Computationally Sound Verification of Security Protocols Using Diffie-Hellman Exponentiation

Yassine Lakhnech and Laurent Mazaré
{yassine.lakhnech,laurent.mazare}@imag.fr

VERIMAG - 2, av. de Vignates, 38610 Gières - FRANCE

Abstract. Recently, it has been proved that computational security can be automatically verified using the Dolev-Yao abstraction. We extend these results by adding a widely used component for cryptographic protocols: Diffie-Hellman exponentiation. Thus our main result is: if the Decisional Diffie-Hellman assumption is verified and the cryptographic primitives used to implement the protocol are secure, then safety in the symbolic world implies safety in the computational world. Therefore, it is possible to prove automatically safety in the computational world.
Keywords: Cryptographic Protocols, Diffie-Hellman, Soundness of Formal Encryption, Probabilistic Encryption.

1 Introduction

Historically, verification of cryptographic protocols has been separated in two distinct branches. *Symbolic verification* of cryptographic protocols, originates from the work of Dolev and Yao [DY83]. The essential part of this approach is the perfect cryptography assumption that can be roughly summarized as follows: messages are represented as algebraic terms, it is impossible to decode an encrypted message without the inverse key, fresh nonce creation is perfect, that is, nonces range over an infinite domain and freshness is absolute, the same holds for key creation. In the *computational approach*, cryptographic primitives operate on strings of bits and their security is defined in terms of high complexity and weak probability of success [GM84,BKR94] of any attacker. Protocols as well as attackers are randomized polynomial-time Turing machines. This computational approach is recognized as more realistic than the symbolic approach, however, its complexity makes it very difficult to design automatic verification tools.

There has been a recent trend in proving that the symbolic model is a sound abstraction of the computational model. Soundness means that a computational attack that has non-negligible probability can be mapped to a symbolic attack. In other words, the non-existence of a symbolic attack implies that any computational attack has a negligible probability. Obviously such soundness result cannot be established without any assumption on the cryptographic primitives. Therefore, the seeked results are of the form: if a protocol is secure in the symbolic model and the cryptographic primitives used to implement it verify some given computational security properties, then this protocol is secure in the computational model. The quest for this kind of results has probably been initiated by the work of Abadi and Rogaway [AR00]. This work essentially shows that, if the underlying symmetric encryption scheme satisfies some computational conditions then symbolic indistinguishability implies computational indistinguishability. In this work passive adversaries, that do not interact with the security protocol, are considered. Soundness of the symbolic model has soon been generalized to active adversaries by Backes et al. in [BPW03a]. The same authors also extended the soundness result to a wide class of cryptographic primitives such as digital signature or symmetric encryption [BPW03b]. Also [MW04,Lau04,JLM05b] relate a symbolic model to the computational although a different one.

Although these results encompass a large number of protocols, they do not apply to protocols that include **Diffie-Hellman** key exchange schema as SSH and TLS [DA99]. On the other hand, recently, symbolic verification of protocols within the symbolic model has been extended to protocols with Diffie-Hellman exponentiation showing that the existence of attacks is an NP-complete problem [CKRT03,MS03]. Moreover, in the computational world, efforts have been made to extend the classical Diffie-Hellman scheme [DH76] in order to design more general protocols [BCP02,BDZ03].

Therefore soundness of symbolic verification when considering Diffie-Hellman exponentiation is an interesting and challenging problem. To our knowledge, there is hardly any work on the soundness of the symbolic model in presence of Diffie-Hellman exponentiation. A notable exception is the work by Jonathan Herzog in [Her03,Her04] where he provides a symbolic model and shows that any attack in this model can be mapped to an attack in the computational model.

In this paper, we provide a symbolic model close to the Dolev-Yao model, that deals with protocols using Diffie-Hellman exponentiation as well as symmetric encryption. We prove that this symbolic model is a sound abstraction of the computational one in the sense explained above. Our result applies to protocols that use products in exponents and Diffie-Hellman values, that is exponentiations, as symmetric keys. In this paper, we only consider symmetric keys but extension to other primitives (e.g. [JLM05b]) such as signature, asymmetric encryption and hashing is straightforward. To prove our result, we introduce new security criteria inspired from the Decisional Diffie-Hellman assumption, chosen-plaintext security and selective forgery. These criteria are of interest on their own, especially the Dynamic Decisional Diffie-Hellman assumption.

Outline of this paper. The next section introduces variations of the classical Diffie-Hellman problem. Then section 3 considers a security criterion that combines modular exponentiation as in Diffie-Hellman and classical criteria for encryption scheme. Section 4 introduces cryptographic protocols with modular exponentiation. Computational and symbolic semantics are given as well as adversary models. This allows us to prove soundness of the symbolic adversary model in section 5. Finally, a conclusion of this paper is drawn.

2 The Diffie-Hellman Problem

For the remainder of this document, let η be the security parameter. Let $\mathbb{G}$ be a cyclic group of prime order q and let g be a generator of $\mathbb{G}$. q is assumed large, i.e. its number of digits is linear in η. We suppose that everyone knows g, $\mathbb{G}$ and q.

An adversary is a random Turing machine which execution time is polynomially bounded in η. An adversary tries to solve some challenge related to a security criterion. The security criterion is *verified* if for any adversary $\mathcal{A}$, its advantage is negligible. Negligible means that for any natural c, there exists η_0 such that for any $\eta > \eta_0$, $|\mathbf{Adv}(\mathcal{A})| \leq \eta^{-c}$.

To illustrate this notion of advantage, let us consider the simplest form of the Diffie-Hellman scheme. Two agents A and B want to create a shared secret value. A randomly chooses an element x in $[1, q]$ and sends g^x to B. B also chooses an element y in $[1, q]$ and sends g^y to A. Then A and B can both compute the shared value g^{xy}. However, it should be hard for any adversary to compute g^{xy} from g^x and g^y.

Formally, the *Computational Diffie-Hellman* (CDH) assumption is that for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined thereafter is negligible.

$$\mathbf{Adv}^{CDH}(\mathcal{A}) = pr\Big(\mathcal{A}(g^x, g^y) \to g^{xy} \Big| x, y \overset{R}{\leftarrow} [1, q]\Big)$$

However, this computational assumption does not immediately guarantee any secrecy property on g^{xy}. It may be feasible to compute the first bits of g^{xy} but infeasible to compute its whole representation. Thus, there exists a stronger assumption: from g^x and g^y, it is impossible to get any information on the shared secret g^{xy}.

The *Decisional Diffie-Hellman* (DDH) assumption is that for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined thereafter is negligible.

$$\mathbf{Adv}^{DDH}(\mathcal{A}) = pr\Big(\mathcal{A}(g^x, g^y, g^{xy}) \to 1 \Big| x, y \overset{R}{\leftarrow} [1, q]\Big) - pr\Big(\mathcal{A}(g^x, g^y, g^r) \to 1 \Big| x, y, r \overset{R}{\leftarrow} [1, q]\Big)$$

If this assumption holds, an adversary is not able to distinguish the shared secret from a random information with non-negligible probability.

The Diffie-Hellman assumption has been generalized in different ways: by authorizing more than two agents [BCP02] or specifying different related challenges [BDZ03]. Here, we introduce a dynamic version that is more general than the group version. The idea is that there are an unbounded number of challenges x_i. The adversary can ask for the exponentiation of any product of x_i and has to answer an exponentiation that it did not ask before. For example, the adversary can ask first for $g^{x_1 x_2}$ then for g^{x_2} then it may solve the challenge by outputting g^{x_1}. To ask for exponentiations, the adversary gives a finite list of integers (with no repetition) to an oracle and receives the exponentiation of the product of x_i which index appears in the list.

Let n be an integer greater or equal to 1, n is the bound on the request size. Like before, the computational version is more simple than the decisional one. The *Dynamic Computational Diffie-Hellman* (DCDH_n) assumption is a generalization of CDH. There are an unbounded number of challenges x_i which are random numbers. The adversary $\mathcal{A}$ has access to an oracle F. This oracle takes as argument a finite sub-set E of $\mathbb{N}$ (which size is lower than n) and returns $g^{\prod_{i \in E} x_i}$ (as soon as there are no possible confusion on the x_i, $g^{\prod_{i \in E} x_i}$ is denoted by g^E). At the end, $\mathcal{A}$ returns an element v of $\mathbb{G}$ and another finite sub-set E' of $\mathbb{N}$ (which size is also lower than n). and $\mathcal{A}$ wins its challenge iff E' has not been submitted to oracle F and $v = g^{E'}$. The advantage of $\mathcal{A}$ is the probability that it wins its challenge.

$$\mathbf{Adv}(\mathcal{A}) = pr\Big(\mathcal{A}/F \rightarrow (g^{E'}, E')\Big| x_i \stackrel{R}{\leftarrow} [1, q]\Big)$$

The DDH assumption is strong enough to imply this dynamic assumption. Note that it is not clear whether CDH implies DCDH_n.

Proposition 1. *If the DDH assumption is verified, then the $DCDH_n$ assumption is also verified.*

Proof. See appendix A.

A decisional version of DCDH_n is useful to prove our main results. The *Dynamic Decisional Diffie-Hellman* (DDDH_n) assumption is the decisional version of DCDH_n. A bit b is generated and the adversary tries to guess its value. The oracle F contains the previous oracle that computes g^E from E (these are called *standard requests*). The adversary can also ask for *challenge requests*: it submits a finite sub-set E' (size lower than n) of $\mathbb{N}$ and receives $g^{E'}$ if $b = 1$ and g^r for some random r otherwise. This time, the restriction is that any sub-set can be submitted only once to F. The advantage of $\mathcal{A}$ is given by:

$$\mathbf{Adv}(\mathcal{A}) = pr\big(\mathcal{A}/F \rightarrow 1 | b = 1\big) - pr\big(\mathcal{A}/F \rightarrow 1 | b = 0\big)$$

For both dynamic assumptions, an important point is that as the adversary has a bounded execution time, it can only access a finite number of x_i. Hence probabilities are still defined on finite domains.

Proposition 2. *If the DDH assumption is verified, then the $DDDH_n$ assumption is also verified. Reciprocally, if the $DDDH_n$ assumption is verified then so is DDH for $n \geq 2$.*

Proof. See appendix A.

Let rs be an integer. For the rest of the document, DDDH is used instead of DDDH_{rs} and any request related to an exponentiation only accepts argument which size is lower than rs.

A straightforward extension would be to allow lists with repetition as oracle's argument. However, as noted in [BDZ03], the equivalence between DDH and such extension is a difficult yet unsolved problem. Moreover, this restriction is not really relevant when considering protocols.

3 Melting SYM-CPA and DDH

A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is defined by three algorithms. The key generation algorithm $\mathcal{KG}$ is a randomized function which given a security parameter η outputs a key

k. The encryption algorithm $\mathcal{E}$ is also a randomized function which given a message and a key outputs the encryption of the message by this key. Finally the decryption algorithm $\mathcal{D}$ takes as input a key and a cypher-text and outputs the corresponding plain-text, i.e., $\mathcal{D}(\mathcal{E}(m,k),k) = m$. The execution time of the three algorithms is assumed polynomially bounded by η. Moreover, we ask that if r is randomly sampled from $[1, q]$, then the key generated by $\mathcal{KG}$ using g^r as random coins has the same distribution as keys generated by $\mathcal{KG}$ using classical random coins.

Security criteria are introduced using the game formalism given in [JLM05b]. A security game is defined as an experiment involving an adversary. The experiment proceeds as follows. First some parameters θ are generated randomly. The adversary is executed and can use an oracle F which depends on θ. At the end, the adversary has to answer a string of bits which is verified by an algorithm V which also uses θ (e.g. θ includes a bit b and the adversary has to output the value of b).

3.1 Security Game

A game γ is a triple $(\Theta; F; V)$ where

- Θ is a PRTM (polynomial random Turing machine) that randomly generates some challenge θ (for example, a bit b and a pair of key (pk, sk)).
- F is a PRTM that takes as arguments a string of bits s and a challenge θ and outputs a new string of bits. F represents the oracles that an adversary can call to solve its challenge. A/F denotes the execution of adversary $\mathcal{A}$ that may call oracle F using $\underline{F}$ in its code.
- V is a PRTM that takes as arguments a string of bits s and a challenge θ and outputs either true or false. It represents the verification made on the result computed by the adversary. The answer true (resp. false) means that the adversary solved (resp. did not solve) the challenge.

Note that Θ can generate an arbitrary number of parameters and F can represent an arbitrary number of oracles. Thus, it is possible to define games with multiple Θ and F. As soon as there is no risk for comprehension, we use the same notation for the challenge generator Θ and the generated challenge θ (both are denoted using θ).

The advantage of an adversary $\mathcal{A}$ against γ is

$$\mathbf{Adv}_{\mathcal{A}}^{\gamma}(\eta) = 2.\big(pr(\mathbf{Exp}_{\mathcal{A}}^{\gamma}(\eta) = true) - PrRand^{\gamma}\big)$$

Where **Exp** is the Turing machine defined by:

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\gamma}(\eta)$:
$\quad \theta \leftarrow \Theta(\eta)$
$\quad d \leftarrow \mathcal{A}/\eta, \lambda s.F(s,\theta)$
$\quad$ **return** $V(d,\theta)$

And $PrRand^{\gamma}$ is the best probability to solve the challenge that an adversary can have without using oracle F. Formally, $PrRand^{\gamma}$ is the maximum of $pr(\mathbf{Exp}_{\mathcal{A}}^{\gamma'}(\eta) = true)$ where $\mathcal{A}$ ranges over any possible PRTM and γ' is $(\Theta; 0; V)$ (0 is an oracle that always answer the same result, 0).

A game γ is said *safe* if for any PRTM $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\gamma}(\eta)$ is a negligible function in η. Properties and a reduction theorem for games appear in [JLM05b].

3.2 Patterns

Patterns are first order terms which extend bit-strings with pattern variables. These variables represent the different challenge secret information and are denoted by $[k_i]$ for keys (this asks the oracle to replace the pattern variable by the value of symmetric key k_i) and $[N_i]$ for nonces used. Variables can be used as atomic messages (data pattern) or at a key position (key pattern). When a left-right oracle is given a pattern term, it replaces pattern variables using the corresponding values and encodes the so-obtained message. More formally, patterns are given by the following grammar where bs is a bit-string and i is an integer.

$$\begin{aligned}
pat &::= \langle pat, pat\rangle \mid bs \mid [N_i] \mid [k_i] \mid \{pat\}_{key} \mid exp(prod) \\
key &::= bs \mid exp(prod) \mid [k_i] \\
prod &::= bs \mid [N_i] \mid prod \cdot prod
\end{aligned}$$

This grammar defines general patterns. Patterns that only use symmetric encryption as cryptographic primitive are called *symmetric patterns*.

The computation (valuation) of a pattern is easily defined recursively in a context θ associating bit-string values to the different variables. θ associates to each integer i a symmetric key $\theta_k(i)$ and a bit-string $\theta_N(i)$. The valuation produces a bit-string and it uses the symmetric encryption algorithm $\mathcal{E}$, the concatenation denoted by the operator $\cdot$, the exponentiation algorithm Exp (from $\mathbb{G} \times \mathbb{N}$ to $\mathbb{G}$) and the product algorithm $Prod$ (from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$).

$$\begin{aligned}
v(bs, \theta) &= bs \\
v([k_i], \theta) &= \theta_k(i) \\
v([N_i], \theta) &= \theta_N(i) \\
v(\langle p_1, p_2\rangle, \theta) &= v(p_1, \theta).v(p_2, \theta)
\end{aligned}$$

$$\begin{aligned}
v(\{p\}_k, \theta) &= \mathcal{E}(v(p, \theta), v(k, \theta)) \\
v(exp(p), \theta) &= Exp(g, v(p, \theta)) \\
v(p_1 \cdot p_2, \theta) &= Prod(v(p_1, \theta), v(p_2, \theta))
\end{aligned}$$

3.3 *N*-SYM-CPA

The N-SYM-CPA criterion has been introduced formally in [JLM05b]. It includes both aspects indistinguishability and authentication that are present in asymmetric encryption and digital signature respectively. Therefore, our criterion for symmetric encryption is in a combination of IND-CPA and selective forgery. The case $N = 1$ is similar to the IND-CPA $\wedge$ INT-CTXT criterion described in [BN00]. However, we reformulate this in our formalism in order to add the Diffie-Hellman part.

The N-SYM-CPA criterion is $\gamma_N = (\Theta; F; V)$ where Θ generates N symmetric keys and a bit b; F gives access to one oracle for each key: a left-right encryption oracle that takes as argument a pair of symmetric patterns $\langle pat_0, pat_1\rangle$ and outputs pat_b completed with the secret keys ($v(pat_b, \theta)$) and encoded with k_i. There is an acyclicity hypothesis regarding keys: the encryption oracle related to key i works only on pair of symmetric patterns $\langle pat_0, pat_1\rangle$ such that for any j in $var(\langle pat_0, pat_1\rangle)$, $i < j$.

Finally, V is composed of two parts: V_{IND} returns true when the adversary returns bit b; V_{UNF} returns true when the adversary outputs a message encoded by one of the symmetric key and this message has not been produced by an encryption oracle. Then V is satisfied if V_{IND} or V_{UNF} is satisfied. We require that there is no string that satisfies both V_{IND} and V_{UNF} (this can be done by asking the name of the challenge together with its solution to the adversary). The criterion related to IND $(\Theta; F; V_{IND})$ (resp. to UNF $(\Theta; F; V_{UNF})$) is denoted by N-SYM-CPA/IND (resp. N-SYM-CPA/UNF).

A symmetric encryption scheme $\mathcal{SE}$ is said N-SYM-CPA iff for any adversary $\mathcal{A}$ in $PRTM$, $\mathbf{Adv}^{\gamma_N}_{\mathcal{SE},\mathcal{A}}(\eta)$ is negligible. There exist some algorithms strongly believed to be N-SYM-CPA.

Note that left-right oracle can be used with $\langle m_1, m_2\rangle$ where m_1 and m_2 have different sizes. This aspect is discussed in [AR00].

3.4 *N*-DH-SYM-CPA

The N-DH-SYM-CPA security criterion is an extension of N-SYM-CPA to general patterns.

The N-DH-SYM-CPA criterion is $\gamma_N = (\Theta; F; V)$ where Θ randomly generates N symmetric keys, N nonces N_1 to N_N and a bit b; F gives access to one oracle for each key: a left-right encryption oracle that takes as argument a pair of patterns $\langle pat_0, pat_1\rangle$ and outputs pat_b completed with the secret keys ($v(pat_b, \theta)$) and encoded with k_i. Moreover if pat_0 or pat_1 contains a $[N_i]$ and this is replaced by the value defined in θ.

An other oracle is related to the Diffie-Hellman part. The adversary can submit a finite subset E of $\mathbb{N}$ and receives g^E (i.e. $g^{\prod_{i \in E} \theta_k(i)}$).

Finally, the last oracle takes as argument a finite subset of E and a pair of patterns $\langle pat_0, pat_1 \rangle$. It outputs pattern pat_b encoded using key k_E. k_E is produced by $\mathcal{KG}$ using g^E as random coins, thus it is specific to a given E.

There are a few restrictions on how the oracles may be called: g^E can be asked iff no left-right encryption using k_E has been asked. There is also an acyclicity hypothesis: there exists a total order among nonces and keys denoted by $\cdot > \cdot$. If $x > k$ then x cannot be asked in a pattern submitted to the oracle related to key k. If $x > N_i$ then x cannot be asked in a pattern given to an oracle related to key k_E where $i \in E$.

Finally, V is composed of two parts: V_{IND} returns true when the adversary returns bit b; V_{UNF} returns true when the adversary outputs a message encoded by one of the symmetric key and this message has not been produced by an encryption oracle. Then V is satisfied if V_{IND} or V_{UNF} is satisfied. We require that there is no string that satisfies both V_{IND} and V_{UNF}. The criterion related to IND ($\Theta; F; V_{IND}$) (resp. to UNF ($\Theta; F; V_{UNF}$)) is denoted by N-DH-SYM-CPA/IND (resp. N-DH-SYM-CPA/UNF).

A symmetric encryption scheme $\mathcal{SE}$ is said N-DH-SYM-CPA iff for any adversary $\mathcal{A}$, $\mathbf{Adv}^{\gamma_N}_{\mathcal{SE},\mathcal{A}}(\eta)$ is negligible.

The main result concerning this new criterion is that it is equivalent to 1-SYM-CPA if the Decisional Diffie-Hellman assumption holds.

Proposition 3. *If the symmetric encryption scheme $\mathcal{SE}$ is 1-SYM-CPA and DDH is verified, then $\mathcal{SE}$ is also N-DH-SYM-CPA for any integer N.*

Proof. See appendix B

This last proposition links N-DH-SYM-CPA to standard notions in the computational world. Moreover, the proof of our main theorem is greatly simplified by assuming that the encryption scheme used in the implementation is N-DH-SYM-CPA.

For simplicity's sake, we only considered a bounded number of challenges. However, using the technique presented in [JLM05b], one can easily extend this result to a number of challenges that is polynomial in η. As our protocols only use a fixed number of challenges, this is not necessary in this document.

4 Protocol Models

4.1 Messages and Deduction

The symbolic model is an idealized representation of cryptographic protocols. The main abstractions are that nonces values cannot be predicted and that it is impossible to decrypt an encoded message without knowing the inverse key. In this model, messages are represented by first-order terms. Let Ag, No, Ne and Keys be disjoint countable sets of, respectively, identities, nonces, exp-nonces and key names. Let A, N, N_e and key be meta-variables over these sets.

$$\begin{aligned} msg &::= N_e \mid N \mid A \mid \{msg\}_{key} \mid \langle msg, msg \rangle \mid exp(prod) \\ key &::= K \mid exp(prod) \\ prod &::= N_e \mid prod \cdot prod \end{aligned}$$

The function symbols exp and $[\cdot]$ represents modular exponentiation and product. In particular, $[\cdot]$ is considered associative and commutative.

We consider the classical Dolev-Yao [DY83] adversary and let $E \vdash m$ denote that message m is deducible from the set of messages E. This entailment relation is extended to the *exp* operator by adding the two following rules:

$$\frac{E \vdash N_e}{E \vdash exp(N_e)} \qquad\qquad \frac{E \vdash N_e \qquad E \vdash exp(p)}{E \vdash exp(p \cdot N_e)}$$

These new rules seem quite natural in the computational world: it is possible from a value x to compute g^x and from x and g^y to compute g^{xy}.

The main result of this paper is that the Dolev-Yao model extended with these rules yields a sound symbolic model meaning that any computational attack with non-negligible probability corresponds to a symbolic attack.

4.2 Description of Cryptographic Protocols and Semantics

For the sake of presentation, we consider protocols that only involve a single role. Moreover, this role is only instantiated in one session. This is done without loss of generality when a bounded number of sessions is considered. Indeed, each interleaving of the actions of the different participants can be seen as a role and the different interleavings correspond to different protocols.

Thus, a protocol is described by a list of actions which are either emission $!m$ or reception $?m$ of a message m. To make protocols readable, we use the usual BAN syntax. For example, a version of Diffie-Hellman protocol between two roles A and B is:

$$\begin{aligned} &A \to B : exp(N_A) \\ &B \to A : exp(N_B) \\ &A \to B : \{A, B\}_{exp(N_A \cdot N_B)} \end{aligned}$$

The session that involves agents A and B is represented by the simplified protocol:

$$!exp(N_A) \quad ?exp(x) \quad !exp(N_B) \quad ?exp(y) \quad !\{A, B\}_{exp(N_A \cdot y)} \quad ?\{A, B\}_{exp(x \cdot N_B)}$$

We consider the classical adversary model where the adversary controls the network, receives all the outputs ($!m$) and submits some forged message to the inputs ($?m$).

Henceforth, let us consider an arbitrary fixed protocol $\ddagger_1 t_1 ... \ddagger_k t_k$, where $\ddagger_i$ is either "!" or "?" and t_i is a term. There are two different execution models, one for the symbolic setting and one for the computational setting producing a symbolic and a computational trace, respectively. A *symbolic action sequence* is a list of actions $s\ m$ where s is either ? or ! and m is a ground (closed) message. A *symbolic trace* is a symbolic action sequence $\ddagger_1 m_1 ... \ddagger'_k m'_k$ with $k' \leq k$ that satisfies the following conditions:

1. There exists a ground substitution σ such that for any i, $t_i\sigma = m_i$;
2. For any i, if $\ddagger_i$ is "?", then m_i is deducible from the previous messages m_1 to m_{i-1} and the initial knowledge of the adversary E_0,i.e.,

   $$E_0, m_1, ..., m_{i-1} \vdash m_i.$$

   The set E_0 contains the atomic messages of the m_i's that do not appear in any t_i, i.e. $E_0 = \bigcup_i atoms(m_i) \setminus \bigcup_i atoms(t_i)$.

The set $trace(\Pi)$ contains the possible traces for protocol Π. Our symbolic semantics is essentially the one used in [CKRT03], where it is shown that secrecy is an co-NP-complete problem.

A *computational action sequence* is a list of actions $\ddagger\, bs$ where bs is a bit-string and $\ddagger$ is either "?" or "!". A *computational trace* is the result of the interaction of an adversary $\mathcal{A}$, which is a polynomial random Turing machine, and the protocol. This interaction is defined using the Turing machine $Exec(\mathcal{A}, \Pi)$. Since we are interested in relating the symbolic and computational semantics we define $Exec$ in such way that along the computational trace it outputs a corresponding symbolic action sequence. We then show that the symbolic action sequence is a trace except for negligible probability. The reader should be convinced that producing the symbolic action sequence by no means interferes with the computational semantics.

To simplify the presentation of the *Exec* algorithm, we only give pseudo-code using the following functions:

- $init(\Pi)$ generates the keys, nonces and exp-nonces that are chosen by the protocol Π, i.e., those in $atoms(\Pi)$, and not by the adversary. It returns a substitution θ associating bit-string values to these elements.
- $parse(bs, t, \theta, \sigma)$ parses the bit-string bs using prototype t and knowledge from θ, it returns the updated version of θ as well as an updated symbolic substitution σ.
- $concr(m, \theta)$ concretizes message m using knowledge from θ and returns the corresponding bit-string.
- $compl(\sigma)$ completes the symbolic substitution σ by associating remaining free variables to a distinct fresh nonces.

The *Exec* algorithm uses two substitutions: the symbolic substitution σ that links protocol variables to messages and the computational substitution that links variables to strings of bits. Notice that the adversary can decide to stop interacting with the protocol by providing an answer other than an updated memory *mem* and a bit string *bs* when an action $?t$ is to be executed.

```
Algorithm  Exec(𝒜, ‡₁n₁...‡ₖnₖ):
    θ ← init(‡₁n₁...‡ₖnₖ)
    mem ← []
    for i in [1, k] do
        if ‡ᵢ =! then
            bs ← concr(nᵢ, θ)
            mem ← 𝒜(bs, mem)
            t_c ← append(‡ᵢbs, t_c)
        else
            X ← 𝒜(mem)
            if X = bs, mem then
                σ, θ ← parse(bs, mᵢ, θ, σ)
                t_c ← append(‡ᵢbs, t_c)
            else
                goto done
    done
    σ ← compl(σ)
    return (‡₁m₁...‡ᵢmᵢ₋₁)σ, t_c
```

The next proposition relates precisely the computational trace and symbolic action sequence that *Exec* outputs. A computational trace t_c is a *possible concretization* of a symbolic action sequence t_f if there exists a computational substitution θ such that one of the possible valuation of t_f using θ is t_c.

Proposition 4. *Let $\mathcal{A}$ be an adversary and Π a protocol. If $Exec(\mathcal{A}, \Pi)$ outputs t_f, t_c, then t_c is a possible concretization of t_f.*

5 Soundness of the Symbolic Model

In this section, we show that the symbolic sequence action produced by $Exec(\mathcal{A}, \Pi)$ is a symbolic trace except for negligible probability. Together with Proposition 4, this implies that only computational traces with negligible probability might not correspond to symbolic traces.

To do so, it is important to characterize when a symbolic action sequence is *not* a symbolic trace. Let E be a set of messages. Then, let $keys(E)$ denote the set of keys (including exponentiations) that are deducible from E. Moreover, let $dec(E, K)$ denote the set of messages deducible from messages in E using unpairing and decompositions with keys in K. Then, we have the following:

Proposition 5. *Let E be a set of messages and m be a message. If $E \nvdash m$ then, one of the following holds:*

- *There exists a message $\{n\}_{key}$ in $dec(m, keys(E))$ such that key is not in $keys(E)$ and $\{n\}_{key}$ is not in $dec(E, keys(E))$. This corresponds to the case where the adversary forges $\{n\}_{key}$.*
- *There exists a key k or a nonce N or an exp-nonce N_e or an exponentiation $exp(p)$ in $dec(m, keys(E))$ that is not in $dec(E, keys(E))$. This corresponds to the case where the adversary guesses a secret value or breaks an encryption.*

There are a few restrictions over protocols Π considered. These restrictions are defined in the symbolic world (as they are easier to check with automated tools).

1. Keys and exp-nonces that are not chosen by the adversary remain secret throughout the protocol execution. Moreover, exponentiations that are used as keys also remain secret. Let S be the set of such exponentiations, keys and exp-nonces.
2. There exists an order among exp-nonces and keys from S such that if $u < v$ then for any symbolic trace, v cannot appear encoded by a key using u (the key is exactly u if u is a key, if it is an exp-nonce, the key is an exponentiation using u). This is the usual acyclicity condition on keys.
3. No execution can lead to send $exp(N_e \cdot N_e \cdot ...)$ where N_e is an exp-nonce of S.
4. Whenever a reception of $exp(x)$ occurs, either x is known by the honest parties or $exp(x)$ is signed by a key from S.

Among these conditions the last one seems the most restrictive. Let us discuss it. In general, the Dolev-Yao abstraction is not a sound abstraction of the computational model. To illustrate this, let us consider the protocol $?exp(x)\ !exp(x \cdot N)$ and the group $\mathbb{G}$ of quadratic residues over $\mathbb{Z}/m^2$ (as introduced in [BCP03]). It is clear that a symbolic adversary cannot deduce N. A computational adversary, however, can submit $1 + m$ for $exp(x)$. Then it receives $(1 + m)^N \mod m^2$ which is equal to $(1 + N.m) \mod m^2$. Therefore, the adversary can deduce information on the value of N (i.e. the value of $N \mod m$). That should not be possible as the DDH assumption is classically assumed true on $\mathbb{G}$.

There are at least two other ways to solve this problem:

- A solution is to strengthen the DDH assumption. Parameter g can also be chosen by the adversary. This assumption is less classical. It does not hold for quadratic residue.
- Else, we could put restrictions on protocols and adversaries. For example, there are no problems for a passive adversary.

Our restriction given as hypothesis 4 is a more subtle restriction on protocols than the passive case. It is quite fair as Diffie-Hellman does not provide any authentification. In particular, this restriction is true for the simplified version of TLS that appears in the introduction of [Her03].

The next theorem is our main result : traces produced by a computational adversary can be abstracted to symbolic traces with overwhelming probability. Let Π be a protocol and $\mathcal{SE}$ be the encryption scheme used to implement *Exec*.

Theorem 1. *If the DDH assumption holds and $\mathcal{SE}$ is SYM-CPA then for any concrete adversary $\mathcal{A}$:*

$$pr\big(t_f, t_c \leftarrow Exec(\mathcal{A}, \Pi) \text{ and } t_f \notin traces(\Pi)\big) \text{ is negligible}$$

Proof. Let us consider an adversary $\mathcal{A}$ such that the probability to create a symbolic trace that is not in $traces(\Pi)$ is not negligible. Then it is possible to use $\mathcal{A}$ and a modified version of the *Exec* algorithm in order to gain a non negligible advantage against N-DH-SYM-CPA. This is detailed in appendix C.

Using this main theorem, it is possible to relate properties in the symbolic world to properties in the computational world. These properties can be verified in the symbolic world and then hold in the computational world. This has been done for some trace properties in [JLM05a,MW04] such as authentication and a weak version of secrecy. But also a stronger version of secrecy, called SecNonce, can be verified [CW05] except for exp-nonces. The SecNonce property can be defined as a game that goes as follows: two nonces N_0 and N_1 are chosen randomly. The secrecy of a nonce name N means that an adversary cannot distinguish between the protocol executed with N_0 and N_1 taken as values for N. The SecNonce property for exp-nonces is not correctly abstracted by non-deductibility in the symbolic model. Indeed, if we consider the protocol $!exp(N)$, then if an adversary is given two exp-nonce values N_0 and N_1, it can easily guess which value was used in the protocol (as exponentiation is deterministic). Hence the SecNonce property can only be used to abstract secrecy of nonces (as they do not appear in exponentiations).

6 Conclusion and Future Works

We prove soundness of a symbolic model that deals with Diffie-Hellman exponentiation. Although we only considered Diffie-Hellman exponentiation and symmetric encryption, adding other primitives such as asymmetric encryption, hashing or digital signature should not be complicated. In particular, [JLM05b] uses a specific technique to combine some security primitives and their related criteria.

As future work, we plan to investigate automatic verification of the protocol restrictions in the symbolic world. With such verification, it would be possible to entirely verify a protocol with an automatic prover.

References

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.

[BCP02] E. Bresson, O. Chevassut, and D. Pointcheval. The group Diffie-Hellman problems. In K. Nyberg and H. Heys, editors, *Proc. of SAC'02*, volume 2595 of *LNCS*, pages 325–338, Saint-John's, Newfoundland, CA, August 2002. Springer-Verlag.

[BCP03] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In C. S. Laih, editor, *Proc. of Asiacrypt'03*, volume 2894 of *LNCS*, pages 37–54, Taipei, TW, November-December 2003. IACR, Springer-Verlag.

[BDZ03] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of diffie-hellman problem. In *ICICS*, pages 301–312, 2003.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Proc. CRYPTO 94*, pages 341–358. Springer, 1994. Lecture Notes in Computer Science No. 839.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer-Verlag, 2000.

[BPW03a] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 220–230. ACM Press, 2003.

[BPW03b] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Symmetric authentication within a simulatable cryptographic library. In Einar Snekkenes and Dieter Gollmann, editors, *Proceedings of the 8th European Symposium on Research in Computer Security, ESORICS'03*, volume 2808 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2003.

[CKRT03] 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*, Lecture Notes in Computer Science. Springer-Verlag, 2003. To appear.

[CW05] 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, Edinburgh, U.K, April 2005. Springer.

[DA99] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1. IETF RFC Publication, January 1999. Status: PROPOSED STANDARD.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DY83] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[Her03] Jonathan C. Herzog. The diffie-hellman key-agreement scheme in the strand-space model. In *CSFW*, pages 234–247. IEEE Computer Society, 2003.

[Her04] Jonathan Herzog. *Computational soundness for standard assumptions of formal cryptography*. PhD thesis, MIT, 2004.

[JLM05a] R. Janvier, Y. Lakhnech, and L. Mazare. 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, Edinburgh, U.K, April 2005. Springer.

[JLM05b] R. Janvier, Y. Lakhnech, and L. Mazaré. (De)Compositions of Cryptographic Schemes and their Applications to Protocols. Technical Report TR-2005-03, Verimag, Centre Équation, 38610 Gières, February 2005.

[Lau04] Peeter Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *IEEE Symposium on Security and Privacy*, pages 71–85, 2004.

[LM05] Y. Lakhnech and L. Mazaré. Computationally Sound Verification of Security Protocols Using Diffie-Hellman Exponentiation. Technical Report TR-2005-0X, Verimag, Centre Équation, 38610 Gières, March 2005.

[MS03] J. Millen and V. Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *16th IEEE Computer Security Foundations Workshop*, pages 47–61. IEEE Computer Society, 2003.

[MW04] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proceedings of the Theory of Cryptography Conference*, pages 133–151. Springer, 2004.

A Proofs for Propositions 1 and 2

Proposition 1: If the DDH assumption is verified, then the DCDH_n assumption is also verified.
Proposition 2: If the DDH assumption is verified, then the DDDH_n assumption is also verified. Reciprocally, if the DDDH_n assumption is verified then so is DDH for $n \geq 2$.

We first prove proposition 2. Proposition 1 is a direct consequence. The order $<$ used between pairs of naturals is the lexical order.

DDH $\Rightarrow$ *DDDH* : let $\mathcal{A}$ be an adversary against DDDH_n and P its polynomial bound. Let i and j be two integers in $[1, P(\eta)]$ such that $i < j$. We build some adversaries $\mathcal{B}_{i,j}$ against DDH using $\mathcal{A}$ and a modified version of the F oracle denoted by $F_{i,j}$.

Adversary $\mathcal{B}_{i,j}(X, Y, Z)$
$b \stackrel{R}{\leftarrow} \{0, 1\}$
$b' \leftarrow \mathcal{A}/F_{i,j}$
return $b = b'$

Oracle $F_{i,j}$ uses X, Y, Z and the random bit b to simulate oracle F. The application of $F_{i,j}$ to E is computed as follows.

- For any k in $\mathbb{N}$ different from i and j, x_k is randomly sampled in $[1, q]$ when necessary.
- For k and k' in $\mathbb{N}$, a value $x_{k,k'}$ is also randomly sampled in $[1, q]$ when necessary.

The application of $F_{i,j}$ to E returns g^r for some randomly sampled r if $b = 0$. Else if $b = 1$, it returns $f(g, E)$ which is recursively defined by:

- For the lowest pair (k, k') in E such that $(k, k') < (i, j)$, $f(g, E)$ returns $f(g, E \setminus \{k, k'\})^{x_{k,k'}}$.
- If i and j appear in E, then $f(g, E)$ returns $f(Z, E \setminus \{i, j\})$.
- If only i appears in E, then $f(g, E)$ returns $f(X, E \setminus \{i\})$.
- If only j appears in E, then $f(g, E)$ returns $f(Y, E \setminus \{j\})$.
- For any remaining k in E, $f(a, E)$ returns $f(a, E \setminus \{k\})^{x_k}$.
- $f(a, \emptyset)$ returns a.

Then the advantage of $B_{i,j}$ is defined by:

$$\mathbf{Adv}(\mathcal{B}_{i,j}) = pr(\mathcal{A}/F_{i,j} \text{ wins}|Z = g^{xy}) - pr(\mathcal{A}/F_{i,j} \text{ wins}|Z = g^r)$$

When $i = 1$ and $j = 2$ and $Z = g^{xy}$, the situation is the same as when $\mathcal{A}$ is confronted to F. Moreover, let (i', j') be the successor of (i, j) for the lexical order (i and j are bounded by $P(\eta)$), then the case $F_{i,j}$ when $Z = g^r$ is similar to the case (i', j') when $Z = g^{xy}$. By summing these advantages, we get:

$$\sum_{1 \leq i < j \leq P(\eta)} \mathbf{Adv}(\mathcal{B}_{i,j}) = pr(\mathcal{A}/F \text{ wins}) - pr(\mathcal{A}/F_{P(\eta)-1,P(\eta)} \text{ wins})$$

$$2\Big(\sum_{1 \leq i < j \leq P(\eta)} \mathbf{Adv}(\mathcal{B}_{i,j})\Big) = \mathbf{Adv}(\mathcal{A}) - \mathbf{Adv}(\mathcal{A}^o)$$

The first advantage is related to $\mathcal{A}$ against DDDH_n and the last probability is the advantage of a modified version of $\mathcal{A}$ against $\text{DDDH}_{\lfloor n/2 \rfloor + 1}$. Namely $\mathcal{A}^o$ simulates $\mathcal{A}$ confronted to oracle $F_{P(\eta)-1,P(\eta)}$. Hence $\mathcal{A}^o$ only makes requests of size lower than $\lfloor n/2 \rfloor + 1$ ($\lfloor x \rfloor$ denotes the integral value of x).

Now, we proceed by induction on n to prove our result. For $n = 1$, the requests can only have size one and ask for g^{x_i} for some i. The adversary has to distinguish between g^{x_i} and g^r without any other information on x_i. Therefore, the advantage of any adversary against DDDH_1 is 0.

Let us suppose that DDDH_i holds for any i lower than n. Then let $\mathcal{A}$ be an adversary against DDDH_n which execution is bounded by polynomial P. There exists an adversary $\mathcal{A}^o$ against $\mathrm{DDDH}_{\lfloor n/2 \rfloor+1}$ and $P(\eta)$ adversaries against DDH such that:

$$2\Big(\sum_{1\leq i<j\leq P(\eta)} \mathbf{Adv}(\mathcal{B}_{i,j})\Big) = \mathbf{Adv}(\mathcal{A}) - \mathbf{Adv}(\mathcal{A}^o)$$

As DDH and $\mathrm{DDDH}_{\lfloor n/2 \rfloor+1}$ hold $({}_{\lfloor n/2 \rfloor+1} < n)$, the advantage of $\mathcal{A}$ is negligible. The assumption DDDH_n hold. □

DDDH $\Rightarrow$ *DDH* : Let $\mathcal{A}$ be an adversary against DDH. $\mathcal{B}$ is an equivalent adversary against DDDH_n (for $n \geq 2$):

Adversary $\mathcal{B}$
$X \leftarrow \underline{F}(\text{standard}, \{1\})$
$Y \leftarrow \underline{F}(\text{standard}, \{2\})$
$Z \leftarrow \underline{F}(\text{challenge}, \{1,2\})$
$b \leftarrow \mathcal{A}(X, Y, Z)$
return b

The advantage of $\mathcal{A}$ and $\mathcal{B}$ are equals. As we assume DDDH, the advantage of $\mathcal{B}$ is negligible. Hence the advantage of $\mathcal{A}$ is also negligible. □

DDH $\Rightarrow$ *DCDH* : Let $\mathcal{A}$ be an adversary against DCDH_n. Then $\mathcal{B}$ is the adversary against DDDH_n defined by:

Adversary $\mathcal{B}$
$(v, E') \leftarrow \mathcal{A}/\underline{F}$
$v' \leftarrow \underline{F}(\text{challenge}, E')$
return $v = v'$

The advantage of $\mathcal{B}$ is:

$$\mathbf{Adv}(\mathcal{B}) = pr(\mathcal{A} \text{ wins}|b = 1) - pr(\mathcal{A} \text{ wins}|b = 0)$$

The first probability corresponds to the advantage of $\mathcal{A}$ against DCDH. The second one is the probability that $\mathcal{A}$ outputs g^r where r is randomly sampled from $[1, q]$ and $\mathcal{A}$ has no other information related to r. As q is large, this probability is negligible. Thus the advantage of $\mathcal{A}$ against DCDH is negligible. □

B Proof for Proposition 3

Proposition 3: If the symmetric encryption scheme $\mathcal{SE}$ is 1-SYM-CPA and DDH is verified, then $\mathcal{SE}$ is also n-DH-SYM-CPA for any integer n.

This proof uses the following proposition from [JLM05b]:

Proposition 6. *If the symmetric encryption scheme* $\mathcal{SE}$ *is* 1*-SYM-CPA, then* $\mathcal{SE}$ *is also* n*-DH-SYM-CPA for any integer* n*.*

For this proof, we introduced two new criteria: the first one n, m-DH-SYM-CPA is similar to n-DH-SYM-CPA where the bound on challenge keys is n and the bound on challenge nonces is m. Criterion n, m-DH-SYM-CPA' is the same criterion where the left-right encryption oracles can only be used with strings of bits instead of patterns.

We proceed in three steps. First we prove that an encryption scheme is n-DH-SYM-CPA if and only if it is $0, 2n$-DH-SYM-CPA. After that we prove that if the DDDH assumption holds and an encryption scheme is n-SYM-CPA, then it is also $0, n$-DH-SYM-CPA'. Finally, using our reduction technique, we prove that if an encryption scheme is $0, n$-DH-SYM-CPA', it is also $0, n$-DH-SYM-CPA. From there it is easy to conclude.

Lemma 1. *An encryption scheme is n-DH-SYM-CPA if and only if it is also $0, 2n$-DH-SYM-CPA for any n.*

Proof. It is possible to build an adversary $\mathcal{A}'$ against $0, 2n$-DH-SYM-CPA from any adversary $\mathcal{A}$ against n-DH-SYM-CPA such that the advantages of $\mathcal{A}$ and $\mathcal{A}'$ are the same. Adversary $\mathcal{A}'$ executes $\mathcal{A}$ as a sub-routine. Oracle calls from $\mathcal{A}$ are submitted to the oracle of $\mathcal{A}'$ except that whenever $\mathcal{A}$ issues a request related to a key k, $\mathcal{A}'$ replaces the reference to k by a reference to $i(k)$ where $i(k)$ is the index of a nonce that $\mathcal{A}$ does not use. Moreover, the acyclicity hypothesis is preserved. □

Lemma 2. *If $DDDH_n$ hold, an encryption is n-SYM-CPA implies that it is also $0, n$-DH-SYM-CPA' for any n.*

Proof. Here, we consider the case where patterns can only be strings of bits. Let $\mathcal{A}$ be an adversary against 0-DH-SYM-CPA'. Then we build an adversary $\mathcal{B}$ against DDDH$_n$ using $\mathcal{A}$. $\mathcal{A}$ is executed as a subroutine, its oracle is implemented by F that works as follows:

- When asked for exponentiation of E, $\mathcal{B}$ issues a *standard request* to its DDDH oracle with argument E.
- When asked for encryption related to E, $\mathcal{B}$ issues a *challenge request* to its DDDH oracle with argument E. Using the result, $\mathcal{B}$ computes the symmetric key k_E.

As each E may only be submitted once to the DDDH oracle, $\mathcal{B}$ has to store the answers of its oracles. $\mathcal{A}$ has two ways to win and $\mathcal{B}$ returns 1 iff $\mathcal{A}$ succeeds.

Adversary $\mathcal{B}$
    $b' \stackrel{R}{\leftarrow} \{0,1\}$
    $res \leftarrow \mathcal{A}/F$
    return $res = b'$ **or** res is a "fresh" encoding by k_E

The advantage of B is defined by:

$$\mathbf{Adv}(\mathcal{B}) = pr(\mathcal{A} \text{ wins}|b=1) - pr(\mathcal{A} \text{ wins}|b=0)$$

The second probability is related to the event "$\mathcal{A}$ wins" when $b = 0$. When b equals 0, keys are generated randomly. Hence this behavior can be simulated by an adversary $\mathcal{A}^o$ against n-SYM-CPA. $\mathcal{A}^o$ executes $\mathcal{A}$, it also generates the necessary values for the different x_k. When asked for a left-right encryption related to E, $\mathcal{A}^o$ uses the left-right encryption oracle related to a given key k. It also stores the association of E and k in order to use the right key for the next oracle calls. Hence, we get:

$$2.\mathbf{Adv}(\mathcal{B}) = \mathbf{Adv}(\mathcal{A}) - \mathbf{Adv}(\mathcal{A}^o)$$

The advantage of $\mathcal{B}$ is related to DDDH$_n$. The advantage of $\mathcal{A}$ is against 0-DH-SYM-CPA' whereas the advantage of $\mathcal{A}^o$ is against n-SYM-CPA. Hence the advantage of $\mathcal{A}$ is negligible.

□

Lemma 3. *If $DDDH_n$ hold. Then for any n an encryption scheme is $0, n$-DH-SYM-CPA' if and only if it is $0, n$-DH-SYM-CPA.*

Proof. This proof uses the reduction theorem from [JLM05b]. This theorem is stated in appendix E. Let us consider the $0, n$-DH-SYM-CPA criterion and the order between nonces: $N_1 < N_2 < N_n$. We first treat the case of the indistinguishability part of the criterion. A valid partition of this criterion is:

- θ_1 generates nonce N_1 and θ_2 generates nonces N_2 to N_n and the challenge bit b.
- Oracle F_1 can be cut in two layers G and H using for G the classical left-right oracle related to θ_1 and:

$$H(bs, \theta_2, \theta_2') = \langle F_2(bs, \theta_2), F_2(bs, \theta_2') \rangle$$

– The verification oracle only uses bit b and so θ_2.

Hence, applying the reduction theorem 2 gives us that for any adversary $\mathcal{A}$ there exists two adversaries $\mathcal{A}^o$ and $\mathcal{B}$ such that:

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)|$$

Criterion γ is $0, n$-DH-SYM-CPA, γ_1 is $0,1$-DH-SYM-CPA and γ_2 is $0,(n-1)$-DH-SYM-CPA. The acyclicity hypothesis implies that criterion $0,1$-DH-SYM-CPA is equivalent to $0,1$-DH-SYM-CPA'. Hence, using an easy recursion, we get that there exists some adversaries $\mathcal{B}_i$ verifying:

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.\sum_{i=1}^{n} |\mathbf{Adv}^{\gamma_1}_{\mathcal{B}_i}(\eta)|$$

Hence, if γ_1 is safe then γ is also safe.

For the UNF part of the criterion, the proof is more straightforward. Let $\mathcal{A}$ be an adversary against $0, n$-DH-SYM-CPA/UNF. The adversary $\mathcal{B}$ against DDDH$_n$ uses $\mathcal{A}$ as a sub-routine. At first, $\mathcal{B}$ randomly chooses a subset E_r of $[1, n]$. With non-negligible probability, E_r is related to the key that $\mathcal{A}$ finally attacks. After that, $\mathcal{B}$ executes $\mathcal{A}$ and simulates the necessary oracles using challenge request for g^{E_r} and standard requests for the other exponentiations. Finally, $\mathcal{B}$ returns one if $\mathcal{A}$ correctly solved its challenge. As usual, the advantage of $\mathcal{B}$ is

$$\mathbf{Adv}(\mathcal{B}) = pr(\mathcal{A} \text{ wins}|b = 1) - pr(\mathcal{A} \text{ wins}|b = 0)$$

The case $b = 1$ corresponds to a standard execution of $\mathcal{A}$ whereas in the case $b = 0$, $\mathcal{A}$ is confronted to a random key if it tries to attack E_r.

$$\mathbf{Adv}(\mathcal{A}) \leq \mathbf{Adv}(\mathcal{B}) + 2^n.\mathbf{Adv}(\mathcal{A}^o)$$

The advantage of $\mathcal{B}$ and $\mathcal{A}$ are respectively related to DDDH$_n$ and 1-SYM-CCA. Therefore the advantage of $\mathcal{A}$ is negligible. □

We have that DDH implies DDDH$_n$. Moreover 1-SYM-CPA implies n-SYM-CPA. Hence DDH and 1-SYM-CPA imply $0, 2n$-DH-SYM-CPA and so n-DH-SYM-CPA.

C Proof of the Main Theorem

Theorem: Let Π be a protocol. Let $\mathcal{SE}$ be the encryption scheme. If the DDH assumption holds and $\mathcal{SE}$ is SYM-CPA then for any concrete adversary $\mathcal{A}$:

$$pr\big(t_c, t_f \leftarrow Exec(\mathcal{A}, \Pi) \text{ and } t_f \notin traces(\Pi)\big) \text{ is negligible}$$

In this section, we suppose that all the nonces randomly generated have different values. This is justified in appendix D. Let $\mathcal{A}$ be an adversary such that its probability to create a symbolic trace that is not in $traces(\Pi)$ is not negligible. Let N be a bound on the number of different keys and nonces that Π may use. Using $\mathcal{A}$, we build an adversary $\mathcal{B}$ against N-DH-SYM-CPA. $\mathcal{B}$ randomly executes one of the two machines $\mathcal{B}_0$ and $\mathcal{B}_1$.

Adversary $\mathcal{B}_0$ handles the case where the trace is not possible because of $\mathcal{A}$ output a fresh symmetric encryption, an element of S or a fresh exponentiation. It is defined by $Exec(\mathcal{A}, \Pi)$ with modified versions of the *init*, *parse* and *concr* primitives.

– Exp-nonces and keys from S are the challenges of our criterion N-DH-SYM-CPA. Hence only the remaining nonces and keys have to be generated by *init*.
– *concr* uses oracles to concretizes messages encoded by a challenge key or challenge exp-nonces. These symbolic messages and their concretization are stored so that *parse* can use them. The first three hypothesis over protocols make it possible to use these oracles in every possible cases if the beginning of the trace is possible.

– *parse* parses the messages as usual. As it cannot decrypt messages m encoded by challenges there are two possibilities: either m has been generated in *concr* and its symbolic version is used by parse or m is a "fresh" encryption. In this last case, $\mathcal{B}_0$ wins its challenge against the UNF part of N-DH-SYM-CPA. If $\mathcal{B}_0$ receives a challenge exp-nonce or a key, it can deduce the value of the challenge bit b and $\mathcal{B}_0$ wins its challenge. The same thing occurs if $\mathcal{B}_0$ receives an exponentiation that *concr* did not build using its oracle (the way to recover b is detailed further in this section).

When $\mathcal{B}$ receives an exponentiation g^x and has to output g^{xy}, then our hypothesis over protocols allow two possible cases: x is known by $\mathcal{B}$ (either a bit-string or a challenge exp-nonce), hence the exponentiation oracle can be used to generate g^{xy}; or g^x is signed by a key in S, hence $\mathcal{B}$ generated g^x and also knows x.

Finally, if *Exec* terminates then $\mathcal{B}_0$ plays against IND and returns randomly 0 or 1.

Adversary $\mathcal{B}_1$ works exactly like $\mathcal{B}_0$. It handles the case where $\mathcal{A}$ output the value of a nonce that it should not know. For that purpose, $\mathcal{B}_1$ randomly chooses a nonce N among the $P(\eta)$ possible nonces. For this nonce, two values are generated N_0 and N_1. $\mathcal{B}_1$ uses its left-right encryption oracle to simulate the protocol using N_0 when $b = 0$ and N_1 when $b = 1$ (this technique was initiated in [MW04]). If $\mathcal{A}$ reveals the value of N, then $\mathcal{B}_1$ deduce the value of b. Else $\mathcal{B}_1$ returns randomly 0 or 1 for the value of b.

According to proposition 5, if *Exec* outputs t_f that is not in $traces(\Pi)$ then either $\mathcal{B}_0$ or one of the $P(\eta)$ possible $\mathcal{B}_1$ wins its challenge. The advantage of $\mathcal{B}$ can be approximated by:

$$(1 + P(\eta)).|\mathbf{Adv}(\mathcal{B})| \geq pr\big(t_c, t_f \leftarrow Exec(\mathcal{A}, \Pi) \text{ and } t_f \notin traces(\Pi)\big)$$

However, as $\mathcal{SE}$ verifies N-DH-SYM-CPA the advantage of $\mathcal{B}$ is negligible. Therefore, the probability to output a trace that is not in $traces(\Pi)$ is also negligible.

There is one last thing to detail: how can $\mathcal{B}_0$ deduce the value of b from a challenge nonce, key or an "interesting" exponentiation. If $\mathcal{B}_0$ knows a secret symmetric key, it uses the associated left-right oracle with $\langle 0, 1\rangle$ to get the value of b. If $\mathcal{B}_0$ knows nonce N, then it asks for g^E where E is a set such that $\mathcal{B}_0$ did not ask for $g^{E \cdot N}$ before. Then it uses the left-right oracle with $\langle 0, 1\rangle$ and $g^{E \cdot N}$ as key. Finally, if $\mathcal{B}_0$ knows g^E, then it uses directly the left-right oracle related to g^E (this is possible as g^E is not the output of the exponentiation oracle).

D Nonces are Probably Different

We consider that anytime a computational adversary picks up some nonces, they are different one from another. The adversary can only get a number m of nonces that is polynomial in η and we suppose that the number n of possible nonces is exponential in η (so $m < n$). Let p be the probability that the adversary gets two times the same nonces.

$$1 - p = \frac{n}{n}\frac{n-1}{n}...\frac{n-(m-1)}{n}$$

Thus, we have the following inequalities:

$$0 \leq p \leq 1 - \big(1 - \frac{m-1}{n}\big)^m$$

Proposition 7. *For any $x \in [0, 1[$ and $a \geq 1$,*

$$\big(1 - x\big)^a \geq 1 - x.a$$

Proof. Consider the function $f(x) = \big(1 - x\big)^a - 1 + x.a$. Derive it twice to get the result.

Applying the proposition, we get:

$$0 \leq p \leq \frac{m.(m-1)}{n}$$

As m is polynomial and n is exponential in η, p is negligible in η. When considering an adversary that has a non-negligible advantage against something, it still has its advantage if we consider only executions where nonces are distinct.

E Reduction Theorem

In order for the paper to be self-contained, the main theorem from [JLM05b] is given in this appendix:

Let $\gamma = (\theta_1, \theta_2; F_1, F_2; V_2)$ be a criterion. Let γ_1 and γ_2 be two criteria such that:

- There exist two PRTM G and H such that:

$$G(H(s, \theta_2, \theta_2'), 1, \theta_1) = F_1(s, \theta_1, \theta_2)$$
$$G(H(s, \theta_2, \theta_2'), 0, \theta_1) = F_1(s, \theta_1, \theta_2')$$

  Oracle G operates on a string of bits, thus it must receive two challenge information, a bit b and θ_1.
- $\gamma_2 = (\theta_2; F_2; V_2)$ and $\gamma_1 = (b, \theta_1; G; verif_b)$ where b generates a random bit and $verif_b$ is the PRTM verifying that the output of the adversary is b: $verif_b(s, b, \theta_1) = (s \Leftrightarrow b)$.
- $F_2(s, \theta_1, \theta_2)$ and $V_2(s, \theta_1, \theta_2)$ do not depend on θ_1.

Then we say that (γ_1, γ_2) is a *valid simplified partition* of γ.

Theorem 2 (Simplified Reduction Theorem). *Let (γ_1, γ_2) be a valid simplified partition of γ. For any PRTM $\mathcal{A}$, there exist two PRTM $\mathcal{A}^o$ and $\mathcal{B}$ such that*

$$|\mathbf{Adv}_{\mathcal{A}}^{\gamma}(\eta)| \leq 2.|\mathbf{Adv}_{\mathcal{B}}^{\gamma_1}(\eta)| + |\mathbf{Adv}_{\mathcal{A}^o}^{\gamma_2}(\eta)|$$