] extended the computational soundness result of Abadi and Rogaway [3] to Diffie-Hellman exponentiation. This result relies on a powerful generalization of the Decisional Diffie-Hellman (DDH) assumption and its equivalence with the original DDH assumption. However pairings are not considered in their work, neither in the computational soundness result, nor in the generalization of DDH.

Outline of this paper. The next section recalls the necessary definition for bilinear pairings and introduces BDDH security. Section 3 details our symbolic model: terms, deducibility and equivalence are defined in this setting. In section 4 we present our computational setting by giving concrete semantics to symbolic terms. Our main soundness result is given in section 5: symbolic indistinguishability implies computational indistinguishability for secure cryptographic primitives. Section 6 illustrates this soundness result on some simple protocols using bilinear pairings. Finally a short conclusion is drawn in section 7.

2 Preliminaries on Bilinear Pairings

In this section, we briefly recall the basics of bilinear pairings. The formal definition is given in section 4. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of same prime order q. Let g_1 be a generator of $\mathbb{G}_1$. We use multiplicative notations for both groups. A mapping e from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$ is called a *cryptographic bilinear map* if it satisfies the three following properties.

- **Bilinearity**: $e(g_1^x, g_1^y) = e(g_1, g_1)^{xy}$ for any x, y in $\mathbb{Z}_q$.
- **Non-degeneracy**: $e(g_1, g_1)$ is a generator of $\mathbb{G}_2$ which is also denoted by g_2, *i.e.*, $g_2 \neq 1_{\mathbb{G}_2}$.
- **Computable**: there exists an efficient algorithm to compute $e(u, v)$ for any u and v in $\mathbb{G}_1$.

Examples of cryptographic bilinear maps include modified Weil pairing [12] and Tate pairing [9]: $\mathbb{G}_1$ is a group of points on an elliptic curve and $\mathbb{G}_2$ is a multiplicative subgroup of a finite field. The traditional notation for group $\mathbb{G}_1$ originates from elliptic curve groups and thus is additive. However we prefer a multiplicative notation in order to simplify our symbolic model of section 3.

The classical decisional security assumption for groups with pairing is the Bilinear Decisional Diffie-Hellman (BDDH) assumption. This assumption states that it is difficult for an adversary that has access to three elements of $\mathbb{G}_1$, g_1^x, g_1^y and g_1^z to distinguish g_2^{xyz} from a randomly sampled element g_2^r of $\mathbb{G}_2$.

A simple key exchange protocol has been proposed by Joux in [29]. This protocol is an extension of the classical Diffie-Hellman key exchange for three participants. Let A, B and C be the three participants. Each of them randomly samples a value in $\mathbb{Z}_q$ (denoted by x for A, by y for B and by z for C). Then the three following messages are exchanged:

$$(1)\ A \rightarrow B, C : g_1^x \qquad (2)\ B \rightarrow A, C : g_1^y \qquad (3)\ C \rightarrow A, B : g_1^z$$

The shared secret key is g_2^{xyz}. It is easy to check that A, B and C can compute this key by using the bilinear map e on the two messages that they have received. Security of this protocol with respect to key indistinguishability in the passive setting is identical to the BDDH assumption [29]. No form of authentication is provided in this protocol, so it is trivially subject to man-in-the-middle attacks.

In the following sections, our objective is to provide a symbolic model for protocols using bilinear maps and to give a computational justification of this model. We stick to the passive setting but as noted earlier this is not a real restriction thanks to the Katz and Yung compiler [30]. As usual the computational setting is parameterized by a security parameter η which can be thought of as the key length. Adversaries are probabilistic polynomial-time (in η) Turing machines. In this paper, we suppose that the adversaries are given implicit access to as many fresh random coins as needed, as well as to the complexity parameter η.

3 The Symbolic Setting

In this section, we introduce the symbolic view of cryptography: messages are represented as algebraic terms, the adversary's capabilities are defined by an entailment relation $\vdash$ and an observational equivalence $\cong$. This equivalence is an extension of the well-known Abadi-Rogaway logic to terms using symmetric encryption and pairing. The main difference with the original logic is that we introduce generator g_1 for the first group ($\mathbb{G}_1$), and generator g_2 for the second group ($\mathbb{G}_2$) as well as an infinite set of names representing exponents.

3.1 Terms and Deducibility

Let **Keys** and **Exponents** be two countable disjoint sets of symbols for keys and exponents. A power-free 3-monomial is a product of three *distinct* exponents and a power-free 3-polynomial is a linear combination of monomials using coefficients in $\mathbb{Z}$ (with no constant coefficient). Hence let x_1, x_2, x_3, x_4 and x_5 be five distinct elements of **Exponents**, $2x_1x_2x_3 + x_3x_4x_5$ is a power-free 3-polynomial but $x_1^2x_2$ and $x_1x_2x_3+1$ are not. We let **Poly** be the set of power-free 3-polynomials with variables in **Exponents** and coefficients in $\mathbb{Z}$. With a slight abuse of notation, we often refer to power-free 3-monomials as monomials and to power-free 3-polynomials as polynomials. Our symbolic setting is restricted to 3-monomials because this is the classical way to use bilinear pairing; using pairings with monomials of order different than 3 might be unsafe.

Let k, x and p be meta-variables over **Keys**, **Exponents** and **Poly** respectively. Polynomials can be used as exponents and the set **T** of terms is built using symbolic encryption and concatenation of keys, exponents and exponentiations:

$$\begin{array}{rcl} msg & ::= & (msg, msg) \mid \{msg\}_{key} \mid x \mid key \mid g_1^x \\ key & ::= & k \mid g_2^p \end{array}$$

Term (t_1, t_2) represents the pair composed of terms t_1 and t_2, $\{t\}_k$ represents (symmetric) encryption of term t using key k. In the remainder of the paper we will sometimes use tuples instead of nested pairs in order to simplify the notation. $\{t\}_{g_2^p}$ represents encryption of term t using a key derived from g_2^p (in the computational semantics we assume the implicit application of a deterministic key extraction algorithm Kex which is detailed below). g_1^x and g_2^p represent modular exponentiation of g_1 (generator of the first group) and g_2 (generator of the second group) to the power of an exponent x in the first case and a polynomial p in the second case.

We use classical notations for manipulating terms. A position is a finite word over the natural numbers, ϵ denotes the empty word and $w_1 \cdot w_2$ is the concatenation of w_1 and w_2. The set of positions $pos(t)$ of a term t is inductively defined as $pos(x) = pos(k) = pos(g_i) = \{\epsilon\}$ and $pos(f(t_1, t_2)) = \{\epsilon\} \cup \bigcup_{i \in \{1,2\}} i \cdot pos(t_i)$ where f represents either pairing, encryption or exponentiation. If p is a position of t then the expression $t|_p$ denotes the subterm of t at the position p, *i.e.*, $t|_\epsilon = t$ and $f(t_1, t_2)|_{i \cdot p} = t_i|_p$.

Example 3.1 *Let $t = (k, \{k'\}_k)$. The set of positions $pos(t)$ of t is $\{\epsilon, 1, 2, 2 \cdot 1, 2 \cdot 2\}$. Moreover, $t|_1 = t|_{2\cdot 2} = k$ and $t|_{2\cdot 1} = k'$.*

We say that g_2^p *occurs at a key position* in term t if $\{t'\}_{g_2^p}$ is a subterm of t for some t'. Otherwise we say that g_2^p *occurs as data.* Note that in a same term g_2^p may occur both at a key position and as data. An exponent x can be used as an exponent of either g_1 (*e.g.*, in term g_1^x) or g_2 (*e.g.*, in term $g_2^{xx_2x_3}$). Otherwise, if x is not used used as an exponent of either g_1 or g_2, we say that x is used as data.

For any term t, $\mathsf{pol}\,(t)$ designates the set of polynomials p such that g_2^p is a subterm of t and $\mathsf{mon}\,(t)$ designates the set of monomials used by polynomials in $\mathsf{pol}\,(t)$.

Example 3.2 *Let $t = (\{k\}_{g_2^{2x_1x_2x_3+x_4x_5x_6}}, g_1^{x_1}, g_2^{x_1x_2x_3})$. Then $\mathsf{pol}\,(t) = \{2x_1x_2x_3 + x_4x_5x_6, x_1x_2x_3\}$ and $\mathsf{mon}\,(t) = \{x_1x_2x_3, x_4x_5x_6\}$.*

Equality between polynomials is considered modulo the classical equational theory: associativity and commutativity for addition and multiplication, distributivity of multiplication over addition. Equality can easily be decided, for instance by rewriting polynomials in some normal form $\sum_{i=1}^{n} \lambda_i x_1^{p_{i,1}} \ldots x_k^{p_{i,k}}$ and comparing these normal forms.

First we define a deduction relation $E \vdash t$ where E is a finite set of terms and t is a term. The intuitive meaning of $E \vdash t$ is that t can be deduced from E. The deducibility relation is an extension of the classical Dolev-Yao inference system [20]:

$$\frac{t \in E}{E \vdash t} \quad \frac{E \vdash (t_1, t_2)}{E \vdash t_1} \quad \frac{E \vdash (t_1, t_2)}{E \vdash t_2} \quad \frac{E \vdash \{t\}_{key} \quad E \vdash key}{E \vdash t}$$

Note that we did not consider *composition rules* such as if t_1 and t_2 are deducible then (t_1, t_2) is also deducible. Indeed these rules are not necessary as deduction is only used to check whether some key can be deduced from a term. As keys are atomic, it is sufficient to consider the four previous rules. By atomic we mean that keys do not include pairs or encryptions but they may obviously be of the form g_2^p. We add four new deduction rules in order to handle pairing. The three first rules correspond to the three possible ways to obtain an exponentiation g_2^{xyz} using the cryptographic bilinear map:

$$\frac{E \vdash x \quad E \vdash g_1^y \quad E \vdash g_1^z}{E \vdash g_2^{xyz}} \quad \frac{E \vdash x \quad E \vdash y \quad E \vdash g_1^z}{E \vdash g_2^{xyz}} \quad \frac{E \vdash x \quad E \vdash y \quad E \vdash z}{E \vdash g_2^{xyz}}$$

Note that these three rules correspond to "real" capacities of the adversary in the computational setting. In the first case, an adversary knowing g_1^y and g_1^z can use the bilinear map to produce g_2^{yz}. As he also knows x he can exponentiate g_2^{yz} to obtain g_2^{xyz}. In the second case, the adversary knows y so he can produce g_1^y and act as in the first case. Finally, the third case is also similar, as the adversary can compute g_1^z and act as in the second case.

The fourth rule handles *linear relations* between polynomials.

$$\frac{E \vdash g_2^p \quad E \vdash g_2^q}{E \vdash g_2^{\lambda p+q}} \lambda \in \mathbb{Z}$$

In the computational world an adversary can multiply two group elements g_2^p and g_2^q in order to get g_2^{p+q}. He can also exponentiate a group element g_2^p and obtain $g_2^{\lambda p}$. Thus it is feasible for the adversary to produce $g_2^{\lambda p+q}$ from g_2^p and g_2^q.

Given this deduction relation we can define the set of *deducible keys of term* t as

$$K(t) = \{k \mid t \vdash k\} \cup \{g_2^p \mid t \vdash g_2^p \wedge g_2^p \text{ is a subterm of } t\}$$

After adding the new deductions, the deducibility relation is still decidable.

Proposition 3.3 *Let t be a term and E be a finite set of terms. Then deducibility of t from E is decidable.*

Proof. In this proof, we use the notion of reachability. First remember that a *key term* is either an element k of **Keys** or an exponentiation g_2^q where q is an element of **Poly**.

A subterm t' of t is *reachable* from t using a set K of *key* terms, iff there exists a position p in t such that $t|_p = t'$ and for any prefix p' of p, *i.e.*, $p = p' \cdot p''$, if $t|_{p'}$ is an encryption $\{u\}_{key}$ then $key \in \mathsf{K}$.

We first show that the set $K(t)$ of deducible keys is computable. Note that $K(t)$ is bounded (for inclusion) by the set of keys and exponentiations of t. The set $K(t)$ can be iteratively computed as follows.

1. Initially, K is empty.

   Iterate the following steps until reaching a fix-point:

2. At each step, any key k and exponentiations g_2^p that is reachable in t using keys and exponentiations from K is added to K.

3. We build the set of reachable monomials rm which contains all the monomials $x_1x_2x_3$ from t such that either

   - x_1, x_2 and x_3,
   - or x_1, x_2 and $g_1^{x_3}$,
   - or x_1, $g_1^{x_2}$ and $g_1^{x_3}$

   are reachable in t using K.

4. At the end of each step, if p is a polynomial from $\mathsf{pol}\,(t)$ which is a linear combination of polynomials from K and monomials from rm, then g_2^p is also added to K.

Now let t be a term and E a finite set of terms t_1 to t_n.

1. If t is an atomic key k, then t is deducible if and only if k appears in $K((t_1, \dots, t_n))$. Thus deducibility is decidable.
2. Else if t is a key g_2^p, then t is deducible if and only if $E, \{k\}_{g_2^p} \vdash k$ where k is a fresh atomic key (*i.e.*, k does not appear in E). Thus deducibility can be decided as in the previous case.
3. Otherwise t is either an exponent, or a pair, or an encryption, or an exponentiation of g_1. As we do not have any composition rule in the definition of $\vdash$, t is deducible if and only if t appears as a subterm in one of the t_j and is reachable using $K((t_1, \dots, t_n))$. Hence a decision algorithm can first build $K = K((t_1, \dots, t_n))$ then check for reachability of t in any of the t_j using K.

$\square$

Alternative definitions are possible for the deduction system. For example, we could consider adding the deduction rule $\frac{E \vdash x}{E \vdash g_1^x}$. Then rules $\frac{E \vdash x \quad E \vdash y \quad E \vdash g_1^z}{E \vdash g_2^{xyz}}$ and $\frac{E \vdash x \quad E \vdash y \quad E \vdash z}{E \vdash g_2^{xyz}}$ would not be necessary anymore and the computational soundness results presented later in this document would still be true. However we stick to our deduction system as it reflects in a simple way how a key g_2^p can be deduced from other terms.

Note that we have only shown decidability of the deduction relation. As, in contrast to a computational adversary, a symbolic adversary is not resource-bounded (in particular it is not polynomial-time bounded) we do not need to detail the complexity for our soundness result. From a verification perspective, efficient algorithms are of course needed which would require a more fine-grained complexity analysis of the above procedure.

3.2 Equivalence

Patterns. Patterns are used to characterize the information that can be extracted from a term. These patterns are close to those introduced in [3, 32] but are extended in order to handle modular exponentiation. We introduce a new symbol $\square$ representing a ciphertext that the adversary cannot decrypt. Moreover we consider that the encryption scheme is not necessarily key-concealing. Hence it may be possible for an adversary to observe whether two ciphertexts have been produced using the same key.

Let t be a term and K be a finite set of keys and elements of the second group g_2^p, then the pattern of t using K, $\mathsf{pat}(t, \mathsf{K})$ is inductively defined by:

$$\begin{array}{lcll}
\mathsf{pat}((t_1, t_2), \mathsf{K}) & = & \big(\mathsf{pat}(t_1, K), \mathsf{pat}(t_2, \mathsf{K})\big) & \\
\mathsf{pat}(\{t'\}_{key}, \mathsf{K}) & = & \{\mathsf{pat}(t', \mathsf{K})\}_{key} & \text{if } key \in \mathsf{K} \\
\mathsf{pat}(\{t'\}_{key}, \mathsf{K}) & = & \{\square\}_{key} & \text{if } key \notin \mathsf{K} \\
\mathsf{pat}(a, \mathsf{K}) & = & a & \text{for } a \text{ in } x,\, k,\, g_1^x \text{ and } g_2^p
\end{array}$$

The set K is used to store keys that are known by the adversary.

We say that two terms t_1 and t_2 are *equivalent*, $t_1 \equiv t_2$, if they have the same pattern: $t_1 \equiv t_2$ if and only if $\mathsf{pat}\,(t_1, K(t_1)) = \mathsf{pat}\,(t_2, K(t_2))$. Intuitively patterns hide information that are encrypted with undeducible keys. Hence two terms have the same pattern if the information that can be extracted is the same, so it is impossible to distinguish these two terms.

Equivalence up to renaming. We allow (bijective) renaming of keys in a similar way as [3] but renaming of polynomials is slightly more complex and relies on a linear relation preserving bijection between polynomials. Let us illustrate this on the two following examples.

- Let t_1 be the term $(x_1, x_2, g_1^{x_3}, g_2^{x_4x_5x_6}, g_2^{x_1x_2x_3+x_4x_5x_6})$ and t_2 be the term $(x_1, x_2, g_1^{x_3}, g_2^{x_4x_5x_6}, g_2^{x_7x_8x_9})$. A bijection from polynomials of t_2 to polynomials of t_1 could be

  $$\{x_7x_8x_9 \mapsto x_1x_2x_3 + x_4x_5x_6\ ;\ x_4x_5x_6 \mapsto x_4x_5x_6\}$$

  However this bijection does not correctly preserve linear relations. In term t_1, $g_2^{x_1x_2x_3+x_4x_5x_6}$ can be obtained by multiplying $g_2^{x_4x_5x_6}$ with $g_2^{x_1x_2x_3}$ (which is obtained by applying the bilinear map to $g_1^{x_2}$ and $g_1^{x_3}$ and raising the result to the power x_1). In term t_2, $g_2^{x_7x_8x_9}$ cannot be obtained in a similar way.

- Let t_1 be the term $(g_2^{x_4x_5x_6}, g_2^{x_1x_2x_3+x_4x_5x_6})$ and t_2 be the term $(g_2^{x_4x_5x_6}, g_2^{x_7x_8x_9})$. The associated bijection is

  $$\{x_7x_8x_9 \mapsto x_1x_2x_3 + x_4x_5x_6\ ;\ x_4x_5x_6 \mapsto x_4x_5x_6\}$$

  This bijection correctly preserves linear relations as $g_2^{x_1x_2x_3+x_4x_5x_6}$ cannot be obtained from other parts of t_1 ($x_1x_2x_3 + x_4x_5x_6$ is not involved in any linear relations) and $g_2^{x_7x_8x_9}$ cannot be obtained from other parts of t_2.

In order to properly define what is a linear relation preserving bijection, we first introduce the set $dm(t)$ of *deducible monomials* from t, *i.e.*, monomials that can be obtained using the bilinear map operation (this is a slight abuse of notation as a monomial m may not be deducible itself while its exponentiation g_2^m is deducible). A monomial $x_1x_2x_3$ from $\mathsf{mon}\,(t)$ is in $dm(t)$ if one or more of the following conditions hold:

- x_1, x_2 and x_3 are deducible from t,
- x_1, x_2 and $g_1^{x_3}$ are deducible from t,
- x_1, $g_1^{x_2}$ and $g_1^{x_3}$ are deducible from t.

We can now formalize the definition. Let t_2 and t_1 be two terms. A bijection σ from $\mathsf{pol}\,(t_2)$ to $\mathsf{pol}\,(t_1)$ is *linear relation preserving* for t_2 and t_1 if the same linear relations are verified between polynomials from t_2 and their image using σ. However monomials from $dm(t_2)$ cannot be renamed as they are linked to other parts of term t_2 due to the bilinear pairing. Formally, σ has to verify the following condition:

$$\forall p_1, ..., p_n \in \mathsf{pol}\,(t_2),\ \forall a_1, ..., a_n \in \mathbb{Z}, \quad \forall m_1, ..., m_{n'} \in dm(t_2),\ \forall b_1, ..., b_{n'} \in \mathbb{Z},$$
$$\sum_{i=1}^{n} a_i p_i = \sum_{j=1}^{n'} b_j m_j \Leftrightarrow \sum_{i=1}^{n} a_i (p_i \sigma) = \sum_{j=1}^{n'} b_j m_j$$

Reconsider our first example: t_1 is the term $(x_1, x_2, g_1^{x_3}, g_2^{x_4x_5x_6}, g_2^{x_1x_2x_3+x_4x_5x_6})$ and t_2 is the term $(x_1, x_2, g_1^{x_3}, g_2^{x_4x_5x_6}, g_2^{x_7x_8x_9})$. We define the bijection $\sigma = \{x_7x_8x_9 \mapsto x_1x_2x_3+x_4x_5x_6\}$. We have that σ is *not* a linear relation preserving bijection for t_2 and t_1 because $x_1x_2x_3$ is in $dm(t_2)$ and

$$\begin{array}{ll} & (x_7x_8x_9) + (-1)(x_4x_5x_6) \neq x_1x_2x_3 \\ \text{but} & (x_7x_8x_9)\sigma + (-1)(x_4x_5x_6)\sigma = (x_1x_2x_3 + x_4x_5x_6) - (x_4x_5x_6) = x_1x_2x_3 \end{array}$$

Definition 3.4 *Two terms t_1 and t_2 are* equivalent up to renaming, *$t_1 \cong t_2$ if they are equivalent up to some renaming of keys of polynomials.*

$$\begin{array}{lll} t_1 \cong t_2 & \textit{iff} & \exists \sigma_1 \textit{ a renaming of } \textbf{Keys} \\ & & \exists \sigma_2 \textit{ a bijection preserving linear relations from } \mathsf{pol}\,(t_2) \textit{ to } \mathsf{pol}\,(t_1) \\ & & \textit{such that } t_1 \equiv t_2\sigma_1\sigma_2 \end{array}$$

In this definition of equivalence, we have not considered renaming of **Exponents** to preserve simplicity but this can easily be added. Using this new definition, an interesting result is the decidability of equivalence up to renaming.

Proposition 3.5 *Let t_1 and t_2 be two terms. Equivalence up to renaming of t_1 and t_2 is decidable.*

Proof. As detailed in the proof of proposition 3.3, there exists an algorithm that takes as input a term t and outputs the finite set $K(t)$. This allows us to build an algorithm that takes as input a term t and outputs $\mathsf{pat}\,(t, K(t))$.

Let t_1 and t_2 be two terms. Then it is possible to compute $\mathsf{pat}\,(t_1, K(t_1))$ and $\mathsf{pat}\,(t_2, K(t_2))$ (and so equivalence *without* renaming, $\equiv$, is decidable).

In order to decide equivalence up to renaming of terms t_1 and t_2, we apply a unification algorithm recursively on $\mathsf{pat}\,(t_1, K(t_1))$ and $\mathsf{pat}\,(t_2, K(t_2))$ resulting in a renaming σ_1 and a bijection σ_2 from $\mathsf{pol}\,(t_2)$ to $\mathsf{pol}\,(t_1)$. This unification algorithm takes two terms u_1 and u_2 as an input and works as follows:

1. If u_1 is a pair (v_1, w_1) and u_2 is a pair (v_2, w_2) the algorithm is applied recursively on v_1 and v_2 resulting in σ_1 and σ_2. This algorithm is also applied recursively on w_1 and w_2 resulting in σ'_1 and σ'_2. If σ_1 and σ'_1 are compatible (*i.e.*, for any atomic key k that is in the domain of both σ_1 and σ'_1, $k\sigma_1 = k\sigma'_1$) and σ_2 and σ'_2 are also compatible, then u_1 and u_2 can be unified resulting in $\sigma_1 \cup \sigma'_1$ and $\sigma_2 \cup \sigma'_2$. Otherwise u_1 and u_2 cannot be unified and t_1 and t_2 are not equivalent up to renaming.

2. If u_1 is an encryption $\{v_1\}_{key_1}$ and u_2 is an encryption $\{v_2\}_{key_2}$ we proceed as for pairs in the previous point: v_1 and v_2 are unified, key_1 and key_2 are unified and the compatibility is checked.

3. If u_1 is an atomic key k_1 and u_2 is an atomic key k_2. Then $\sigma_1 = \{k_2 \mapsto k_1\}$ and $\sigma_2 = \emptyset$.

4. If u_1 is a key $g_2^{p_1}$ and u_2 is a key $g_2^{p_2}$ then $\sigma_1 = \emptyset$ and $\sigma_2 = \{p_2 \mapsto p_1\}$.

5. If u_1 is an exponentiation $g_1^{x_1}$ and u_2 is an exponentiation $g_1^{x_2}$ or if u_1 is an exponent x_1 and u_2 is an exponent x_2 and x_1 is equal to x_2, then u_1 and u_2 can be unified resulting in $\sigma_1 = \sigma_2 = \emptyset$. Otherwise t_1 and t_2 are not equivalent up to renaming.

6. Otherwise, u_1 and u_2 cannot be unified and terms t_1 and t_2 are not equivalent up to renaming.

Now, it only remains to check that σ_2 is a linear relation preserving bijection for t_2 and t_1. First the set $dm(t_2)$ is computed. Notice that elements of $dm(t_2)$ are monomials using exponents from t_2. For each possible monomial m, m is in $dm(t_2)$ if and only if $m = x_1x_2x_3$ and one of the three following holds:

- x_1, x_2 and x_3 are reachable in t_2 using $K(t_2)$.
- x_1, x_2 and $g_1^{x_3}$ are reachable in t_2 using $K(t_2)$.
- x_1, $g_1^{x_2}$ and $g_1^{x_3}$ are reachable in t_2 using $K(t_2)$.

In order to check that σ_2 preserves linear relations of t_2 we need to check that σ_2 does neither remove nor add any linear relation. To check whether σ_2 removes a linear relation in t_2 we use the following algorithm. Let P be an initially empty set of polynomials. The algorithm iterates on polynomials from $\mathsf{pol}\,(t_2)$. For each such polynomial p, the algorithm tests whether p is involved in a linear relation with polynomials from P and monomials from $dm(t_2)$. This can be tested by checking whether the system of linear equations $\sum_{1\leq i\leq n} \lambda_i p_i + \sum_{1\leq i\leq j} \lambda'_j m_j - p = 0$ with $P = \{p_1, \ldots, p_n\}$ and $dm(t_2) = \{m_1, \ldots, m_j\}$ has a solution, e.g. using Gauss elimination. If this is the case, then if $p\sigma_2$ verifies the same relation with $P\sigma_2$ and $dm(t_2)$, the algorithm continues, else if the relation is not satisfied by $p\sigma_2$, $P\sigma_2$ and $dm(t_2)$, then σ_2 is not linear relation preserving. If p is not involved in a linear combination with polynomials from P, then p is added to P. After that, the loop continues. As $\mathsf{pol}\,(t_2)$ is finite, this algorithm always terminates. To check whether σ_2 adds a linear relation to t_2, we use the previous algorithm and (equivalently) check whether σ_2^{-1} removes a linear relation in t_1. $\square$

3.3 Examples

Here we give some examples that illustrate the choices we made when defining the equivalence. These choices are motivated by the possibilities of adversaries in the computational setting. Unlike [3], our symbolic model does not include symbolic constants like 0 or 1 as data. However these constants can be easily encoded using for instance two key names k_0 and k_1 which are explicitly revealed. Then 1 denotes k_1 and 0 denotes k_0. Instead of verifying the equivalence between t and t', we check whether (k_0, k_1, t) and (k_0, k_1, t') are equivalent.

1. $\{0\}_k \cong \{1\}_k$. This example shows that symmetric encryption perfectly hides its plaintext.

2. $(\{0\}_k, \{0\}_k) \cong (\{0\}_k, \{1\}_k)$. Symmetric encryption also hides equalities among the underlying plaintexts. To achieve this, encryption has to be probabilistic. As modular exponentiation is deterministic, we cannot ask modular exponentiation to hide such relations.

3. $(g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_2^{x_1x_2x_3}) \cong (g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_2^{x'_1x'_2x'_3})$. This example illustrates security of Joux's protocol [29] against passive adversaries. The adversary observes the unfolding of the protocol where three exponentiations are exchanged. These exponentiations allows the three participants to build a shared secret key $g_2^{x_1x_2x_3}$. Then the adversary cannot distinguish the shared key from a randomly sampled element of the second group $g_2^{x'_1x'_2x'_3}$ (as the order of the group is prime, $g_2^{x'_1x'_2x'_3}$ has a uniform distribution over elements of the second group).

   Moreover the symbolic setting can be used to verify that each participant is able to compute the shared key. For example the first participant generates exponent x_1 and receives $g_1^{x_2}$ and $g_1^{x_3}$ from the second and third participants. Using this knowledge, he is able to compute the shared secret key as $x_1, g_1^{x_2}, g_1^{x_3} \vdash g_2^{x_1x_2x_3}$.

4. $(g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, \{0\}_{g_2^{x_1x_2x_3}}) \cong (g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, \{1\}_{g_2^{x_1x_2x_3}})$. This example combines the Joux protocol with an exchange of secret information using the shared key. Thus in this example symmetric encryption and bilinear pairing are used simultaneously.

5. $(g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, x_4, x_5, x_6, g_2^{x_1x_2x_3+x_4x_5x_6}) \cong (g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, x_4, x_5, x_6, g_2^{x'_1x'_2x'_3})$. Let t_2 be the second term in the equivalence relation. This example illustrates a more complex renaming. The adversary has access to some exponents from the key $g_2^{x_1x_2x_3+x_4x_5x_6}$ but is still unable to distinguish it from a randomly sampled key. $x_4x_5x_6$ can be seen as a vulnerable part of the key but $x_1x_2x_3$ makes the whole key secure. The two terms are equivalent up to renaming because bijection $\{x'_1x'_2x'_3 \mapsto x_1x_2x_3 + x_4x_5x_6\}$ is linear relation preserving; indeed $x'_1x'_2x'_3$ and $x_1x_2x_3 + x_4x_5x_6$ are both not involved in any linear relation with monomials from $dm(t_2)$.

6. In the following example, there are two shared keys.

$$\begin{array}{rl} & (g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, x_4, x_5, x_6, g_2^{x_1x_2x_3+x_4x_5x_6}, g_2^{x_1x_2x_3}) \\ \not\cong & (g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, x_4, x_5, x_6, g_2^{x'_1x'_2x'_3}, g_2^{x'_4x'_5x'_6}) \end{array}$$

Let t_1 and t_2 be the first and second term in this (non-)equivalence relation. The bijection $\sigma = \{x_1'x_2'x_3' \mapsto x_1x_2x_3 + x_4x_5x_6 \ ; \ x_4'x_5'x_6' \mapsto x_1x_2x_3\}$ is not linear relation preserving. Indeed, the monomial $x_4x_5x_6$ is in $dm(t_2)$ and there is a relation among polynomials used in the two keys of t_1 and $x_4x_5x_6$ which is not true in t_2:

$$\begin{array}{rcl}(x_1'x_2'x_3')\sigma + (-1)(x_4'x_5'x_6')\sigma & = & x_4x_5x_6 \\ (x_1'x_2'x_3') + (-1)(x_4'x_5'x_6') & \neq & x_4x_5x_6\end{array}$$

4 The Computational Setting

In this section, we formalize the mapping between symbolic terms and distributions of bit-strings. This mapping depends on the algorithms used to implement the two cryptographic primitives used in the symbolic setting: symmetric encryption and pairing.

4.1 Encryption Scheme

We recall the standard definition for symmetric encryption schemes. A symmetric encryption scheme $\mathcal{SE}$ is defined by three algorithms $\mathcal{KG}$, $\mathcal{E}$ and $\mathcal{D}$. The key generation algorithm $\mathcal{KG}$ takes as input the security parameter η and outputs a key k. The encryption algorithm $\mathcal{E}$ is randomized. It takes as input a bit-string s and a key k and returns the encryption of s using k. The decryption algorithm $\mathcal{D}$ takes as input a bit-string c representing a ciphertext and a key k and outputs the corresponding plaintext. Given $k \leftarrow \mathcal{KG}(\eta)$, we have that for any bit-string s, if $c \leftarrow \mathcal{E}(s,k)$ then it is required that $\mathcal{D}(c) = s$.

In order to characterize security of a symmetric encryption scheme, we use the classical IND-CPA (indistinguishability against chosen plaintext attacks) notion [23].

IND-CPA security. In this paper we use schemes that satisfy length-concealing semantic security[1]. The definition that we recall below uses a left-right encryption oracle $LR^b_{\mathcal{SE}}$. This oracle first generates a key k using $\mathcal{KG}$. Then it answers queries of the form (bs_0, bs_1), where bs_0 and bs_1 are bit-strings, an important point is that bs_0 and bs_1 may have different lengths. The oracle returns ciphertext $\mathcal{E}(bs_b, k)$. The goal of the adversary $\mathcal{A}$ is to guess the value of bit b and for that purpose $\mathcal{A}$ has access to oracle $LR^b_{\mathcal{SE}}$. His advantage is defined as the probability that he outputs 1 when using oracle $LR^1_{\mathcal{SE}}$ minus the probability that he outputs 1 when using oracle $LR^0_{\mathcal{SE}}$.

$$\mathrm{Adv}^{\mathsf{CPA}}_{\mathcal{SE},\mathcal{A}}(\eta) = \left|\mathbb{P}\left[\mathcal{A}^{LR^1_{\mathcal{SE}}}(\eta) = 1\right] - \mathbb{P}\left[\mathcal{A}^{LR^0_{\mathcal{SE}}}(\eta) = 1\right]\right|$$

An encryption scheme $\mathcal{SE}$ is said to be IND-CPA secure if the advantage of any polynomial-time adversary $\mathcal{A}$ is negligible in η.

[1]Such schemes can only exist if the maximum length of plaintexts is bounded, however we do not take this into account in this paper.

The difference with the standard notion of semantic security is that an adversary can call oracle $LR^b_{\mathcal{SE}}$ on two bit-strings bs_0 and bs_1 of different lengths. Therefore in order to be secure for our notion, an encryption scheme has to hide the length of the plaintext. By abuse of notation we call the resulting scheme also IND-CPA secure.

4.2 Pairing

A *bilinear pairing instance generator* is defined as a probabilistic polynomial-time algorithm IG which given a security parameter η outputs a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, g_1, e)$ composed of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, a generator g_1 of $\mathbb{G}_1$ and a cryptographic bilinear map e between $\mathbb{G}_1$ and $\mathbb{G}_2$. A generator g_2 of group $\mathbb{G}_2$ is obtained by applying e to (g_1, g_1).

BDDH security. An instance generator IG satisfies the *Bilinear Decisional Diffie-Hellman assumption*, BDDH, iff for any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ against BDDH, $\mathrm{Adv}^{\mathsf{BDDH}}_{\mathcal{A},IG}$, defined below is negligible in η.

$$\begin{aligned}\mathrm{Adv}^{\mathsf{BDDH}}_{\mathcal{A},IG}(\eta) \quad = \quad & \mathbb{P}\left[\begin{array}{c}(q, \mathbb{G}_1, \mathbb{G}_2, g_1, e) \leftarrow IG(\eta)\\ x, y, z \leftarrow \mathbb{Z}_q\end{array}, \ \mathcal{A}(g_1, g_1^x, g_1^y, g_1^z, g_2^{xyz}) = 1\right]\\ & -\mathbb{P}\left[\begin{array}{c}(q, \mathbb{G}_1, \mathbb{G}_2, g_1, e) \leftarrow IG(\eta)\\ x, y, z, r \leftarrow \mathbb{Z}_q\end{array}, \ \mathcal{A}(g_1, g_1^x, g_1^y, g_1^z, g_2^r) = 1\right]\end{aligned}$$

This means that an adversary that is given g_1^x, g_1^y and g_1^z can only make the difference between g_2^{xyz} and a random group element with negligible probability.

4.3 Computational Semantics of Terms

Computational semantics depend on a symmetric encryption scheme $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ and of an instance generator IG. In order to transform elements of the second group into keys usable for $\mathcal{SE}$, we assume the existence of a key extractor [17] algorithm Kex (this can be done for example by extracting randomness using an entropy smoothing hash function [27]). We suppose that the distribution of keys generated by $\mathcal{KG}$ is equal to the distribution obtained by applying Kex to a random element of $\mathbb{G}_2$ (which is the second group generated by IG). We associate to each symbolic term t a distribution of bit-strings $[\![t]\!]_{\mathcal{SE},IG}$ that depends on the security parameter η. This distribution is defined by the following random algorithm:

1. Algorithm IG is used to generate two paired groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q and of generators $\widehat{g_1}$ and $\widehat{g_2}$. For each key k from t, a value $\widehat{k}$ is randomly drawn using $\mathcal{KG}$. For each exponent x, a value $\widehat{x}$ is randomly sampled in $\mathbb{Z}_q$ equipped with the uniform distribution.
2. Then the bit-string evaluation of term t is computed recursively on the structure of t:

- If t is a key k or an exponent x, then the value $\widehat{t}$ is returned.
- If t is an exponentiation g_1^x, then the exponentiation of $\widehat{g_1}$ to the power of $\widehat{x}$ is returned.
- If t is an exponentiation g_2^p, then the algorithm computes the value n of p in $\mathbb{Z}_q$, and the exponentiation of $\widehat{g_2}$ to the power of n is returned.
- If t is a pair (t_1, t_2), the algorithm is applied recursively on t_1 holding bs_1 and on t_2 holding bs_2. The output of the algorithm is the concatenation of bs_1 and bs_2.
- If t is an encryption $\{t'\}_k$, the algorithm is applied recursively on t' holding bs' and on k holding bs_k. The output of the algorithm is $\mathcal{E}(bs', bs_k)$.
- If t is an encryption $\{t'\}_{g_2^p}$, the algorithm is applied recursively on t' holding bs' and on g_2^p holding bs_k. The output of the algorithm is $\mathcal{E}(bs', \mathsf{Kex}(bs_k))$.

5 Soundness of the Symbolic Model

In this section we prove that the extension of the Abadi-Rogaway logic given in section 3 is computationally sound when implemented using an IND-CPA encryption scheme and using an instance generator satisfying BDDH: if two terms are equivalent up to renaming in the symbolic setting, their evaluations (given by the computational semantics of section 4) are computationally indistinguishable.

Well-formed Terms. Our soundness result is only true for terms that make a correct use of the bilinear pairing. Such terms are called *well-formed* terms. Formally a term t is *well-formed* if for any monomial m in $\mathsf{mon}(t)$:

- either for any monomial m' in $\mathsf{mon}(t)$ different from m, m and m' do not have any common exponent;
- or none of the three exponents used by m occurs as data in t.

This technical restriction is necessary to obtain soundness. Indeed let us consider $t_1 = (x, y, g_2^{xz_1z_2}, g_2^{yz_1z_2})$ and $t_2 = (x, y, g_2^{r_1r_2r_3}, g_2^{r_4r_5r_6})$. Note that t_1 is not well-formed as xz_1z_2 and yz_1z_2 have common exponents (z_1 and z_2) and exponents x and y occur as data. Terms t_1 and t_2 are equivalent up to renaming. However it is possible to build an adversary $\mathcal{A}$ that can distinguish the corresponding distributions efficiently (the precise definition of indistinguishability will be given below). Adversary $\mathcal{A}$ takes as input (x, y, U, V) and has to decide whether $U = g_2^{xz_1z_2}$ and $V = g_2^{yz_1z_2}$ or $U = g_2^{r_1r_2r_3}$ and $V = g_2^{r_4r_5r_6}$. $\mathcal{A}$ proceeds as follows:

- compute x^{-1} and y^{-1};

- output 1 if $U^{x^{-1}} = V^{y^{-1}}$;
- output 0 otherwise.

If $\mathcal{A}$ outputs 1 it was indeed given the distribution corresponding to t_1 with probability close to 1 (the probability that $r_1 r_2 r_3 x^{-1} = r_4 r_5 r_6 y^{-1}$ is negligible). Otherwise, if A outputs 0 it must have been given the distribution corresponding to t_2. Hence, $\mathcal{A}$ efficiently distinguishes two equivalent terms. We forbid such use of bilinear pairing by considering only well-formed terms.

Acyclicity Restrictions. The importance of key cycles was already described in [3]. In the setting of [3] a key cycle is a sequence of keys $K_1, \dots, K_n$ such that K_{i+1} encrypts (possibly indirectly) K_i and K_n encrypts K_1. An encryption of key K with itself, *i.e.*, $\mathcal{E}_K(K)$ is a key cycle of length 1. An example of a key cycle of size 2 would be $\mathcal{E}_{K_1}(K_2), \mathcal{E}_{K_2}(K_1)$. In general $\mathsf{IND\text{-}CPA}$ is not sufficient to prove any soundness result in presence of key cycles. To better understand the problem of key cycles suppose that $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is a semantically secure encryption scheme and let $\mathcal{SE}' = (\mathcal{KG}', \mathcal{E}', \mathcal{D}')$ be defined as follows:

$$\begin{aligned}
\mathcal{KG}' &= \mathcal{KG} \\
\mathcal{E}'_k(m, r) &= \begin{cases} \mathcal{E}_k(m, r) & \text{if } m \neq k \\ \mathsf{const} \cdot k & \text{if } m = k \end{cases} \\
\mathcal{D}'_k(c) &= \begin{cases} \mathcal{D}_k(c) & \text{if } c \neq \mathsf{const} \cdot k \\ k & \text{if } c = \mathsf{const} \cdot k \end{cases}
\end{aligned}$$

where const is a constant such that for any key k, the concatenation $\mathsf{const} \cdot k$ does not belong to the set of possible ciphertexts obtained by $\mathcal{E}$. Obviously, if the attacker is given a key cycle of length 1, *e.g.*, $\mathcal{E}'_k(k, r)$, the attacker directly learns the key. It is also easy to see that $\mathcal{SE}'$ is a semantic secure encryption scheme as it behaves as $\mathcal{SE}$ in nearly all cases (in the security experiment the adversary could make a query for encrypting k with itself only with negligible probability). Hence, as in numerous previous work we forbid the symbolic terms to contain such cycles. (Another possibility to handle key cycles is to consider stronger computational requirements like Key Dependent Message – KDM – security as done in [4].)

We now define a similar notion of key cycles in our setting. For any term t, let $kp(t)$ be the set of polynomials p such that g_2^p occurs at a key position in t and g_2^p is not deducible from t. Let $pm(t)$ be the set of monomials $x_1 x_2 x_3$ such that either:

- x_1, x_2 and x_3 occur as data in t;
- x_1 and x_2 occur as data in t and $g_1^{x_3}$ also appears in t;
- x_1 occurs as data in t and $g_1^{x_2}$ and $g_1^{x_3}$ also appear in t.

A term t is *acyclic* if the two following restrictions are verified.

- For any p in $kp(t)$, p is linearly independent from any other polynomials from $\mathsf{pol}\,(t)$ and from monomials from $pm(t)$, *i.e.*, if $\mathsf{pol}\,(t) = \{p, p_1, ..., p_n\}$ and $pm(t) = \{m_1, ..., m_{n'}\}$ then there does not exist any integers a, a_1 to a_n and b_1 to $b_{n'}$ such that $a \neq 0$ and:

$$a.p = \sum_{i=1}^{n} a_i p_i + \sum_{j=1}^{n'} b_j m_j$$

- There exists an order $\prec$ among keys used in t such that for any subterm $\{u\}_{key}$ of t, either key is deducible from t or for each key key' that occurs in u, $key' \prec key$.

We illustrate the notion of key cycles on several examples.

- The terms $\{k\}_k$ and $(\{k_1\}_{k_2}, \{k_2\}_{k_1})$ contain key cycles, as those considered already in [3].
- The term $t = (\{k\}_{g_2^{x_1x_2x_3}}, \{g_2^{x_1x_2x_3}\}_k)$ obviously contains a key cycle while $(\{k\}_{g_2^{x_1x_2x_3}}, \{g_2^{x_1x_2x_3}\}_{k'})$ does not.
- The term $t = (g_1^{x_1}, g_1^{x_2}, x_3, \{k\}_{g_2^{x_1x_2x_3}}, \{g_2^{x_1x_2x_3}\}_k)$ is acyclic as $g_2^{x_1x_2x_3}$ is deducible (and hence $kp(t) = \emptyset$).
- The term $t = \{(x_1, g_1^{x_2}, g_1^{x_3})\}_{g_2^{2x_1x_2x_3}}$ contains a key cycle because $pm(t) = \{x_1x_2x_3\}$ and $2x_1x_2x_3 \in kp(t)$ is linearly dependent.

Our acyclicity restriction is stronger than what is strictly required for computational soundness: for example $\{\{k\}_{k'}\}_k$ is considered as a cycle whereas it is not problematic as the underlying k is hidden by k'. We consider this stronger definition of acyclicity as it is easier to define and it also makes our main proof simpler.

5.1 Soundness Result

Indistinguishable Distributions. Before giving our soundness result, we recall the usual notion of indistinguishable distributions. Intuitively, two distributions D_1 and D_2 are computationally indistinguishable if for any adversary $\mathcal{A}$, the probability for $\mathcal{A}$ to detect the difference between a randomly sampled element of D_1 and a randomly sampled element of D_2 is negligible in η.

Definition 5.1 *Let D_1 and D_2 be two distributions (that depend on η). The advantage of an adversary $\mathcal{A}$ in distinguishing D_1 and D_2 is defined by:*

$$\mathrm{Adv}_{\mathcal{A}}^{D_1,D_2} = \mathbb{P}\left[x \leftarrow D_1(\eta)\ ;\ \mathcal{A}(x) = 1\right] - \mathbb{P}\left[x \leftarrow D_2(\eta)\ ;\ \mathcal{A}(x) = 1\right]$$

Distributions D_1 and D_2 are computationally indistinguishable*, written $D_1 \approx D_2$, if the advantage for any adversary $\mathcal{A}$ in distinguishing D_1 and D_2 is negligible.*

Then our main soundness result states that distributions related to equivalent terms are computationally indistinguishable.

Proposition 5.2 *Let t_0 and t_1 be two acyclic well-formed terms, such that $t_0 \cong t_1$. Let $\mathcal{SE}$ be a symmetric encryption scheme that is secure for* IND-CPA *and IG be an instance generator satisfying* BDDH*, then $[\![t_0]\!]_{\mathcal{SE},IG} \approx [\![t_1]\!]_{\mathcal{SE},IG}$.*

Proof for proposition 5.2

In order to prove our main soundness result, we introduce some intermediate lemmas. First we prove that BDDH implies an extended version of BDDH. Intuitively this first lemma states that if BDDH holds and $\mathcal{A}$ is an adversary that is given some exponents x_1 to x_α and some exponentiations $g_1^{y_1}$ to $g_1^{y_\beta}$, $\mathcal{A}$ cannot distinguish exponentiations of linearly independent polynomials $g_2^{p_1}$ to $g_2^{p_\gamma}$ from exponentiations of fresh exponents $g_2^{r_{1,1}r_{1,2}r_{1,3}}$ to $g_2^{r_{\gamma,1}r_{\gamma,2}r_{\gamma,3}}$.

Lemma 5.3 *Let $X = (x_i)_{1\leq i\leq\alpha}$ and $Y = (y_i)_{1\leq i\leq\beta}$ be $\alpha+\beta$ exponents. Let $P = (p_i)_{1\leq i\leq\gamma}$ be γ polynomials such that there are no linear relations between the p_i and the set of monomials $\{xyz,\ x,y,z \in X\} \cup \{xyz,\ x,y \in X,\ z \in Y\} \cup \{xyz,\ x \in X,\ y,z \in Y\}$.*

If IG is an instance generator satisfying BDDH *and the two following terms are well-formed then:*

$$[\![x_1,...,x_\alpha,g_1^{y_1},...,g_1^{y_\beta},g_2^{p_1},...,g_2^{p_\gamma}]\!]_{IG} \approx [\![x_1,...,x_\alpha,g_1^{y_1},...,g_1^{y_\beta},g_2^{q_1},...,g_2^{q_\gamma}]\!]_{IG}$$

where each q_i is a product of three fresh exponents $r_{i,1}r_{i,2}r_{i,3}$, i.e.*, the part of the distribution related to $g_2^{q_i}$ corresponds to a random group element.*

Proof. First, note that as the order q of the group $\mathbb{G}_2$ is prime, in the computational setting $g_2^{Z_1Z_2}$ and $g_2^{Z_3}$ have the same distribution (Z_1, Z_2, and Z_3 are three independent random variables uniformly sampled over $\mathbb{Z}_q$).

The proof of this lemma is done in two steps.

- The first step consists in replacing monomials in the p_i that are not in $dm(x_1,...,x_\alpha,g_1^{y_1},...,g_1^{y_\beta})$ with fresh monomials. This results in a new term whose computational distribution is indistinguishable from the original term distribution.
- Then, in the second step we prove that the computational distribution of this new term using fresh monomials is exactly equal to the distribution related to $x_1,\ldots,x_\alpha,g_1^{y_1},\ldots,g_1^{y_\beta},g_2^{q_1},\ldots,g_2^{q_\gamma}$.

Step 1. Let M be the set of monomials from P that are not in $\{xyz,\ x,y,z \in X\} \cup \{xyz,\ x,y \in X,\ z \in Y\} \cup \{xyz,\ x \in X,\ y,z \in Y\}$. The first step consists in using BDDH to replace these monomials with fresh monomials $r_1r_2r_3$. Let p'_1 to p'_γ be polynomials p_1 to p_γ where each monomial of M has been replaced with a fresh monomial. We prove that:

$$[\![x_1,\ldots,x_\alpha,g_1^{y_1},\ldots,g_1^{y_\beta},g_2^{p_1},\ldots,g_2^{p_\gamma}]\!]_{IG} \approx [\![x_1,\ldots,x_\alpha,g_1^{y_1},\ldots,g_1^{y_\beta},g_2^{p'_1},\ldots,g_2^{p'_\gamma}]\!]_{IG}$$

This proof is done by induction on the number j of monomials in M that use at least one exponent which is also present in X, Y or in any other monomial used in a polynomial from P.

If $j = 0$ then for each monomial m used in p_1 to p_γ, m uses exponents that are not in X or Y nor in any other monomial from polynomials of P. Thus p_1 to p_γ are equal to p'_1 to p'_γ up to renaming of the exponents and so:

$$[\![x_1,\dots,x_\alpha,g_1^{y_1},\dots,g_1^{y_\beta},g_2^{p_1},\dots,g_2^{p_\gamma}]\!]_{IG} = [\![x_1,\dots,x_\alpha,g_1^{y_1},\dots,g_1^{y_\beta},g_2^{p'_1},\dots,g_2^{p'_\gamma}]\!]_{IG}$$

If $j > 0$ then let $m = xyz$ be a monomial in M that uses an exponent from X or Y or from another monomial of P and let m' be a fresh monomial. Let p''_1 to p''_γ be polynomials p_1 to p_γ where m has been replaced with m'. There are two cases to consider:

- First if x, y and z do not appear in X. Let $\mathcal{A}$ be an adversary trying to distinguish distribution

$$[\![x_1,\dots,x_\alpha,g_1^{y_1},\dots,g_1^{y_\beta},g_2^{p_1},\dots,g_2^{p_\gamma}]\!]_{IG}$$

  from distribution

$$[\![x_1,\dots,x_\alpha,g_1^{y_1},\dots,g_1^{y_\beta},g_2^{p''_1},\dots,g_2^{p''_\gamma}]\!]_{IG}.$$

  We build an adversary $\mathcal{B}$ against BDDH that executes $\mathcal{A}$ as a subroutine. As $\mathcal{B}$ tries to break BDDH, $\mathcal{B}$ receives four arguments (A, B, C, D). Intuitively, $\mathcal{B}$ uses the inputs A, B, C and D for g_1^x, g_1^y, g_1^z and g_2^{xyz}. $\mathcal{B}$ queries $\mathcal{A}$ with the input

$$a_1,\dots,a_\alpha,b_1,\dots,b_\beta,c_1,\dots,c_\gamma$$

  where

  – a_i are values in $\mathbb{Z}_q$ randomly generated by $\mathcal{B}$;

  – b_i is computed as follows. If y_i equals x, y or z then b_i is set to A, B or C respectively. Otherwise b_i is set to g_1^u where $u = a_j$ if $y_i = x_j$ for some j or u is randomly sampled from $\mathbb{Z}_q$;

  – for each monomial m_0 appearing in p_i $\mathcal{B}$ computes the implementation for $g_2^{m_0}$ as follows. If $m_0 = m$ then $g_2^{m_0}$ is implemented by D. If m_0 shares two exponents with m, for example $m_0 = xyz'$, then the corresponding value is generated using the bilinear map: $\mathcal{B}$ computes $e(A, B)^c$ where c is either freshly generated by $\mathcal{B}$ or has been previously generated for z'. If m_0 only shares one exponent with m, for example $m_0 = xy'z'$, then $\mathcal{B}$ computes A^{bc} where where b and c are either freshly generated by $\mathcal{B}$ or have been previously genrated for y' and z'. Given the implementations of $g_2^{m_0}$ for each m_0 in p_i $\mathcal{B}$ computes implementations for each $g_2^{p_i}$ and use these values for $c_1,\dots,c_\gamma$.

Finally, $\mathcal{B}$ returns the same output as $\mathcal{A}$. The advantage of $\mathcal{B}$ against BDDH is given by

$$\mathrm{Adv}^{\mathsf{BDDH}}_{\mathcal{B},IG}(\eta) = \mathbb{P}\left[\mathcal{B}(g_1^x, g_1^y, g_1^z, g_2^{xyz}) = 1\right] - \mathbb{P}\left[\mathcal{B}(g_1^x, g_1^y, g_1^z, g_2^r) = 1\right]$$

When $\mathcal{B}$ receives as input $(g_1^x, g_1^y, g_1^z, g_2^{xyz})$, $\mathcal{A}$ is given a sample from distribution $[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p_1}, \ldots, g_2^{p_\gamma}]\!]_{IG}$.

When $\mathcal{B}$ receives as input $(g_1^x, g_1^y, g_1^z, g_2^r)$, $\mathcal{A}$ is given a sample from distribution $[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p''_1}, \ldots, g_2^{p''_\gamma}]\!]_{IG}$.

Therefore the advantage of $\mathcal{A}$ in distinguishing the two distributions is equal to the advantage of $\mathcal{B}$ against BDDH. As BDDH holds, the advantage of $\mathcal{B}$ is negligible and so the advantage of $\mathcal{A}$ is also negligible. Hence,

$$\begin{array}{c}[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p_1}, \ldots, g_2^{p_\gamma}]\!]_{IG} \\ \approx \\ [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p''_1}, \ldots, g_2^{p''_\gamma}]\!]_{IG}\end{array}$$

- If x appears in X, then by definition of M either y or z does not appear in X and Y. Let us suppose that it is y. Exponent y only appears in m and, as the order of the group is prime, we have the following equality between distributions:

  $$[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{m}]\!]_{IG} = [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{m'}]\!]_{IG}$$

  Moreover as the original terms are well-formed and x occurs as data, m does not share any exponent with other monomials used in P. Hence, we also obtain that:

  $$\begin{array}{c}[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p_1}, \ldots, g_2^{p_\gamma}]\!]_{IG} \\ \approx \\ [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p''_1}, \ldots, g_2^{p''_\gamma}]\!]_{IG}\end{array}$$

We have proved that:

$$[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p_1}, \ldots, g_2^{p_\gamma}]\!]_{IG} \approx [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p''_1}, \ldots, g_2^{p''_\gamma}]\!]_{IG}$$

where p''_1 to p''_n use $j-1$ monomials that use an exponent from $X \cup Y$. Hence using our induction hypothesis, we get that

$$[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p''_1}, \ldots, g_2^{p''_\gamma}]\!]_{IG} \approx [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p'_1}, \ldots, g_2^{p'_\gamma}]\!]_{IG}$$

And so we proved that

$$[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p_1}, \ldots, g_2^{p_\gamma}]\!]_{IG} \approx [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p'_1}, \ldots, g_2^{p'_\gamma}]\!]_{IG}$$

Step 2. Let n be the number of elements in M.

For the second step, we recall that (as q is prime), the number of solutions over $\mathbb{Z}_q$ for a linear system of γ independent equations involving n variables is $q^{n-\gamma}$.

Let $(\hat{x}_i)_{1\leq i\leq\alpha}$, $(\hat{y}_i)_{1\leq i\leq\beta}$ and $(\hat{p}_i)_{1\leq i\leq\gamma}$ be elements of $\mathbb{Z}_q$. We now compute the probability for the distribution to output value v defined by:

$$v = \left(\hat{x}_1, \ldots, \hat{x}_\alpha, g_1^{\hat{y}_1}, \ldots, g_1^{\hat{y}_\beta}, g_2^{\hat{p}_1}, \ldots, g_2^{\hat{p}_\gamma}\right)$$

It is important to see that this is a computational value and not a symbolic term.

Then we associate to each monomial from M a variable over $\mathbb{Z}_q$ and we obtain a system involving γ linear equations using n variables.

$$\left(\hat{x}_1, \ldots, \hat{x}_\alpha, g_1^{\hat{y}_1}, \ldots, g_1^{\hat{y}_\beta}, g_2^{p'_1}, \ldots, g_2^{p'_\gamma}\right) = v$$

(The system is given by the equations between $g_2^{\cdot}$ as the other equalities are trivially satisfied.) The number of solutions of this system is $q^{n-\gamma}$. Hence when randomly sampling values for monomials in M, the probability to obtain v is $q^{n-\gamma}/q^n$ which is equal to $q^{-\gamma}$.

On the other side, the probability to obtain v by randomly sampling γ group elements for the $g_2^{q_i}$ is also equal to $q^{-\gamma}$ so the distributions are identical:

$$[\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p'_1}, \ldots, g_2^{p'_\gamma}]\!]_{IG} = [\![x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{q_1}, \ldots, g_2^{q_\gamma}]\!]_{IG}$$

And so we obtain the expected result. $\square$

In order to introduce the following lemmas, we define the computational semantics of patterns (*i.e.*, terms using $\square$) by extending the semantics for terms with $[\![\square]\!]_{\mathcal{SE},IG} = 0$. Our second lemma states that evaluations of a term and of its pattern are indistinguishable in the computational setting.

Lemma 5.4 *Let t be an acyclic well-formed term. Let $\mathcal{SE}$ be an* IND-CPA *secure symmetric encryption scheme and let IG be an instance generator satisfying* BDDH. *Then we have that*

$$[\![t]\!]_{\mathcal{SE},IG} \approx [\![\mathsf{pat}\,(t, K(t))]\!]_{\mathcal{SE},IG}$$

Proof. Let t be an acyclic well-formed term. Then any p in $kp(t)$ is linearly independent of any other polynomials from $\mathsf{pol}\,(t)$. Let $\overline{K(t)}$ be the set of keys and exponentiations g_2^p used at a key position in t that are not in $K(t)$, *i.e.*, that are not deducible. Let *key* be a metavariable over $\overline{K(t)}$. As t is acyclic there exists a total order $\prec$ between elements of $\overline{K(t)}$ such that for any subterm $\{t'\}_{key}$ of t, key' can only appear in t' if $key' \prec key$.

This proof follows the lines of the main proof in [3]. The main difference with the original proof is that keys can be an exponentiation g_2^p. However as p is not involved in any linear relation, using this key is indistinguishable from using an atomic key.

Let us now detail the proof. Let n be the number of keys in $\overline{K(t)}$ and let key_i be the i^{th} key from $\overline{K(t)}$ with respect to $\prec$, *i.e.*:

$$\overline{K(t)} = \{key_1, \ldots, key_n\} \text{ and } key_1 \prec key_2 \prec \ldots \prec key_n$$

For i in $[0, n]$, term t_i is defined as $pat_i(t)$ where pat_i is recursively defined by:

$$\begin{array}{lcll} pat_i((t_1, t_2)) & = & \big(pat_i(t_1), pat_i(t_2)\big) & \\ pat_i(\{t'\}_{key}) & = & \{\Box\}_{key} & \text{if } key = key_j \text{ for } j \leq i \\ pat_i(\{t'\}_{key}) & = & \{pat_i(t')\}_{key} & \text{else} \\ pat_i(a) & = & a & \text{for } a \text{ in } x,\ k,\ g_1^x \text{ and } g_2^p \end{array}$$

In t_i, encryptions using keys key_j for $j \leq i$ have been replaced by encryptions of $\Box$. Hence $pat_0(t) = t$ and $pat_n(t) = \mathsf{pat}\,(t, K(t))$. The advantage of an adversary $\mathcal{A}$ which tries to distinguish $[\![t]\!]_{\mathcal{SE},IG}$ and $[\![\mathsf{pat}\,(t, K(t))]\!]_{\mathcal{SE},IG}$ can be written as:

$$\begin{array}{rl} & \mathrm{Adv}_{\mathcal{A}}^{[\![t]\!]_{\mathcal{SE},IG},[\![\mathsf{pat}(t,K(t))]\!]_{\mathcal{SE},IG}} = \mathrm{Adv}_{\mathcal{A}}^{[\![t_0]\!]_{\mathcal{SE},IG},[\![t_n]\!]_{\mathcal{SE},IG}} \\ = & \mathbb{P}\,[x \leftarrow [\![t_0]\!]_{\mathcal{SE},IG}\ ;\ \mathcal{A}(x) = 1] - \mathbb{P}\,[x \leftarrow [\![t_n]\!]_{\mathcal{SE},IG}\ ;\ \mathcal{A}(x) = 1] \\ = & \displaystyle\sum_{i=1}^{n} (\mathbb{P}\,[x \leftarrow [\![t_{i-1}]\!]_{\mathcal{SE},IG}\ ;\ \mathcal{A}(x) = 1] - \mathbb{P}\,[x \leftarrow [\![t_i]\!]_{\mathcal{SE},IG}\ ;\ \mathcal{A}(x) = 1]) \\ = & \displaystyle\sum_{i=1}^{n} \mathrm{Adv}_{\mathcal{A}}^{[\![t_{i-1}]\!]_{\mathcal{SE},IG},[\![t_i]\!]_{\mathcal{SE},IG}} \end{array}$$

We build n adversaries $(\mathcal{B}_i)_{1 \leq i \leq n}$ against IND-CPA that use $\mathcal{A}$ as a subroutine and such that the advantage of $\mathcal{B}_i$ against IND-CPA can be linked to the advantage $\mathrm{Adv}_{\mathcal{A}}^{[\![t_{i-1}]\!]_{\mathcal{SE},IG},[\![t_i]\!]_{\mathcal{SE},IG}}$.

Each adversary $\mathcal{B}_i$ uses his challenge key for key key_i and has access to a left-right encryption oracle $LR^b_{\mathcal{SE}}$. If key key_i is an exponentiation g_2^p then as p is not involved in any linear relation and because of lemma 5.3, the evaluation g_2^p is indistinguishable from a random group element. The key extraction algorithm Kex applied to a random group element returns a random key (whose distribution corresponds to the one of $\mathcal{KG}$). Hence, using g_2^p is indistinguishable from using a fresh atomic key.

Adversary $\mathcal{B}_i$ generates values for each atom used in t. For any subterm a of t which is of the form x, k, g_1^x or g_2^p, $\mathcal{B}_i$ computes a bit-string value bs_a according to the values generated previously. Using his left-right encryption oracle, $\mathcal{B}_i$ computes a bit-string bs which is either an evaluation of t_i or an evaluation of t_{i-1} depending on the challenge bit b. Formally bit-string bs is obtained by applying the recursive $eval_i$ function on t_{i-1}:

$$\begin{array}{lcll} eval_i((t_1, t_2)) & = & eval_i(t_1) \cdot eval_i(t_2) & \\ eval_i(\{t'\}_{key_i}) & = & LR^b_{\mathcal{SE}}(0, eval_i(t')) & \\ eval_i(\{t'\}_{key}) & = & \mathcal{E}(eval_i(t'), eval_i(key)) & \text{for } key \neq key_i \\ eval_i(\Box) & = & 0 & \\ eval_i(a) & = & bs_a & \text{for } a \text{ in } x,\ k,\ g_1^x \text{ and } g_2^p \end{array}$$

This algorithm works well as due to acyclicity key_i does not occur as data in t_{i-1}. Note that oracle $LR^b_{\mathcal{SE}}$ can be given as arguments two bit-strings of different lengths. This is why we had to assume that encryption scheme $\mathcal{SE}$ is length-concealing.

After computing bs, $\mathcal{B}_i$ executes $\mathcal{A}$ with input bs and returns the same result as $\mathcal{A}$. Let us sum up how $\mathcal{B}_i$ works:

```
**Adversary** $\mathcal{B}_i^{LR^b_{\mathcal{SE}}}(\eta)$
    **for each** $a$, compute $bs_a$
    $bs \leftarrow eval_i(t_{i-1})$
    $d \leftarrow \mathcal{A}(bs)$
    **return** $d$
```

If bit b equals 0, then $\mathcal{A}$ is confronted to an evaluation of t_i whereas if b equals 1, then $\mathcal{A}$ is given an evaluation of t_{i-1}. The advantages of $\mathcal{A}$ and $\mathcal{B}_i$ can be linked in the following way:

$$\mathrm{Adv}_{\mathcal{A}}^{[\![t_{i-1}]\!]_{\mathcal{SE},IG},[\![t_i]\!]_{\mathcal{SE},IG}} = \mathrm{Adv}_{\mathcal{SE},\mathcal{B}_i}^{\mathsf{CPA}}$$

Therefore we have that:

$$\mathrm{Adv}_{\mathcal{A}}^{[\![t]\!]_{\mathcal{SE},IG},[\![\mathsf{pat}(t,K(t))]\!]_{\mathcal{SE},IG}} = \sum_{i=1}^{n} \mathrm{Adv}_{\mathcal{SE},\mathcal{B}_i}^{\mathsf{CPA}}$$

As $\mathcal{SE}$ is assumed to be $\mathsf{IND\text{-}CPA}$ secure, the advantage of $\mathcal{B}_i$ is negligible for any i. Hence the advantage of $\mathcal{A}$ is also negligible. $\square$

Our third lemma states that two patterns equal up to renaming are also indistinguishable in the computational setting.

Lemma 5.5 *Let t_0 and t_1 be two well-formed terms such that $\mathsf{pat}\,(t_0, K(t_0)) \cong \mathsf{pat}\,(t_1, K(t_1))$. Let $\mathcal{SE}$ be a symmetric encryption scheme (not necessarily secure) and let IG be an instance generator satisfying* BDDH*, then*

$$[\![\mathsf{pat}\,(t_0, K(t_0))]\!]_{\mathcal{SE},IG} \approx [\![\mathsf{pat}\,(t_1, K(t_1))]\!]_{\mathcal{SE},IG}$$

Proof. Let t'_0 be the term $\mathsf{pat}\,(t_0, K(t_0))$ and t'_1 be the term $\mathsf{pat}\,(t_1, K(t_1))$. There exists a renaming of **Keys** σ_1 and a bijection σ_2 preserving linear relations between polynomials from t_1 to t_0 such that $t'_0 = t'_1\sigma_1\sigma_2$. Permutation of keys is easy to handle: $[\![t'_1\sigma_1]\!]_{\mathcal{SE},IG}$ and $[\![t'_1]\!]_{\mathcal{SE},IG}$ output exactly the same distribution.

There only remains to prove that $[\![t'_0]\!]_{\mathcal{SE},IG} \approx [\![t'_1\sigma_1]\!]_{\mathcal{SE},IG}$. For this purpose, let $u_0 = t'_0$ and $u_1 = t'_1\sigma_1$. Let $\mathcal{A}$ be an adversary trying to distinguish the distribution related to u_0 from the distribution related to u_1. In the remaining, we prove that the advantage of $\mathcal{A}$ is negligible if the BDDH assumption holds. For this purpose, we introduce a term u such that:

$$\mathrm{Adv}_{\mathcal{A}}^{[\![u_1]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}} = \mathrm{Adv}_{\mathcal{A}}^{[\![u_1]\!]_{\mathcal{SE},IG},[\![u]\!]_{\mathcal{SE},IG}} + \mathrm{Adv}_{\mathcal{A}}^{[\![u]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}}$$

Intuitively u is equal to u_0 where polynomials have been replaced by fresh monomials whenever possible while conserving linear equalities. u is also equal to u_1 where the same modification has been applied. From there, due to the nature of u it is easy to prove that the two advantages on the right part are negligible using lemma 5.3.

First let us define the following sets:

1. Let $X = (x_i)_{1 \leq i \leq \alpha}$ be the exponents that are deducible from u_0 (using u_1 instead of u_0 would give exactly the same X as $u_0 = u_1\sigma_2$).
2. Let $Y = (y_i)_{1 \leq i \leq \beta}$ be the exponents such that $g_1^{y_i}$ is deducible from u_0 (as previously, using u_1 instead of u_0 would give exactly the same Y).
3. Let $M = (m_i)_{1 \leq i \leq \delta}$ be the set of monomials $dm(u_0)$ which can easily be obtained from X and Y.
4. The two sets of polynomials $P_0 = (p_{0,i})_{1 \leq i \leq \gamma}$ and $P_1 = (p_{1,i})_{1 \leq i \leq \gamma}$ are built as follows:
   - Initially P_0 and P_1 are empty.
   - For each polynomial p such that g_2^p is a sub-term of u_0 at position q, we have that the sub-term of u_1 at position q is also an exponentiation $g_2^{p'}$.
   - If p is not involved in any linear relation with polynomials from the current P_0 and monomials from M, then p is appended to P_0 and p' is appended to P_1. Note that in this case, p' is not involved in any linear relation with polynomials from the current P_1 and monomials from M neither.

Let σ and σ' be the polynomial bijections defined respectively on polynomials p such that g_2^p occurs in term u_0 for σ and on polynomials p such that g_2^p occurs in term u_1 for σ'. These two bijections are defined by:

- For $p_{0,i}$ in P_0 , $p_{0,i}\sigma$ is defined as a fresh monomial $r_{1,i}r_{2,i}r_{3,i}$.
- For $p_{1,i}$ in P_1 , $p_{1,i}\sigma'$ is defined as a fresh monomial $r_{1,i}r_{2,i}r_{3,i}$.
- Let p be a polynomial such that g_2^p occurs in u_0 and such that p is not in P_0. Then by definition of P_0, p is linked via a linear relation to polynomials in P_0 and monomials in M:

  $$p = \sum_{j=1}^{\gamma} \lambda_j p_{0,j} + \sum_{j=1}^{\delta} \mu_j m_j$$

  And we define $p\sigma$ as

  $$p\sigma = \sum_{j=1}^{\gamma} \lambda_j \left(p_{0,j}\sigma\right) + \sum_{j=1}^{\delta} \mu_j m_j$$

- In a similar way, let p be a polynomial such that g_2^p occurs in u_1 and such that p is not in P_1. Then p is linked via a linear relation to polynomials in P_1 and monomials in M:

  $$p = \sum_{j=1}^{\gamma} \lambda_j p_{1,j} + \sum_{j=1}^{\delta} \mu_j m_j$$

  And we define $p\sigma'$ as

  $$p\sigma' = \sum_{j=1}^{\gamma} \lambda_j \left(p_{1,j}\sigma'\right) + \sum_{j=1}^{\delta} \mu_j m_j$$

Let u be the term $u_0\sigma$. As σ_2 is linear relation preserving, u is equal to $u_1\sigma'$. Then the advantage of $\mathcal{A}$ can be written as:

$$\mathrm{Adv}_{\mathcal{A}}^{[\![u_1]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}} = \mathrm{Adv}_{\mathcal{A}}^{[\![u_1]\!]_{\mathcal{SE},IG},[\![u]\!]_{\mathcal{SE},IG}} + \mathrm{Adv}_{\mathcal{A}}^{[\![u]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}}$$

We now prove that the advantage $\mathrm{Adv}_{\mathcal{A}}^{[\![u]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}}$ is negligible. The proof that $\mathrm{Adv}_{\mathcal{A}}^{[\![u_1]\!]_{\mathcal{SE},IG},[\![u]\!]_{\mathcal{SE},IG}}$ is also negligible is similar. Let w and w' be the two terms

$$\begin{array}{rcl} w & = & (x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{p_1}, \ldots, g_2^{p_\gamma}) \\ w' & = & (x_1, \ldots, x_\alpha, g_1^{y_1}, \ldots, g_1^{y_\beta}, g_2^{r_{1,1}r_{2,1}r_{3,1}}, \ldots, g_2^{r_{1,\gamma}r_{2,\gamma}r_{3,\gamma}}) \end{array}$$

We build an adversary $\mathcal{B}$ that tries to distinguish $[\![w]\!]_{IG}$ from $[\![w']\!]_{IG}$ and that uses $\mathcal{A}$ as a subroutine. $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives as argument a bit-string tuple $(X_1, \ldots, X_\alpha, Y_1, \ldots, Y_\beta, P_1, \ldots, P_\gamma)$ which is either generated by $[\![w]\!]_{IG}$ or by $[\![w']\!]_{IG}$.

2. $\mathcal{B}$ generates bit-string value bs_k for any atomic key k used in u using $\mathcal{KG}$ (these keys are also the ones used in u_0).

3. $\mathcal{B}$ recursively computes a bit-string bs' which is either an evaluation of u (in case $\mathcal{B}$ received as input an evaluation of w') or an evaluation of u_0 (in case $\mathcal{B}$ received as input an evaluation of w). The computation of bs' is done recursively on the structure of u by using the *eval* algorithm:

   - If u is a pair (v, w), then $eval(u) = eval(v) \cdot eval(w)$.
   - If u is an encryption $\{v\}_{key}$, then $eval(u) = \mathcal{E}(eval(v), eval(key))$.
   - If u is an atomic key k, then $eval(u) = bs_k$.
   - If u is an exponent x_i, then $eval(u) = X_i$.
   - If u is an exponentiation $g_1^{y_i}$, then $eval(u) = Y_i$.
   - If u is an exponentiation $g_2^{p_i}$, then $eval(u) = P_i$.

4. Then $\mathcal{B}$ executes $\mathcal{A}$ with bs' as input and returns the same output as $\mathcal{A}$.

We have the following relation among the advantages of $\mathcal{A}$ and $\mathcal{B}$:

$$\mathrm{Adv}_{\mathcal{A}}^{[\![u]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}} = \mathrm{Adv}_{\mathcal{B}}^{[\![w']\!]_{IG},[\![w]\!]_{IG}}$$

As BDDH holds, we apply lemma 5.3 and obtain that the advantage of $\mathcal{B}$ is negligible and hence $\mathrm{Adv}_{\mathcal{A}}^{[\![u]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}}$ is negligible.

Thus $\mathrm{Adv}_{\mathcal{A}}^{[\![u_1]\!]_{\mathcal{SE},IG},[\![u_0]\!]_{\mathcal{SE},IG}}$ is also negligible and we finally obtain that:

$$[\![\mathsf{pat}\,(t_0, K(t_0))]\!]_{\mathcal{SE},IG} \approx [\![\mathsf{pat}\,(t_1, K(t_1))]\!]_{\mathcal{SE},IG}$$

$\square$

It is now easy to obtain our main result by using transitivity of the $\approx$ relation. Let t_0 and t_1 be two acyclic well-formed terms. Let $\mathcal{SE}$ be an $\mathsf{IND\text{-}CPA}$ secure symmetric encryption scheme and let IG be an instance generator satisfying BDDH. Then we have:

$$[\![t_0]\!]_{\mathcal{SE},IG} \approx [\![\mathsf{pat}\,(t_0, K(t_0))]\!]_{\mathcal{SE},IG} \approx [\![\mathsf{pat}\,(t_1, K(t_1))]\!]_{\mathcal{SE},IG} \approx [\![t_1]\!]_{\mathcal{SE},IG}$$

The previous result states soundness of symbolic equivalence in the computational world. However, the reciprocal (*i.e.*, completeness) is false in general. There are two main problems that prevent completeness. First, the symmetric encryption scheme may allow decryption with the wrong key and output a random bit-string in that case. Then the distributions related to terms $(\{x\}_k, k)$ and $(\{x\}_k, k')$ can be computationally indistinguishable, even though these two terms do not have the same pattern. This can be solved by requiring symmetric encryption to be confusion free [33, 2] or to admit weak key-authenticity tests for expressions [33, 2, 26]. The second problem is that the symmetric encryption scheme can satisfy key concealing (this is ensured by type 0 security in [3]). Then the distributions related to terms $(\{0\}_k, \{0\}_{k'})$ and $(\{0\}_k, \{0\}_k)$ are computationally indistinguishable but these terms are not equivalent even with renaming. To solve this, one can either ask the encryption scheme to be key revealing or modify the pattern definition in order to hide the key name (but the encryption scheme has to be key concealing in order to prove soundness). Soundness and completeness results when symmetric encryption is key and length revealing are given in [5].

The previous proposition considers the case of equivalence and is typically used to verify security of key-exchange protocols. In the next proposition, we are interested in completeness for deducibility. We prove even more than completeness: if t is deducible from E then there exists an efficient algorithm which is able to build an evaluation of t from an evaluation of E with probability 1. This result can be used to verify that a key-agreement protocol can really be implemented in the computational setting: we first check that the shared key is deducible from the knowledge of any participants in the symbolic setting, then applying the following proposition tells us that there exists an efficient algorithm to obtain the shared key from the participant knowledge in the computational setting.

Proposition 5.6 *Let E be a finite set of terms t_1 to t_n and t be a term that does not use any encryption (*e.g.*, a modular exponentiation). If $E \vdash t$ then there exists a polynomial-time (with respect to the security parameter η) algorithm A such that $A\left([\![(t_1,\ldots,t_n)]\!]_{\mathcal{SE},IG}\right)$ outputs the evaluation of t using values for exponents and keys that have been generated to compute $[\![(t_1,\ldots,t_n)]\!]_{\mathcal{SE},IG}$, i.e.:*

$$(bs, bs') \leftarrow [\![((t_1,\ldots,t_n),t)]\!]_{\mathcal{SE},IG} \ : \ A(bs) = bs'$$

Proof. Let t be a term and E be a finite set of terms such that $E \vdash t$. First note that the structure of the proof of $E \vdash t$ does not depend on the security parameter η.

Each deduction rule from the symbolic setting corresponds to an operation which is tractable in the computational setting in polynomial-time in η using a deterministic algorithm (note that the deducibility relation does not give the adversary the ability to encrypt data). Hence it is easy to build the algorithm A by following the structure of a proof of $E \vdash t$. We nevertheless need to restrict ourselves to the case where t does not contain any encryption, as the concrete algorithm for encryption is not deterministic: we indeed have that $\{\{0\}_k\} \vdash \{0\}_k$ while in the computational setting $(bs, bs') \leftarrow [\![(\{0\}_k, \{0\}_k)]\!]_{\mathcal{SE},IG}$ yields two different biststrings bs and bs' as the encryption algorithm is run twice. □

Note that it is not necessary for terms to be well-formed or acyclic in this proposition.

6 Examples of Application

Now we illustrate how proposition 5.2 can be used to prove a key-exchange protocol secure in the computational world.

Our notion of security is strong secrecy of the shared key in the passive setting: the adversary gets to observe messages exchanged between the participants and has to distinguish the shared key from a random group element. In the symbolic world, let us suppose that the exchanged terms were t_1 to t_n and that the shared key is g_2^p. Then security in the symbolic setting holds if:

$$(t_1, ..., t_n, g_2^p) \approx (t_1, ..., t_n, g_2^{r_1 r_2 r_3})$$

where r_1, r_2 and r_3 are three fresh exponent names. It is then possible to apply proposition 5.2 in order to prove security in the computational setting.

We are also interested in executability of key exchange protocols. A protocol is executable if it is feasible for any participant to compute the shared key from his knowledge. Let us again suppose that the exchanged terms are t_1 to t_n and that the shared key is g_2^p. Moreover let $x_i^1, ..., x_i^{k_i}$ be the exponents which are generated by the i^{th} participant. The protocol is executable in the symbolic setting if for any i,

$$t_1, ..., t_n, x_i^1, ..., x_i^{k_i} \vdash g_2^p$$

Executability in the computational world can easily be obtained from here by applying proposition 5.6.

6.1 Joux Protocol

The Joux protocol has been described in section 2. In an execution of this protocol, three messages are sent, corresponding to terms $g_1^{x_1}$, $g_1^{x_2}$ and $g_1^{x_3}$. The shared key is $g_2^{x_1x_2x_3}$. Strong secrecy for this key-exchange protocol has been given as an example for our symbolic equivalence notion:

$$(g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_2^{x_1x_2x_3}) \cong (g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_2^{x'_1x'_2x'_3})$$

Proposition 5.2 can be applied to show that this protocol is secure in the computational setting if the BDDH assumption holds.

We also verify that this protocol is executable. In the symbolic setting this is the case as we have the following deducibility relation:

$$x_1, g_1^{x_2}, g_1^{x_3} \vdash g_2^{x_1x_2x_3}$$

Similar relations hold when permuting the roles of x_1 and x_2 and of x_1 and x_3. Thus proposition 5.6 proves that there exists an efficient algorithm in the computational setting which allows each participant to compute his shared secret key.

6.2 TAK-2 and TAK-3 Protocols

The TAK-2 and TAK-3 protocols are two variants of the Joux protocol which were proposed by Al-Riyami and Paterson in [6]. TAK-1 and TAK-2 are tripartite key-exchange protocols which work in the same way, the only difference lies in the shared key. These protocols uses certificates to provide authentication. However as we are only interested in indistinguishability of the shared key, we use a simplified version of the protocol. Let A, B and C be three participants:

$$\begin{array}{llll} (1)\ A & \rightarrow & B, C & : & (g_1^{x_1}, g_1^{y_1}) \\ (2)\ B & \rightarrow & A, C & : & (g_1^{x_2}, g_1^{y_2}) \\ (3)\ C & \rightarrow & A, B & : & (g_1^{x_3}, g_1^{y_3}) \end{array}$$

In TAK-2, the shared key is $g_2^{x_1x_2y_3+x_1y_2x_3+y_1x_2x_3}$. In TAK-3, $g_2^{x_1y_2y_3+y_1x_2y_3+y_1y_2x_3}$ is used as shared key. Our simplified version of the two protocols are quite close as we do not make any difference between short-term secrets (y_1, y_2 and y_3) and long-term secrets (x_1, x_2 and x_3). Thus in our setting it is sufficient to analyze one of the protocol, TAK-2 for example.

Security. In the symbolic setting, strong secrecy of the key generated by the TAK-2 protocol comes from the following equivalence (up to renaming). Note that the two equivalent terms are trivially well-formed and acyclic:

$$\left(g_1^{x_1}, g_1^{y_1}, g_1^{x_2}, g_1^{y_2}, g_1^{x_3}, g_1^{y_3}, g_2^{x_1x_2y_3+x_1y_2x_3+y_1x_2x_3}\right) \\ \cong \\ \left(g_1^{x_1}, g_1^{y_1}, g_1^{x_2}, g_1^{y_2}, g_1^{x_3}, g_1^{y_3}, g_2^{x'_1x'_2x'_3}\right)$$

This equivalence is true because the set of deducible monomials dm is empty for both terms and neither $x_1x_2y_3 + x_1y_2x_3 + y_1x_2x_3$ in the first term nor $x'_1x'_2x'_3$ in the second term is involved in a linear relation. Hence by using proposition 5.2, we obtain that in the computational setting an adversary that has access to values for $g_1^{x_1}$, $g_1^{y_1}$, $g_1^{x_2}$, $g_1^{y_2}$, $g_1^{x_3}$ and $g_1^{y_3}$ cannot distinguish the shared key $g_2^{x_1x_2y_3+x_1y_2x_3+y_1x_2x_3}$ from a random group element, so the adversary is not able to obtain a single bit of information on the shared key.

Executability. We also verify executability of the protocol. By symmetry we consider the case of A. A generates two exponents x_1 and y_1 and receives two messages corresponding to terms $(g_1^{x_2}, g_1^{y_2})$ and $(g_1^{x_3}, g_1^{y_3})$. Hence executability in the symbolic setting is a consequence of the following deduction:

$$x_1, y_1, g_1^{x_2}, g_1^{y_2}, g_1^{x_3}, g_1^{y_3} \vdash g_2^{x_1x_2y_3+x_1y_2x_3+y_1x_2x_3}$$

Thus proposition 5.6 proves that there exists an efficient algorithm in the computational setting which allows participant A to compute his shared secret key. The same thing holds for B and C.

Active attacks. The TAK protocol family was designed to be secure even in the presence of an active adversary. However, as shown by Shim [37], TAK-2 is vulnerable to active attacks (the other variants are subject to similar attacks). Completely defining a formal model for active adversaries is outside the scope of this paper. Nevertheless the *role* of participant A could be described as follows:

$$\begin{array}{l}\texttt{send}(g_1^a, g_1^\alpha)\\ \texttt{recv}(g_1^{x_B}, g_1^\beta)\\ \texttt{recv}(g_1^{x_C}, g_1^\gamma)\end{array}$$

where a is a fresh exponent generated by A, x_B and x_C are variables and $g_1^\alpha, g_1^\beta, g_1^\gamma$ are the public keys which aim at guaranteeing authenticity. The key computed by A corresponds to $K_A = g_2^{ax_B\gamma + a\beta x_C + \alpha x_B x_C}$. An active adversary can substitute x_B and x_C by two fresh names b' and c' yielding an attack: the key computed by A, $g_2^{ab'\gamma + a\beta c' + \alpha b'c'}$, is indeed deducible from the attacker's knowledge $\{g_1^a, g_1^\alpha, b', g_1^{b'}, g_1^\beta, c', g_1^{c'}, g_1^\gamma\}$. It follows directly from proposition 5.6 that this symbolic attack can be efficiently implemented by a computational adversary. More generally, active symbolic attacks aiming at *deducing* the key (weak secrecy) correspond to computational attacks. This is not surprising and the converse is obviously not true: it does not follow from our results that a symbolic security proof (in the presence of an active attacker) gives a computational security guarantee.

6.3 A Variant of the Burmester-Desmedt Protocol using Pairings

As an additional example which illustrates the scope of our results, we show how to apply our results to a variant of the group key exchange protocol introduced by Burmester and Desmedt in [14]. The aim of this protocol is to establish a secret key shared among the members of the group. It is scalable as it requires only two rounds and a constant number of modular exponentiation per user. This protocol is only designed for security against passive adversaries.

The Original Burmester-Desmedt Protocol. Consider a network in which members of a group can broadcast messages to each other. Let η be a security parameter and let A_1,A_2,$\cdots$,A_n, for $n \in \mathbb{Z}$, be members of a group. We fix the security parameter η, a finite cyclic group $\mathbb{G}$ of generator g and of prime order q. These parameters $\mathbb{G}$, g and q are published.

- **Round 1:** Each participant A_i samples a random $x_i \in \mathbb{Z}_q$, and broadcasts $Z_i = g^{x_i}$.
- **Round 2:** Each participant A_i broadcasts $X_i = (Z_{i+1}/Z_{i-1})^{x_i} = g^{x_i x_{i+1} - x_{i-1} x_i}$, where the indexes are taken modulo n.
- **Key computation:** Each party A_i computes the shared key $K = g^{\sum_{i=1}^{n} x_i x_{i+1}}$.

The Bilinear Burmester-Desmedt Protocol. Now we define a family of variants of the Burmester-Desmedt protocol. Protocols in this family are parameterized by three integers α, β and γ such that $\alpha + \beta + \gamma = 0$ and either α, β or γ is different from 0. The instance of the protocol corresponding to α, β and γ is denoted by α, β, γ-BBD (Bilinear Burmester-Desmedt).

We still consider a group of n members A_1 to A_n. This time the protocol does not use a single cyclic group but uses a bilinear pairing between two cyclic groups. Hence we fix the security parameter η and two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q with respective generators g_1 and g_2, as well as a pairing operation e from $\mathbb{G}_1 \times \mathbb{G}_1$ to G_2 such that $e(g_1, g_1) = g_2$.

- **Round 1:** Each participant A_i samples a random $x_i \in \mathbb{Z}_q$, and broadcasts $Z_i = g_1^{x_i}$.
- **Round 2:** Each participant A_i broadcasts X_i defined by

  $$\begin{array}{rcl} X_i & = & e(Z_{i-2}, Z_{i-1})^{\alpha x_i} e(Z_{i-1}, Z_{i+1})^{\beta x_i} e(Z_{i+1}, Z_{i+2})^{\gamma x_i} \\ & = & g_2^{\alpha x_{i-2} x_{i-1} x_i + \beta x_{i-1} x_i x_{i+1} + \gamma x_i x_{i+1} x_{i+2}} \end{array}$$

  where the indexes are still taken modulo n.
- **Key computation:** Each party A_i computes the shared key

  $$K = g_2^{\sum_{i=1}^{n} x_i x_{i+1} x_{i+2}}$$

Security Analysis. We first prove strong secrecy for the shared key in the symbolic setting. This secrecy property is defined as the equivalence between the protocol execution transcript concatenated to the shared key and the transcript concatenated with a random group element from $\mathbb{G}_2$. Hence α, β, γ-BBD verifies strong secrecy of the shared key in the symbolic setting iff the following equivalence holds:

$$(Z_1, ..., Z_n, X_1, ..., X_n, K) \cong (Z_1, ..., Z_n, X_1, ..., X_n, g_2^{r_1 r_2 r_3})$$

In order to obtain this equivalence, we use the following lemma which proves that the exponent used in the key is linearly independent from other exponents if $\alpha + \beta + \gamma = 0$.

Lemma 6.1 *Let α, β, γ and n be four integers. Let V be a real vector space and u_1 to u_n be n linearly independent elements of V. If $\alpha + \beta + \gamma = 0$, then $\sum_{i=1}^{n} u_i$ is linearly independent from the family of vectors $(\alpha u_i + \beta u_{i+1} + \gamma u_{i+2})_i$ (indexes are taken modulo n).*

Proof. Let U be the set of vectors $(\alpha u_i + \beta u_{i+1} + \gamma u_{i+2})_i$ for i between 1 and n. For any vector v in $span(U)$, there exists a unique decomposition $v = \sum_{i=1}^{n} \lambda_i u_i$ and $\sum_{i=1}^{n} \lambda_i$ is equal to 0. Hence $\sum_{i=1}^{n} u_i$ is not in $span(U)$ and is linearly independent from vectors in U. $\square$

A direct consequence of this is strong secrecy of α, β, γ-BBD in the symbolic setting. By applying proposition 5.2, we obtain strong secrecy of the key in the computational setting for a passive adversary.

7 Conclusions and Future Work

We have proposed a first symbolic model to analyze cryptographic protocols which use a bilinear pairing. This model can be used to verify security of well-known key-exchange protocols using pairing like Joux protocol or the TAK-2 and TAK-3 protocol. Moreover our symbolic model consists in an extension of Abadi-Rogaway logic which is computationally sound provided that the encryption scheme and the pairing satisfy classical requirements from provable security. A direct consequence of this soundness result is that the Joux, TAK-2 and TAK-3 protocol are also secure in the computational setting. We also design a variant based on pairings of the Burmester-Desmedt protocol and prove its security against passive adversaries.

This paper only consider passive adversaries. An obvious line for future work is to extend the results to deal with active adversaries. Another interesting follow-up would be to investigate completeness of the extended version of Abadi-Rogaway logic as in [33]. However this would require either to tighten the symbolic model or to use stronger versions of the computational requirements IND-CPA and BDDH.

References

[1] M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Proc. 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2006.

[2] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4th International Symposium on Theoretical Aspects of Computer Software (TACS'01)*, volume 2215 of *Lecture Notes in Computer Science*. Springer, 2001.

[3] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proc. IFIP International Conference on Theoretical Computer Science (IFIP TCS'00)*. Springer, 2000.

[4] P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396. Springer, 2005.

[5] P. Adão, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 170–184. IEEE, 2005.

[6] S. S. Al-Riyami and K. G. Paterson. Tripartite authenticated key agreement protocols from pairings. In *Proc. 9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 332–359. Springer, 2003.

[7] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proc. 10th conference on Computer and Communication Security (CCS'03)*, pages 220–230. ACM, 2003.

[8] G. Bana, P. Mohassel, and T. Stegers. The computational soundness of formal indistinguishability and static equivalence. In *Proc. 11th Asian Computing Science Conference (ASIAN'06)*, volume 4435 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2008.

[9] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – CRYPTO'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.

[10] R. Barua, R. Dutta, and P. Sarkar. Extending Joux's protocol to multi party key agreement (extended abstract). In *Proc. 4th International Conference on Cryptology in India (INDOCRYPT'03)*, volume 2904 of *Lecture Notes in Computer Science*, pages 205–217. Springer, 2003.

[11] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer, 2005.

[12] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[13] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi. A generalization of DDH with applications to protocol analysis and computational soundness. In *Advances in Cryptology – CRYPTO'07*, volume 4622 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2007.

[14] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system (extended abstract). In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1994.

[15] R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In *Proc. Theory of Cryptography Conference (TCC'06)*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.

[16] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2003.

[17] O. Chevassut, P.-A. Fouque, P. Gaudry, and D. Pointcheval. Key derivation and randomness extraction. Technical Report 2005/061, Cryptology ePrint Archive, 2005. `http://eprint.iacr.org/`.

[18] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[19] A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proc. 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 321–334. IEEE, 2006.

[20] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 1983.

[21] R. Dutta, R. Barua, and P. Sarkar. Pairing-based cryptographic protocols: A survey. Cryptology ePrint Archive, Report 2004/064, 2004. `http://eprint.iacr.org/`.

[22] F. D. Garcia and P. van Rossum. Sound computational interpretation of symbolic hashes in the standard model. In *Advances in Information and Computer Security. Proc. 1st International Workshop on Security (IWSEC'06)*, volume 4266 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2006.

[23] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proc. 14th Symposium on Theory of Computing (STOC'82)*. ACM, 1982.

[24] P. Gupta and V. Shmatikov. Towards computationally sound symbolic analysis of key exchange protocols. In *Proc. 3rd Workshop on Formal Methods in Security Engineering: From Specifications to Code (FMSE'05)*, pages 23–32. ACM, 2005.

[25] J. Herzog. *Computational soundness for standard assumptions of formal cryptography.* PhD thesis, MIT, 2004.

[26] O. Horvitz and V. D. Gligor. Weak key authenticity and the computational completeness of formal encryption. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2003.

[27] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proc. 30th IEEE Symposium on Foundations of Computer Science (FOCS'89)*, pages 248–253, 1989.

[28] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185. Springer, 2005.

[29] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Proc. 4th International Symposium on Algorithmic Number Theory (ANTS-IV)*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[30] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003.

[31] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In *Proc. 6th International Conference on Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2004.

[32] D. Micciancio and S. Panjwani. Adaptive security of symbolic encryption. In *Proc. Theory of Cryptography Conference (TCC'05)*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2005.

[33] D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 2004.

[34] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. Theory of Cryptography Conference (TCC'04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[35] A. Roy, A. Datta, A. Derek, and J. C. Mitchell. Inductive trace properties for computational security. In *Proc. 7th Workshop on Issues in the Theory of Security (WITS'07)*, 2007.

[36] A. Roy, A. Datta, and J. C. Mitchell. Formal proofs of cryptographic security of Diffie-Hellman based protocols. In *Proceedings of the 3rd Symposium on Trustworthy Global Computing (TGC'07)*, volume 4912 of *Lecture Notes in Computer Science*, pages 312–329. Springer, 2008.

[37] K. Shim. Cryptanalysis of Al-Riyami-Paterson's authenticated three party key agreement protocols. Technical Report 2003/122, Cryptology ePrint Archive, 2003. `http://eprint.iacr.org/`.