2013

CIL security proof for a password-based key exchange

Cristian Ene
Universite Grenoble, cristian.ene@imag.fr

Clementine Gritti
University of Wollongong, cjpg967@uowmail.edu.au

Yassine Lakhnech
Universite Grenoble, yassine.lakhnech@imag.fr

Publication Details

Ene, C., Gritti, C. & Lakhnech, Y. (2013). CIL security proof for a password-based key exchange. Lecture Notes in Computer Science, 8209 59-85.

CIL security proof for a password-based key exchange

Abstract

Computational Indistinguishability Logic (CIL) is a logic for reasoning about cryptographic primitives in computational model. It is sound for standard model, but also supports reasoning in the random oracle and other idealized models. We illustrate the benefits of CIL by formally proving the security of a Password-Based Key Exchange (PBKE) scheme, which is designed to provide entities communicating over a public network and sharing a short password, under a session key.

Keywords

cil, security, key, password, exchange, proof

Disciplines

Engineering | Science and Technology Studies

Publication Details

Ene, C., Gritti, C. & Lakhnech, Y. (2013). CIL security proof for a password-based key exchange. Lecture Notes in Computer Science, 8209 59-85.

CIL Security Proof for a Password-Based Key Exchange

Cristian Ene[1], Clémentine Gritti[2], and Yassine Lakhnech[1]

[1] Université Grenoble 1, CNRS, Verimag, France
{cristian.ene,yassine.lakhnech}@imag.fr
[2]Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
cjpg967@uowmail.edu.au

Abstract. Computational Indistinguishability Logic (CIL) is a logic for reasoning about cryptographic primitives in computational model. It is sound for standard model, but also supports reasoning in the random oracle and other idealized models. We illustrate the benefits of CIL by formally proving the security of a Password-Based Key Exchange (PBKE) scheme, which is designed to provide entities communicating over a public network and sharing a short password, under a session key.

Keywords: Password-Based Key Exchange, Logic, Security Proof.

1 Introduction

Cryptography plays a central role in the design of secure and reliable systems. It consists in the conception and analysis of protocols achieving various aspects of information security such as authentication. In particulary, the *provable cryptography* is defined as the conception of proofs accounting for the exact amount of security supplied by cryptographic protocols.

In the computational model, Computational Indistinguishability Logic (CIL) supports concise and intuitive proofs accross several models of cryptography. This logic features the notion of oracle system, an abstract model of interactive games in which adaptative adversaries play against a cryptographic scheme by interacting with oracles. Moreover, it states a small set of rules that capture common reasoning patterns and interface rules to connect with external reasoning. To illustrate applicability of CIL, we consider the security proof of the Password-Based Key Exchange (PBKE) protocol.

1.1 Related Work

About Security of PBKE Protocols: EKE (Encrypted Key Exchange) was introduced by Bellovin and Merritt, [1]. In their protocol, two users execute an encrypted version of the Diffie-Hellman key exchange protocol, in which each flow is encrypted using the password shared between these two users as the symmetric key. Due to the simplicity of their protocol, other protocols were proposed in the literature based on it, each with its own instantiation of the encryption function such that OEKE (One-Encryption Key-Exchange) protocol.

Since 2003, E. Bresson *et al.*, [3], have been working on the analysis of very efficient schemes on password-based authenticated key exchange methods, but for which actual security was an open problem. In 2012, B. Blanchet have focused on a crytpgraphic protocol verifier, called CryptoVerif, to mechanically prove OEKE.

About CIL: DCS (Distributed and Complex Systems) is working on the logic CIL for proving concrete security of cryptographic schemes. It enables reasonning about schemes directly in the computational settings. The main contribution is to support the design of proofs at a level of abstraction which allows to bridge the gap between pencil-and-paper fundamental proofs and existing pratical verification tools (see article [7]).

1.2 Contributions and Contents

For the first time, we bring out the applicability of CIL for formalizing computational proofs. The tool CIL allows us to give a new kind of analysis that has advantages over the traditional as in [3] and [9]. As we use a tool based on general and extended logic rules, the proofs are well constructed and easy to understand, and achieve good results.

The paper begins with a recall of the framework to capture cryptographic games(Section 2). The main technical contributions of the paper are: i) an extension of reasoning tools for oracle systems (Section 3); ii) a formal proof in CIL of an efficient PBKE protocol (Section 4).

2 Oracle systems

2.1 Preliminaries

ICM: An ideal block cipher is a totally random permutation from l-bit strings to l-bit strings.

ROM: A random oracle is a mathematical function mapping every possible query to a uniformly random response from its output domain.

Miscellaneous: Let $\mathbf{1}$ to denote the unit type and (x, y) to denote pairs. For a set A, $U(A)$ defines the set of uniform distributions over A. Let $\_$ to denote arguments that are not used or elements of tuples whose value is irrevelant in the final distribution.

2.2 Semantics

The interaction between an oracle system and an adversary proceeds in three successive phases:
- the initialization oracle sets the initial memory distributions of the oracle system;
- the adversary performs computations, updates its state and submits a query to the oracle system; the oracle system performs computations, updates its state, and replies to the adversary, which updates its state;
- the adversary outputs a result calling the finalization oracle.

During his attack, the adversary has access to the oracles, which modelize his capacities to obtain (partial) information or to execute some party of the protocol in the reality. His resources are bounded by two parameters: the number of queries he performs to the oracles and his running time.

2.3 Oracle systems and adversaries

Oracle systems and adversaries are modeled as stateful systems meant to interact with each another. An oracle system O is a stateful system that provides oracle access to adversaries and given by:
- sets of oracle memories and of oracles;
- a query domain, an answer domain and the related implementation;
- a distinguished initial memory, and distinguished oracles o_I for initialization and o_F for finalization.

Oracle systems O and O' are *compatible* iff they have the same sets of oracle names and the query and the answer domains of each oracle name coincide in both oracle systems. We build compatible systems out of systems we have already defined by modifying the implementation of one of the oracles.

2.4 Events

The interaction between oracle system and adversary seems as this of the pattern consisting in the query of an oracle, the computation of an answer by the oracle, and the update of its state by the adversary. This is formalized as a transition system, where a step consists in one occurence of the pattern.

Security properties abstract away from the state of adversaries and are modeled using traces. A trace is an execution sequence from which the adversary memories have been erased. The subset of traces verifying the predicate is considered to assign a probability to an event defined by a predicate.

For a step-predicate ϕ, let the event "eventually ϕ" be denoted by F_ϕ and correspond to ϕ satisfied at one step of the trace. Furthermore, the event "always ϕ", denoted by G_ϕ, is true iff ϕ is satisfied at every step of the trace. You can find an example of this concept in Appendix A.3.

For more details and examples, you can see the Appendix A or refer to the article [7].

3 Computational Indistinguishability Logic

3.1 Statements: judgments

For an event $\mathbf{E}$, a statement $O :_\varepsilon \mathbf{E}$ is valid iff for every (k,t)-adversary A, $Pr(A \mid O : \mathbf{E}) \le \varepsilon(k,t)$. For O and O' compatible oracle systems which expect a boolean as result, a statement $O \sim_\varepsilon O'$ is valid iff for every (k,t)-adversary A, $\mid Pr[A \mid O : R = \mathbf{True}] - Pr[A \mid O' : R = \mathbf{True}] \mid \le \varepsilon(k,t)$. Let $\mathbf{E}$ be an event of compatible systems O and O'. A statement $O \overset{\mathbf{E}}{\sim}_\varepsilon O'$ is valid iff for every (k,t)-adversary A, $\mid Pr[A \mid O : R = \mathbf{True} \wedge \mathbf{E}] - Pr[A \mid O' : R = \mathbf{True} \wedge \mathbf{E}] \mid \le \varepsilon(k,t)$. As $O \sim_\varepsilon O' \Leftrightarrow O \overset{\mathbf{True}}{\sim}_\varepsilon O'$, we write $O \sim_\varepsilon O'$ for the two statements. See Appendix B.1 for details.

3.2 Rules and their extensions

We expose briefly the rules used in our proof on Figure (1). You can find more classic and extended rules in Appendix B.1.

$$\frac{O :_{\varepsilon_2} \mathbf{E}_2 \quad O' :_{\varepsilon_1} F_{\neg\varphi} \quad O \equiv_{\mathcal{R},\varphi} O' \quad \mathbf{E}_1 \mathcal{R} \mathbf{E}_2}{O' :_{\varepsilon_1+\varepsilon_2} \mathbf{E}_1} \; \textit{UpToBad} \qquad \frac{}{O :_\varepsilon F_\varphi} \; \textit{Fail}$$

$$\frac{O \le_{\det,\gamma} O' \qquad O :_\varepsilon \mathbf{E} \circ \pi}{O' :_\varepsilon \mathbf{E}} \; \textit{B-Det-Left} \qquad \frac{O :_\varepsilon \mathbf{E} \circ C}{C[O] :_{\varepsilon'} \mathbf{E}} \; \textit{B-Sub}$$

$$\frac{O \overset{\mathbf{E}_2}{\sim}_{\varepsilon_1} O' \qquad \mathbf{E}_2 \Rightarrow \mathbf{E}_1 \qquad O :_{\varepsilon_2} \mathbf{E}_1 \wedge \neg\mathbf{E}_2 \qquad O' :_{\varepsilon_2} \mathbf{E}_1 \wedge \neg\mathbf{E}_2}{O \overset{\mathbf{E}_1}{\sim}_{\varepsilon_1+\varepsilon_2} O'} \; \textit{URCd} \qquad \frac{}{O :_{\varepsilon'} F_{\varphi'}} \; \textit{Fail2}$$

$$\frac{O \overset{\mathbf{E}_1 \wedge \mathbf{E}_2}{\sim}_{\varepsilon_2} O' \qquad O :_{\varepsilon_1} \neg\mathbf{E}_1 \wedge \mathbf{E}_2 \qquad O' :_{\varepsilon_1} \neg\mathbf{E}_1 \wedge \mathbf{E}_2}{O \overset{\mathbf{E}_2}{\sim}_{\varepsilon_1+\varepsilon_2} O'} \; \textit{FTr} \qquad \frac{O \overset{\mathbf{E}_1}{\sim}_{\varepsilon_1} O' \qquad O' \overset{\mathbf{E}_2}{\sim}_{\varepsilon_2} O"}{O \overset{\mathbf{E}_1 \vee \mathbf{E}_2}{\sim}_{\varepsilon_1+\varepsilon_2} O"} \; \textit{TrCd}$$

$$\frac{O :_{\varepsilon_1} F_{\varphi_1} \wedge G_{\varphi_2} \quad O :_{\varepsilon_2} F_{\neg\varphi_2} \quad O \equiv_{\mathcal{R},\varphi_2} O'}{O' :_{\varepsilon_1+\varepsilon_2} F_{\varphi_1}} \; \textit{B-BisG2} \qquad \frac{O' :_\varepsilon F_{\neg\varphi_2} \wedge G_{\varphi_1} \qquad O \overset{\varphi_1}{\equiv}_{\mathcal{R},\varphi_2} O'}{O \overset{G_{\varphi_1}}{\sim}_\varepsilon O'} \; \textit{I-BisCd}$$

Fig. 1. Rules used in the proof (classic and extended rules). For compatible oracle systems O, O' and $O"$, events $\mathbf{E}$, $\mathbf{E}_1$ and $\mathbf{E}_2$ of O, O' and $O"$, and step-predicates φ, φ_1 and φ_2.

3.3 Contexts

A context C is an intermediary between an oracle system O and adversaries. One can compose a O-context C with O to obtain a new oracle system $C[O]$ and with a $C[O]$-adversary to obtain a new O-adversary $C \parallel A$. Procedures for contexts differ of these for oracle systems: one that transfers calls from the adversary to the oracles and another one that tranfers answers from the oracles to the adversary. See Appendix B.2.

3.4 Bisimulation

Game-based proofs proceed by transforming an oracle system into an equivalent one, or in case of imperfect simulation into a system that is equivalent up to some bad event. The notion of bisimulation-up-to is defined as two probabilistic transition systems are bisimilar until the failure of a condition on their tuple states-transitions. Bisimulations are closely related to obversational equivalence and relational Hoare logic and allow to justify proofs by simulations. Besides, bisimulations-up-to subsume the Fundamental Lemma of Victor Shoup. See Appendix B.3.

3.5 Determinization

Using the concept of automata determinization technique, the definition is based on the possibility to decompose states of a system into two components and to exhibit a distribution γ allowing to obtain the second component given the first one. See Appendix B.4.

4 CIL Security Proof for an efficient PBKE

4.1 Preliminaries

In the computational model, messages are bitstrings, cryptographic primitives are functions from bitstrings to bitstrings and adversary is any Probabilistic Polynomial time Turing Machine.

Scheme: We denote objects describing the model:

- two sets $Users$ and $Servers$ such that $u \in [Users]$ and $s \in [Servers]$;
- for the arithmetic, $G = < g >$ is a cyclic group of l-bit prime order q and $\bar{G} = G \setminus 1_G = \{g^x \mid x \in \mathbb{Z}_q^*\}$ (g is a fixed parameter);
- for $i = \{0,1\}$, l_i is the parameter of data size for Hash function H_i;
- a set $Password$ as a small dictionary (polynomial in the security parameter), of size N, equipped with the uniform distribution.

Encryption/Decryption: E is the Encryption and D is the Decryption in the Ideal Cipher Model .

Hash Functions: There are two hash functions H_0 and H_1 in the Random Oracle Model.

We want to bound the probability for an adversary, within time t, and with less than N_u sessions with a client, N_s sessions with a server (active attacks), and asking q_H hash queries and q_E Encryption/Decryption queries, to distinguish the session key from a random key.

4.2 One-Encryption Key-Exchange (OEKE), a password-based key exchange

On Figure (2) (with a honest execution of the OEKE protocol), the protocol runs between a client u and a server s. The session key space associated to this protocol is $\{0,1\}^{l_0}$ equipped with the uniform distribution. u and s initially share a low-quality string pw, the password, from $Password$.

Client u		Server s
pw		pw
accept $\leftarrow$ false ; terminate $\leftarrow$ false		accept $\leftarrow$ false ; terminate $\leftarrow$ false
$x \leftarrow [1..(q-1)]$		$y \leftarrow [1..(q-1)]$
$X \leftarrow g^x$	$\xrightarrow{u,X}$	$Y \leftarrow g^y$
$Y \leftarrow D(pw, Y^\star)$	$\xleftarrow{s,Y^\star}$	$Y^\star \leftarrow E(pw, Y)$
$K_u \leftarrow Y^x$; $Auth \leftarrow H_1(Z \parallel K_u)$; $sk_u \leftarrow H_0(Z \parallel K_u)$		$K_s \leftarrow X^y$
accept $\leftarrow$ true	$\xrightarrow{Auth}$	$Auth \stackrel{?}{=} H_1(Z \parallel K_s)$; if true, accept $\leftarrow$ true
		$sk_s \leftarrow H_0(Z \parallel K_s)$
terminate $\leftarrow$ true		terminate $\leftarrow$ true

Fig. 2. An execution of the protocol OEKE, run by the client u and the server s. We let Z be equal to $u \parallel s \parallel X \parallel Y$.

The real game O_0^1: This game consists of: initialization and finalization oracles, Encryption/Decryption oracles, Hash oracles, oracles that simulate the protocol (named U_1, S_1, U_2 and S_2), Execute oracle, Test oracle and Reveal oracle. In the initialization oracle, the bit b is equal to 1 and hence, the Test oracle returns the real value of the session key.

$\mathrm{Imp}(o_I)() =$
$\quad pw \leftarrow Password;\ L_{H_0} := [];\ L_{H_1} := [];$
$\quad L_E := [];\ L_{pw} := [];\ L_O := [];$
$\quad var_X := \perp;\ var_\theta := \perp;\ var_\varphi := \perp;\ var_{sk} := \perp;$
$\quad b := 1$
$\quad$ return $\mathbf{1}$

$\mathrm{Imp}(E)(pw, x) =$
$\quad$ if $(pw, x, \_, \_) \notin L_E$ then
$\qquad y \leftarrow \bar{G};\ L_E := L_E.(pw, x, y, \perp);$
$\quad$ endif
$\quad$ return y such that $(pw, x, y, \_) \in L_E$

$\mathrm{Imp}(D)(pw, y) =$
$\quad$ if $(pw, \_, y, \_) \notin L_E$ then
$\qquad \phi \leftarrow \mathbb{Z}_q^*;\ x = g^\phi\ ;\ L_E := L_E.(pw, x, y, \phi);$
$\quad$ endif
$\quad$ return x such that $(pw, x, y, \_) \in L_E$

$\mathrm{Imp}(H_0)(x) =$
$\quad$ if $x \notin L_{H_0}$ then
$\qquad y \leftarrow U(l_0);\ L_{H_0} := L_{H_0}.(x, y);$
$\quad$ endif
$\quad$ return $L_{H_0}(x)$

$\mathrm{Imp}(H_1)(x) =$
$\quad$ if $x \notin L_{H_1}$ then
$\qquad y \leftarrow U(l_1);\ L_{H_1} := L_{H_1}.(x, y);$
$\quad$ endif
$\quad$ return $L_{H_1}(x)$

$\mathrm{Imp}(U_1)(u, i) =$
$\quad \theta \leftarrow \mathbb{Z}_q^*;\ X = g^\theta\ ;\ var_\theta[(u, i)] = (\theta, X);$
$\quad$ return (u, X)

$\mathrm{Imp}(S_1)((s, j), (u, X)) =$
$\quad \varphi \leftarrow \mathbb{Z}_q^*;\ Y = g^\varphi;\ Y^\star = E(pw, Y);$
$\quad var_\varphi[(s, j)] = (\varphi, Y, Y^\star);\ var_X[(s, j)] = X;$
$\quad K_s = X^\varphi$
$\quad$ return $(s, Y^\star)$

$\mathrm{Imp}(U_2)((u, i), (s, Y^\star)) =$
$\quad$ if $var_\theta[(u, i)]! = \perp$ then
$\qquad Y = D(pw, Y^\star);\ (\theta, X) = var_\theta[(u, i)];$
$\qquad K_u = Y^\theta;$
$\qquad Auth = H_1(u \parallel s \parallel X \parallel Y \parallel K_u);$
$\qquad var_{sk}[(u, i)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_u)$
$\quad$ endif
$\quad$ return $Auth$

$\mathrm{Imp}(S_2)((s, j), u, Auth) =$
$\quad$ if $var_\varphi[(s, j)]! = \perp$ then
$\qquad (\varphi, Y, Y^\star) = var_\varphi[(s, j)];\ X = var_X[(s, j)];$
$\qquad K_s = X^\varphi;$
$\qquad H' = H_1(u \parallel s \parallel X \parallel Y \parallel K_s);$
$\qquad$ if $H' = Auth$ then
$\qquad\quad var_{sk}[(s, j)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_s)$
$\qquad$ endif
$\quad$ endif
$\quad$ return $\mathbf{1}$

$\mathrm{Imp}(Reveal)(p, k) =$
$\quad$ if $var_{sk}[(p, k)]! = \perp$ then
$\qquad sk := var_{sk}[(p, k)]$
$\quad$ endif
$\quad$ return sk

$\mathrm{Imp}(Test^1)(p, k) =$
$\quad$ if $var_{sk}[(p, k)]! = \perp$ then
$\qquad sk := var_{sk}[(p, k)]$
$\quad$ endif
$\quad$ return sk

$\mathrm{Imp}(Exec)((u, i), (s, j)) =$
$\quad \theta \leftarrow \mathbb{Z}_q^*;\ X = g^\theta;\ \varphi \leftarrow \mathbb{Z}_q^*;$
$\quad Y = g^\varphi;\ Y^\star = E(pw, Y);\ K_s = X^\varphi;\ K_u = Y^\theta;$
$\quad Auth = H_1(u \parallel s \parallel X \parallel Y \parallel K_u);$
$\quad var_{sk}[(u, i)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_u)$
$\quad$ return $((u, X), (s, Y^\star), Auth)$

$\mathrm{Imp}(o_F)(x) =$ return $\mathbf{1}$

The real game O_0^0: As for O_0^1, this game consists of exactly the same oracles. The differences are in the initialization oracle where $b = 0$ and in the Test oracle where is returned a random value for sk.

Summary: In a first part, we bound the probabilities that two step-predicates occur. The first one, **Cl**, is for formalizing the collisions. The second one, ϕ_{pw}, is for describing the dependence on the password in the oracles. In a second part, we write the general proof in order to obtain the indistinguishability between O_0^0 and O_0^1, considering that the two previous step-prediactes can not occur. For that, we describe the transformations of the game O_0^1, step by step, until finding a simplified game. We notice that we obtain the same thing for the game O_0^0.

These two parts are very similar: the same tranformations are made in order to obtain the wanted result. Therefore, we explain clearly the first proof and we expose briefly the second one.

N.B.: The list L_{pw} is created to simulate the oracles E and D in ICM. We suppose that the domain of E matches with the group generated by g. L_O is defined as the list stocking the tuple (oracle o, query q, answer a), writing as $L_O = L_O \cdot (o, q, a)$.

4.3 Proof for bounding the probability of the step-predicate ϕ_{pw}

C.1. Eliminating the Collisions :

We want to eliminate collisions during Hash and Encryption/Decryption processes. We formalize the small probability of that an inappropriate collision could let the adversary to find a sequence without any required effort.

Let the step-predicate **Cl** be defined on the triple $((o, q, a), m, \_)$ as the conjunction of the clauses:
- for $i = 0, 1$, $o = H_i \wedge q \notin m \cdot L_{H_i} \wedge (, a) \in m \cdot L_{H_i}$
- $o = E \wedge (pw, q, \_, \_) \notin m \cdot L_E \wedge (\_, \_, a, \_) \in m \cdot L_E$
- $o = D \wedge (pw, \_, q, \_) \notin m \cdot L_E \wedge (\_, a, \_, \_) \in m \cdot L_E$

To complete and restrict the definition of **Cl**, let us introduce two other clauses:
- if $(pw, Y, Y_1^\star, \varphi)$ and $(pw, Y, Y_2^\star, \varphi)$ then $Y_1^\star = Y_2^\star$
- if $(pw, Y_1, Y^\star, \varphi)$ and $(pw, Y_2, Y^\star, \varphi)$ then $Y_1 = Y_2$

Since **Cl** can only be satisfied when querying H_0, H_1, E or D, applying the rule Fail2 (see Appendix B.1) allows to conclude to:
- on the hash oracles, where $l = max(l_0, l_1)$ and $q_H = q_{H_0} + q_{H_1}$, we obtain $\varepsilon_0^1 = \frac{1}{2} \times \frac{(q_{H_0}+q_{H_1})^2}{2^l} = \frac{q_H^2}{2^{l+1}}$,
- on the Encryption/Decryption oracles, where $q_E = q_{Enc} + q_{Dec}$, we get $\varepsilon_0^2 = \frac{1}{2} \times \frac{(q_{Enc}+q_{Dec})^2}{q-1} = \frac{q_E^2}{2(q-1)}$.

Therefore, we obtain that $O_0^1 :_{\varepsilon_0} F_{\mathbf{Cl}}$ where $\varepsilon_0 = \frac{q_H^2}{2^{l+1}} + \frac{q_E^2}{2(q-1)}$. We perform the same analysis for the other game obtaining $O_0^0 :_{\varepsilon_0} F_{\mathbf{Cl}}$.

For further, at each step, we suppose there is no collision when modifying the game O_0^1. We can introduce a particular equivalence relation under the step-predicate $\neg\mathbf{Cl}$ in order to avoid the collisions, since it steps in over memories. We use the extented notion of bisimulation (for more details, see Appendix B.3). To conclude the proof, we bound the probability of such collisions (this avoids the repetition of the value ε_0 at each transformation).

C.2. Creating the independence from the password in the oracles:

We want to eliminate dependence on *pw* in all the oracles. We formalize the probability that the adversary guesses the good password and succeeds in the acquisition of the session key.

We define the step-predicate $\phi_{pw} = \phi_{pw1} \vee \phi_{pw2}$, where ϕ_{pw1} and ϕ_{pw2} are written as follows:

$$\phi_{pw1} = \lambda(m, \_).\ (U_2, q, \_) \in m \cdot L_O \wedge (m \cdot pw, \_, q, \bot) \in m \cdot L_E$$

$$\begin{aligned}\phi_{pw2} = \lambda(m, \_).\ & (S_1, \_, a) \in m \cdot L_O \wedge (\_, a) \in m \cdot S_1 \wedge (m \cdot pw, Y, a, \_) \in m \cdot L_E \\ & \wedge (\_ \parallel \_ \parallel \_ \parallel Y \parallel \_, a') \in m \cdot L_{H_1} \wedge (S_2, a', \_) \in m \cdot L_O\end{aligned}$$

ϕ_{pw} steps in over memories only. We want to find the value ε_1 such that: $O_0^1 :_{\varepsilon_1} F_{\phi_{pw}} = F_{\phi_{pw1} \vee \phi_{pw2}}$.

We transform the game O_0^1 until finding a game wherein the password is sampled in the finalization oracle. Therefore, we can obtain easily the optimal result $\frac{N_u+N_s}{N}$. Indeed, this means that the adversary can test at most one password per session.

Removing the Encryption in the oracle S_1 The unique way for the adversary to gain something is to correctly guess pw, by either sending a $Y^\star$ that is really an encryption under it of some well-chosen message or using it to decrypt $Y^\star$. In O_1^1, we change S_1 modelizing the Encryption inside this oralce.

$$\begin{aligned}\mathrm{Imp}(S_1)((s,j),(u,X)) = {} & \varphi \leftarrow \mathbb{Z}_q^*;\ Y = g^\varphi;\ Y^\star \leftarrow \bar{G};\ var_\varphi[(s,j)] = (\varphi, Y, Y^\star);\\ & L_E := L_E.(pw, Y, Y^\star, \varphi)\ ;\ var_X[(s,j)] = X;\ K_s = X^\varphi;\\ & \text{return } (s, Y^\star) \text{ such that } (pw, Y, Y^\star, \_) \in L_E\end{aligned}$$

In a particular case, we do not receive an exponent φ but $\perp$: that happens when $Y^\star$ has been previously obtained as a ciphertext returned by an Encryption query. Let the step-predicate **Exp** be this case:

$$\mathbf{Exp} = \lambda((o,\_,a),m,\_).\ o = S_1 \wedge (pw,\_,a,\perp) \in m \cdot L_E$$

Therefore, O_0^1 and O_1^1 are in bisimulation-up-to $\neg\mathbf{Exp}$, using as relation $\mathcal{R}'_1$ the equality on the common components of their states in $M_{\neg\mathbf{Cl}}^{O_i^1}$. Indeed, states m, m' are in relation:

- if $m, m' \in M_{\neg\mathbf{Cl}}^{O_0^1}$ or $M_{\neg\mathbf{Cl}}^{O_1^1}$, $m\mathcal{R}'_1 m'$ iff $m = m'$
- if $m \in M_{\neg\mathbf{Cl}}^{O_0^1}$ and $m' \in M_{\neg\mathbf{Cl}}^{O_1^1}$, $m\mathcal{R}'_1 m'$ iff
  - $\forall (pw,x,y,e) \in m \cdot L_E \setminus m' \cdot L_E \Rightarrow e = \perp \wedge \exists (pw,x,y,\varphi) \in m' \cdot L_E \setminus m \cdot L_E$ s.t. $x = g^\varphi$
  - $\forall (pw,x,y,e) \in m' \cdot L_E \setminus m \cdot L_E \Rightarrow e = \varphi$ s.t. $x = g^\varphi \wedge \exists (pw,x,y,\perp) \in m \cdot L_E \setminus m' \cdot L_E$

Hence, we apply the rule I-BisG2 to result in:

$$\frac{O_1^1 :_{\varepsilon'_2} F_{\mathbf{Exp}}(\wedge G_{\neg\mathbf{Cl}})\quad O_1^1 :_{\varepsilon'_1} F_{\phi_{pw}}(\wedge G_{\neg\mathbf{Cl}})\quad O_0^1 \overset{\neg\mathbf{Cl}}{\equiv}_{\mathcal{R}'_1, \neg\mathbf{Exp}\wedge\neg\phi_{pw}} O_1^1}{O_0^1 :_{\varepsilon'_1+\varepsilon'_2} F_{\phi_{pw}}(\wedge G_{\neg\mathbf{Cl}})}\ \textit{I-BisG2}$$

Applying the rule Fail allows to obtain $O_1^1 :_{\varepsilon'_2} F_{\mathbf{Exp}}$, where $\varepsilon'_2 = \frac{N_s \times q_E}{q-1}$.

Splitting the Hash lists We want to be sure that u will offer a good Authenticator and s will accept it. Therefore, we modify the oracle U_2 in order to get a honest value for Y. We split the lists of the two public hash oracles H_0 and H_1 in O_2^1, introducing two private hash functions $H_2 : \{0,1\}^* \to \{0,1\}^{l_0}$ and $H_3 : \{0,1\}^* \to \{0,1\}^{l_1}$.

```
Imp(o_I)() =
    pw ← Password
    L_H0 := []; L_H1 := []; L_H2 := []; L_H3 := [];
    L_E := []; L_pw := []; L_O := [];
    var_X :=⊥; var_θ :=⊥; var_φ :=⊥; var_sk :=⊥;
    b := 1
    return 1
```

```
Imp(U_2)((u,i),(s,Y*)) =
    if var_θ[(u,i)]! =⊥ then
      (θ,X) = var_θ[(u,i)]
      if ∃Y,∃φ such that (pw,Y,Y*,φ) ∈ L_E
        K_u = Y^θ ;
        Auth = H_1(u || s || X || Y || K_u);
        var_sk[(u,i)] = H_0(u || s || X || Y || K_u)
      else
        Y ← Ḡ; K_u = Y^θ;
        Auth = H_3(u || s || X || Y || K_u);
        var_sk[(u,i)] = H_2(u || s || X || Y || K_u)
      endif
    endif
    return Auth
```

O_1^1 and O_2^1 are $\mathcal{R}'_2$-bismilar up to $\neg\phi_{pw1}$. The equivalence relation $\mathcal{R}'_2$ between states m and m' is as follows:

- if $m, m' \in M_{\neg\mathbf{Cl}}^{O_1^1}$ or $M_{\neg\mathbf{Cl}}^{O_2^1}$, $m\mathcal{R}'_2 m'$ iff $m = m'$
- if $m \in M_{\neg\mathbf{Cl}}^{O_1^1}$ and $m' \in M_{\neg\mathbf{Cl}}^{O_2^1}$, $m\mathcal{R}'_2 m'$ iff $m \cdot L_{H_0} = m' \cdot (L_{H_0} \cup L_{H_2})$ and $m \cdot L_{H_1} = m' \cdot (L_{H_1} \cup L_{H_3})$

Then, applying the rule I-BisG2, we find:

$$\frac{O_2^1 :_{\varepsilon'_3} F_{\phi_{pw1}}(\wedge G_{\neg\mathbf{Cl}})\quad O_2^1 :_{\varepsilon'_4} F_{\phi_{pw}} \wedge G_{\neg\phi_{pw1}}(\wedge G_{\neg\mathbf{Cl}})\quad O_1^1 \overset{\neg\mathbf{Cl}}{\equiv}_{\mathcal{R}'_2, \neg\phi_{pw1}} O_2^1}{O_1^1 :_{\varepsilon'_3+\varepsilon'_4} F_{\phi_{pw}}(\wedge G_{\neg\mathbf{Cl}})}\ \textit{I-BisG2}$$

such that $\varepsilon'_3 + \varepsilon'_4 = \varepsilon'_1$. We notice that: $F_{\phi_{pw}} \wedge G_{\neg\phi_{pw1}} \Leftrightarrow F_{\phi_{pw2}}$.

Randomizing the Hash oracles In O^1_3, we sample the value of Y. Therefore, we no longer use the private hash functions since we internalize the hash functions in another way with the random Y. We modify the oracles U_2 and S_2.

$$\begin{array}{l}
\mathrm{Imp}(U_2)((u,i),(s,Y^\star)) = \\
\quad \text{if } var_\theta[(u,i)]! = \perp \text{ then} \\
\quad\quad Y \leftarrow \bar{G};\ (_,Y,Y^\star) \in var_\varphi[(u,i)]; \\
\quad\quad (\theta,X) = var_\theta[(u,i)];\ K_u = Y^\theta; \\
\quad\quad Auth = H_1(u \parallel s \parallel X \parallel Y \parallel K_u); \\
\quad\quad var_{sk}[(u,i)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_u) \\
\quad \text{endif} \\
\quad \text{return } Auth
\end{array}$$

$$\begin{array}{l}
\mathrm{Imp}(S_2)((s,j),u,Auth) = \\
\quad \text{if } var_\varphi[(s,j)]! = \perp \text{ then} \\
\quad\quad (\varphi,Y,Y^\star) = var_\varphi[(s,j)];\ X = var_X[(s,j)];\ K_s = X^\varphi; \\
\quad\quad H' = H_1(u \parallel s \parallel X \parallel Y \parallel K_s); \\
\quad\quad \text{if } H' = Auth \text{ then} \\
\quad\quad\quad var_{sk}[(s,j)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_s) \\
\quad\quad \text{endif} \\
\quad \text{endif} \\
\quad \text{return } \mathbf{1}
\end{array}$$

Let the step-predicate **Auth** be the conjunction of the following clauses:

- $(pw,Y,Y^\star,\varphi) \in L_E \wedge X \in var_\theta$
- for u and s, $u \parallel s \parallel X \parallel Y \parallel CDH(X,Y) \in L_{H_1}$

The adversary can not see the link between Y and $Y^\star$, except if he calls $E(pw,\_)$ or $D(pw,\_)$.

We notice that the probability that $F_{\phi pw2}$ occurs is very negligible since we suppose that the adversary can not get the password. Since we have $F_{\mathbf{Auth}\vee\phi_{pw2}} = F_{\mathbf{Auth}} \vee (F_{\phi_{pw2}} \wedge G_{\neg\mathbf{Auth}})$, we expose that $F_{\phi_{pw2}} \wedge G_{\neg\mathbf{Auth}}$ occurs with the probability ε'_5 and $F_{\mathbf{Auth}}$ with ε'_6. Using the rule Fail, we get $\varepsilon'_5 = \frac{N_u+N_s}{q-1}$.

We want to establish the indistinguishability between O^1_2 and O^1_3 up to $\neg\mathbf{Auth} \wedge \neg\phi_{pw2}$. We exhibit two equivalence relations $\mathcal{R}'_3$ between both systems. Indeed, states m and m' are in relation:

- if $m,m' \in M^{O^1_2}_{\neg\mathbf{Cl}}$ or $M^{O^1_3}_{\neg\mathbf{Cl}}$, $m\mathcal{R}'_3m'$ iff $m = m'$
- if $m \in M^{O^1_2}_{\neg\mathbf{Cl}}$ and $m' \in M^{O^1_3}_{\neg\mathbf{Cl}}$, $m\mathcal{R}'_3m'$ iff $m\cdot(L_{H_0} \cup L_{H_2}) = m'\cdot L_{H_0}$ and $m\cdot(L_{H_1} \cup L_{H_3}) = m'\cdot L_{H_1}$

On the left hand, focusing on the step-predicate ϕ_{pw1}, we apply the rule I-BisG2 to result in:

$$\frac{O^1_3 :_{\varepsilon'_5+\varepsilon'_6} F_{\mathbf{Auth}\vee\phi_{pw2}}(\wedge G_{\neg\mathbf{Cl}}) \quad O^1_3 :_{\varepsilon'_7} F_{\phi_{pw1}} \wedge G_{\neg\mathbf{Auth}\wedge\neg\phi_{pw2}}(\wedge G_{\neg\mathbf{Cl}}) \quad O^1_2 \overset{\neg\mathbf{Cl}}{\equiv}_{\mathcal{R}'_3,\neg\mathbf{Auth}\wedge\neg\phi_{pw2}} O^1_3}{O^1_2 :_{\varepsilon'_5+\varepsilon'_6+\varepsilon'_7} F_{\phi_{pw1}}(\wedge G_{\neg\mathbf{Cl}})} \textit{ I-BisG2}$$

such that $\varepsilon'_5 + \varepsilon'_6 + \varepsilon'_7 = \varepsilon'_3$.

On the right hand, since we have $F_{\mathbf{Auth}\vee\phi_{pw2}} = [F_{\mathbf{Auth}} \wedge G_{\phi_{pw2}}] \vee [F_{\phi_{pw2}} \wedge G_{\mathbf{Auth}\wedge\phi_{pw2}}]$ and $O^1_3 :_0 F_{\phi_{pw2}} \wedge G_{\mathbf{Auth}\wedge\phi_{pw2}}(\wedge G_{\neg\mathbf{Cl}})$, we simplify the line. Focusing on the step-predicate ϕ_{pw2}, we apply the rule I-BisG2 to result in:

$$\frac{O^1_3 :_{\varepsilon'_6} F_{\mathbf{Auth}} \wedge G_{\phi_{pw2}}(\wedge G_{\neg\mathbf{Cl}}) \quad O^1_3 :_{\varepsilon'_8} F_{\phi_{pw2}}(\wedge G_{\neg\mathbf{Cl}}) \quad O^1_2 \overset{\neg\mathbf{Cl}}{\equiv}_{\mathcal{R}'_3,\neg\mathbf{Auth}\wedge\neg\phi_{pw2}} O^1_3}{O^1_2 :_{\varepsilon'_6+\varepsilon'_8} F_{\phi_{pw2}}(\wedge G_{\neg\mathbf{Cl}})} \textit{ I-BisG2}$$

such that $\varepsilon'_6 + \varepsilon'_8 = \varepsilon'_4$.

We focus on the CDH problem to obtain the value of ε'_6 (for more details about the Computational Diffie-Hellman assumption in G, see Appendix B.2). Hence, we write the game O^1_4 as a context C of CDH. The oracle system CDH captures the game played by an adversary to find the Diffie-Hellman instance (A,B).

We define the step-predicate **Auth'** as follows:

- $o = U_1$ s.t. $(\alpha,X) \in L_A \wedge o = S_1$ s.t. $(\beta,Y) \in L_B$
- for u and s, $u \parallel s \parallel X \parallel Y \parallel CDH(X,Y) \in L_{H_1}$

The adversary has returned a pair (R_1,R_2) that is a valid authentication when $H_1(R_1) = R_2$. Given $(\alpha,X) \in L_A$, $(\beta,Y) \in L_B$ and one CDH instance (A,B), we notice that $CDH(A,B) = CDH(X,Y)^{\alpha^{-1}\beta^{-1}}$.

Therefore, applying the rule B-Sub, we get:

$$\frac{CDH :_{\varepsilon(\mathbf{1}_k,t)} F_{\mathbf{Auth'}} \circ C}{O^1_4 = C[CDH] :_{\varepsilon'_6} F_{\mathbf{Auth'}}} \textit{ B-Sub}$$

where $\varepsilon'_6 = q_H \times \varepsilon(\mathbf{1}_k,t)$ (see Appendix B.2).

Moreover, the games O^1_3 and O^1_4 are in perfect bisimulation. We define the equivalence relation $\mathcal{R}'_4$ between states m and m' as follows:

– if $m, m' \in M^{O_3^1}_{\neg\mathbf{Cl}}$ or $M^{O_4^1}_{\neg\mathbf{Cl}}$, $m\mathcal{R}'_4 m'$ iff $m = m'$
– if $m \in M^{O_3^1}_{\neg\mathbf{Cl}}$ and $m' \in M^{O_4^1}_{\neg\mathbf{Cl}}$, $m\mathcal{R}'_4 m'$ iff there is the equality on the common components of their states, knowing that the added lists L_A and L_B are completely determinated using the other common tables.

Then, we check the compatibility of $F_{\mathbf{Auth}} \cup F_{\mathbf{Auth'}}$ with $\mathcal{R}'_4$, i.e. that given two states $m \in M^{O_3^1}_{\neg\mathbf{Cl}}$ and $m' \in M^{O_4^1}_{\neg\mathbf{Cl}}$ in relation by $\mathcal{R}'_4$, $F_{\mathbf{Auth}}$ holds in state m iff $F_{\mathbf{Auth'}}$ holds in state m', which is obvious by the definition of the relation. Thus, applying the rule UpToBad, we find:

$$\frac{O_4^1 :_{\varepsilon'_6} F_{\mathbf{Auth'}}(\wedge G_{\neg\mathbf{Cl}}) \quad O_3^1 :_0 F_{\neg\mathbf{True}} \quad O_3^1 \overset{\neg\mathbf{Cl}}{\equiv}_{\mathcal{R}'_4,\mathbf{True}} O_4^1 \quad F_{\mathbf{Auth}}\mathcal{R}'_4 F_{\mathbf{Auth'}}}{O_3^1 :_{\varepsilon'_6} F_{\mathbf{Auth}}(\wedge G_{\neg\mathbf{Cl}})}\ \textit{UpToBad}$$

Sorting the password in the finalization oracle We a simplified game such that all the oracles are independent of pw. We modify the finalization oracle in order to draw the password only at the end of O_5^1.

$\mathrm{Imp}(o_F)(x) = x = pw$; return $\mathbf{1}$

The event $F_{\phi_{pw}} \circ \pi$ on O_5^1-traces is defined by $F_{\phi_{pw}} \circ \pi(\tau) = \mathbf{True}$ iff $\pi(\tau)$ verifies $F_{\phi_{pw}}$, where τ is any O_5^1-trace. Therefore, using the rule Fail, we get $O_5^1 :_{\varepsilon_1} F_{\phi_{pw}}$, where $\varepsilon'_9 = \frac{N_u+N_s}{N}$. Then, applying the rule B-Det-Left, we find:

$$\frac{O_3^1 \leq_{\mathrm{det},\gamma} O_5^1 \quad O_5^1 :_{\varepsilon'_9} (F_{\phi_{pw}} \circ \pi) \wedge G_{\neg\mathbf{Cl}}}{O_3^1 :_{\varepsilon'_9} F_{\phi_{pw}}(\wedge G_{\neg\mathbf{Cl}})}\ \textit{B-Det-Left}$$

such that $\varepsilon'_9 = \varepsilon'_7 + \varepsilon'_8 = \frac{N_u}{N} + \frac{N_s}{N}$. More precisely, we get $O_3^1 :_{\varepsilon'_7} F_{\phi_{pw1}}$ and $O_3^1 :_{\varepsilon'_8} F_{\phi_{pw2}}$.

To conclude, we obtain that $O_0^1 :_{\varepsilon_1} F_{\phi_{pw}}$ where $\varepsilon_1 = \frac{N_u+N_s}{N} + \frac{N_u+N_s}{q-1} + \frac{N_s q_E}{q-1} + 2q_H \times \varepsilon(\mathbf{1}_k, t)$. We perform the same analysis for the other game obtaining that $O_0^0 :_{\varepsilon_1} F_{\phi_{pw}}$.

For further, at each step, we suppose there is no dependence on the password when modifying the game O_0^1. We can introduce a particular equivalence relation under the step-predicate $\neg\phi_{pw}$ in order to avoid a query from the adversary with the good pw, since it steps in over memories using the list L_O. From that, E and D no longer give some evidence about the password to the adversary. This process enables to avoid the repetition of the value ε_1 at each transformation in the general proof.

Proof Tree: We illustrate the proof tree for bounding the probability of the step-predicate ϕ_{pw} on Figure (3). For convenience, we understand that each event $F_{\mathbf{Predicate}}$ is associated to the event $G_{\neg\mathbf{Cl}}$ and b is the bit randomly sampled in the initialization oracle.

N.B.: Defining the step-predicate ϕ_{pw} allows us to construct a proof which seems the more general possible. Indeed, we notice that it can be applied in another password-based protocol proof. From that, we hope to get security proofs more easily since we have already met the concept.

4.4 General proof for the indistinguishability between the games O_0^0 and O_0^1

Since the two conditions we described previously seem revelant, we transform the game O_0^1 in several steps under $G_{\neg\mathbf{Cl}} \wedge G_{\neg\phi_{pw}}$. The description of the general proof is less developed since we use the same transformations than for the proof for bounding the probability of ϕ_{pw}. Indeed, except the last game O_5^1 using the concept of determinization, we will apply in the same order each step using in the previous proof.

Removing the Encryption in the oracle S_1 In O_1^1, modified S_1 modelizes the Encryption inside (refer to page 7). If $Y^\star$ exists already then the exponent is equal to $\perp$. The step-predicate **Exp** defines this case (see pagerefexp).

Therefore, O_0^1 and O_1^1 are in bisimulation-up-to $\neg\mathbf{Exp}$, using as relation $\mathcal{R}_1$ the equality on the common components of their states in $M^{O_i^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$. Indeed, states m, m' are in relation:
– if $m, m' \in M^{O_0^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$ or $m, m' \in M^{O_1^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$, $m\mathcal{R}_1 m'$ iff $m = m'$
– if $m \in M^{O_0^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$, $m' \in M^{O_1^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$, $m\mathcal{R}_1 m'$ iff
  • $\forall (pw, x, y, e) \in m \cdot L_E \setminus m' \cdot L_E \Rightarrow e = \perp \wedge \exists (pw, x, y, \varphi) \in m' \cdot L_E \setminus m \cdot L_E$ s.t. $x = g^\varphi$

$$\textit{I-BisG2}\ \dfrac{O_0^b \overset{G_{\neg \mathbf{Cl}}}{\equiv}_{\mathcal{R}_1,\neg\mathbf{Exp}\wedge\neg\phi_{pw}} O_1^b \qquad O_1^b :_{\varepsilon'_2} F_{\mathbf{Exp}} \qquad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_1}{O_1^b :_{\varepsilon' 1} F_{\phi_{pw}}}}{O_0^b :_{\varepsilon'_1+\varepsilon'_2} F_{\phi_{pw}}}$$

Tree'$_1$:

$$\textit{I-BisG2}\ \dfrac{O_1^b \overset{G_{\neg \mathbf{Cl}}}{\equiv}_{\mathcal{R}_2,\neg\phi_{pw1}} O_2^b \quad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_2}{O_2^b :_{\varepsilon'_3} F_{\phi_{pw1}}} \quad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_3}{O_2^b :_{\varepsilon'_4} F_{\phi_{pw}} \wedge G_{\neg\phi_{pw1}}}}{O_1^b :_{\varepsilon'_1} F_{\phi_{pw}}}$$

Tree'$_2$:

$$\textit{I-BisG2}\ \dfrac{O_2^b \overset{G_{\neg \mathbf{Cl}}}{\equiv}_{\mathcal{R}_3,\neg\mathbf{Auth}\wedge\phi_{pw2}} O_3^b \quad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_4}{O_3^b :_{\varepsilon'_6} F_{\mathbf{Auth}\vee\phi_{pw2}}} \quad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_5}{O_3^b :_{\varepsilon'_7} F_{\phi_{pw1}} \wedge G_{\neg\mathbf{Auth}\wedge\neg\phi_{pw2}}}}{O_2^b :_{\varepsilon'_3} F_{\phi_{pw1}}}$$

Tree'$_3$:

$$\textit{I-BisG2}\ \dfrac{O_2^b \overset{G_{\neg \mathbf{Cl}}}{\equiv}_{\mathcal{R}_3,\neg\mathbf{Auth}\wedge\phi_{pw2}} O_3^b \quad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_4}{O_3^b :_{\varepsilon'_6} F_{\mathbf{Auth}} \wedge G_{\phi_{pw2}}} \quad \textit{I-BisG2}\ \dfrac{\mathbf{Tree'}_5}{O_3^b :_{\varepsilon'_8} F_{\phi_{pw2}}}}{O_2^b :_{\varepsilon'_4} F_{\phi_{pw}} \wedge G_{\neg\phi_{pw1}} = F_{\phi_{pw2}}}$$

Tree'$_4$:

$$\textit{Up-To-Bad}\ \dfrac{O_3^b \overset{G_{\neg \mathbf{Cl}}}{\equiv}_{\mathcal{R}_4,\mathbf{True}} O_4^b \quad O_3^b :_0 F_{\neg\mathbf{True}} \quad F_{\mathbf{Auth}} \mathcal{R}_4 F_{\mathbf{Auth'}} \quad \textit{B-Sub}\ \dfrac{CDH :_{\varepsilon(\mathbf{1}_k,t)} F_{\mathbf{Auth'}} \circ C}{O_3^b :_{\varepsilon'_6} F_{\mathbf{Auth'}}}}{O_3^b :_{\varepsilon'_6} F_{\mathbf{Auth}}}$$

Tree'$_5$:

$$\textit{B-Det-Left}\ \dfrac{O_3^b \leq_{\det,\gamma} O_5^b \qquad O_5^b :_{\varepsilon'_7} F_{\phi_{pw1}} \circ \pi}{O_3^b :_{\varepsilon'_7} F_{\phi_{pw1}}}$$

Tree'$_6$:

$$\textit{B-Det-Left}\ \dfrac{O_3^b \leq_{\det,\gamma} O_5^b \qquad O_5^b :_{\varepsilon'_8} F_{\phi_{pw2}} \circ \pi}{O_3^b :_{\varepsilon'_8} F_{\phi_{pw2}}}$$

Fig. 3. Proof Tree for the probability that the step-predicate ϕ_{pw} occurs

- $\forall (pw,x,y,e) \in m' \cdot L_E \setminus m \cdot L_E \Rightarrow e = \varphi$ s.t. $x = g^{\varphi} \wedge \exists (pw,x,y,\perp) \in m \cdot L_E \setminus m' \cdot L_E$

Hence, using the rule Fail, we get $O_1^1 :_{\varepsilon_2 = \frac{N_s \times q_E}{q-1}} F_{\mathbf{Exp}}$ and we apply the rule I-BisCd to result in:

$$\dfrac{O_1^1 :_{\varepsilon_2} F_{\mathbf{Exp}} (\wedge G_{\neg\mathbf{Cl}} \wedge G_{\neg\phi_{pw}}) \quad O_0^1 \overset{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}{\equiv}_{\mathcal{R}_1,\neg\mathbf{Exp}} O_1^1}{O_0^1 \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_2} O_1^1}\ \textit{I-BisCd}$$

Splitting the Hash lists In O_2^1, we split the lists of the hash functions. For that, we create two private hash functions H_2 and H_3 (refer to page 7).

O_1^1 and O_2^1 are $\mathcal{R}_2$-bismilar up to $\neg\phi_{pw1}$ (see page 6). We define the equivalence relation $\mathcal{R}_2$ between states m and m' as follows:

- if $m, m' \in M^{O_1^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$ or $m, m' \in M^{O_2^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$, $m\mathcal{R}_2 m'$ iff $m = m'$
- if $m \in M^{O_1^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$, $m' \in M^{O_2^1}_{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}$, $m\mathcal{R}_2 m'$ iff $m \cdot L_{H_0} = m' \cdot (L_{H_0} \cup L_{H_2}) \wedge m \cdot L_{H_1} = m' \cdot (L_{H_1} \cup L_{H_3})$

We obtain $O_2^1 :_0 F\phi_{pw1}$ since we consider the independence of the password in the oracles. Then, applying the rule I-BisCd, we find:

$$\frac{O_2^1 :_0 F_{\phi_{pw1}}(\wedge G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}) \quad O_1^1 \overset{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}{\equiv}_{\mathcal{R}_2, \neg \phi_{pw1}} O_2^1}{O_1^1 \overset{G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}}{\sim}_0 O_2^1} \text{ I-BisCd}$$

Randomizing the Hash oracles In O_3^1, sampling Y modifies the oracles U_2 and S_2 (refer to page 8).

Auth is defined page 8 and ϕ_{pw2} page 6. We notice that the event $F_{\phi_{pw2}}$ do not occur since we suppose that the adversary can not get the password. Using the equality $F_{\mathbf{Auth} \vee \phi_{pw2}} = F_{\mathbf{Auth}} \vee (F_{\phi_{pw2}} \wedge G_{\neg \mathbf{Auth}}) = F_{\mathbf{Auth}}$, we calculate the value ε_3 of the probability that the event $F_{\mathbf{Auth}}$ occurs.

We want to establish the indistinguishability between O_2^1 and O_3^1 up to $\neg \mathbf{Auth} \wedge \neg \phi_{pw2}$. We exhibit an equivalence relation $\mathcal{R}_3$ between both systems. Indeed, states m and m' are in relation:

- if $m, m' \in M^{O_2^1}_{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}$ or $m, m' \in M^{O_3^1}_{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}$, $m \mathcal{R}_3 m'$ iff $m = m'$
- if $m \in M^{O_2^1}_{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}$, $m' \in M^{O_3^1}_{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}$, $m \mathcal{R}_3 m'$ iff $m \cdot (L_{H_0} \cup L_{H_2}) = m' \cdot L_{H_2} \wedge m \cdot (L_{H_1} \cup L_{H_3}) = m' \cdot L_{H_3}$

Hence, we apply the rule I-BisCd to result in:

$$\frac{O_3^1 :_{\varepsilon_3} F_{\mathbf{Auth} \vee \phi_{pw2}}(\wedge G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}) \quad O_2^1 \overset{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}{\equiv}_{\mathcal{R}_3, \neg \mathbf{Auth} \wedge \neg \phi_{pw2}} O_3^1}{O_2^1 \overset{G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}}{\sim}_{\varepsilon_3} O_3^1} \text{ I-BisCd}$$

In the previous proof, we obtained that $O_3^1 :_{\varepsilon_6'} F_{\mathbf{Auth}}(\wedge G_{\neg \mathbf{Cl}})$. We use classic rule of Logic $O :_\varepsilon \mathbf{A} \Rightarrow O :_\varepsilon \mathbf{A} \wedge \mathbf{B}$ such that $\mathbf{A} = F_{\mathbf{Auth}}(\wedge G_{\neg \mathbf{Cl}})$ and $\mathbf{B} = G_{\neg \phi_{pw}}$. Therefore, we obtain that $O_3^1 :_{\varepsilon_6'} F_{\mathbf{Auth}}(\wedge G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}})$ where $\varepsilon_3 \leq \varepsilon_6'$.

4.5 Digest

Using four steps and the rule TrCd, we find $O_0^1 \overset{G_{\neg \mathbf{Cl}} \wedge \neg \phi_{pw}}{\sim}_{\varepsilon_2 + \varepsilon_3} O_3^1$. Similarly, we get $O_0^0 \overset{G_{\neg \mathbf{Cl}} \wedge \neg \phi_{pw}}{\sim}_{\varepsilon_2 + \varepsilon_3} O_3^0$.

To achieve the conclusion, we compare the games O_3^0 and O_3^1. At present, the adversary can not discern a random value from a real value for the session key sk. From that, he can not guess what was the bit sampled in the initialization oracle. Consequently, the latter discussion implies that the two last modified games O_3^0 and O_3^1 are in perfect bisimulation, with as a relation $\mathcal{R}_5$ the equality on the common components of their states. To conclude, we use the rule I-BisCd:

$$\frac{O_3^0 :_0 F_{\neg \mathbf{True}}(\wedge G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}) \quad O_3^1 :_0 F_{\neg \mathbf{True}}(\wedge G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}) \quad O_3^0 \overset{\neg \mathbf{Cl} \wedge \neg \phi_{pw}}{\equiv}_{\mathcal{R}_5, \mathbf{True}} O_3^1}{O_3^0 \overset{G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}}{\sim}_0 O_3^1} \text{ I-BisCd}$$

We use the rule TrCd to conclude to: $O_0^0 \overset{G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}}{\sim}_{2\varepsilon_2 + 2\varepsilon_3} O_0^1$. Having $O_0^b :_{\varepsilon_1} F_{\phi_{pw}}$ and using the rule FTr, we get: $O_0^0 \overset{G_{\neg \mathbf{Cl}}}{\sim}_{\varepsilon_1 + 2\varepsilon_2 + 2\varepsilon_3} O_0^1$. Since $O_0^b :_{\varepsilon_0} F_{\mathbf{Cl}}$, applying the rule FTr, we obtain: $O_0^0 \sim_{\varepsilon_0 + \varepsilon_1 + 2\varepsilon_2 + 2\varepsilon_3} O_0^1$, where $\varepsilon_0 + \varepsilon_1 + 2\varepsilon_2 + 2\varepsilon_3 = \frac{q_H^2}{2^{l+1}} + \frac{q_E^2}{2(q-1)} + \frac{N_u + N_s}{N} + \frac{N_u + N_s}{q-1} + \frac{N_s q_E}{q-1} + 2q_H \times \varepsilon(\mathbf{1}_k, t) + \frac{2 N_s q_E}{q-1} + 2q_H \times \varepsilon(\mathbf{1}_k, t)$.

General Proof Tree: We illustrate the proof tree on Figure (4). Most of the time, we use the rules I-BisCd and TrCd under the condition $G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}$. For convenience, we understand that each event $F_{\mathbf{Predicate}}$ is associated to the event $G_{\neg \mathbf{Cl}} \wedge G_{\neg \phi_{pw}}$ and b is the bit randomly sampled in the initialization oracle.

4.6 Conclusion

We gave a manual formal proof of the OEKE protocol, as the first application of the tool CIL. This proof is well contructed under two parts; The first proof seems complicated to find the probability of one-step predicate but stays clear. As this proof is similar to the general proof, therefore the latter is concise, precise and easy to understand. We obtained a new kind of security proof for OEKE based on general and extended logic rules, instead of "writing" proofs or "rewriting" proof using CryptoVerif.

$$\mathit{FTr}\ \frac{\mathit{FTr}\ \dfrac{\mathit{TrCd}\ \dfrac{\mathbf{Tree}_1}{O_0^0 \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{2\varepsilon_2+2\varepsilon_3+2\varepsilon_4} O_0^1} \quad \mathit{Fail}\ \dfrac{}{O_0^b :_{\varepsilon_1} F_{\phi_{pw}} \wedge G_{\neg\mathbf{Cl}}}}{O_0^0 \overset{G_{\neg\mathbf{Cl}}}{\sim}_{\varepsilon_1+2\varepsilon_2+2\varepsilon_3+2\varepsilon_4} O_0^1} \quad \mathit{Fail2}\ \dfrac{}{O_0^b :_{\varepsilon_0} F_{\mathbf{Cl}}}}{O_0^0 \sim_{\varepsilon_0+\varepsilon_1+2\varepsilon_2+2\varepsilon_3+2\varepsilon_4} O_0^1}$$

Tree$_1$:

$$\mathit{TrCd}\ \frac{\mathit{TrCd}\ \dfrac{\mathbf{Tree}_2}{O_0^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_2+\varepsilon_3+\varepsilon_4} O_3^b} \quad \mathit{I\text{-}BisCd}\ \dfrac{O_3^b :_0 F_{\neg\mathbf{True}} \qquad O_3^0 \overset{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}{\equiv}_{\mathcal{R}_5,\mathbf{True}} O_3^1}{O_3^0 \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_0 O_3^1}}{O_0^0 \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{2\varepsilon_2+2\varepsilon_3+2\varepsilon_4} O_0^1}$$

Tree$_2$:

$$\mathit{TrCd}\ \frac{\mathit{TrCd}\ \dfrac{\mathbf{Tree}_3}{O_0^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_2} O_2^b} \quad \mathit{I\text{-}BisCd}\ \dfrac{O_3^b :_{\varepsilon_3+\varepsilon_4} F_{\mathbf{Auth}\vee\phi_{pw2}} \qquad O_2^b \overset{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}{\equiv}_{\mathcal{R}_3,\neg\mathbf{Auth}\wedge\neg\phi_{pw2}} O_3^b}{O_2^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_3+\varepsilon_4} O_3^b}}{O_0^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_2+\varepsilon_3+\varepsilon_4} O_3^b}$$

Tree$_3$:

$$\mathit{TrCd}\ \frac{\mathit{I\text{-}BisCd}\ \dfrac{O_1^b :_{\varepsilon_2} F_{\mathbf{Exp}} \qquad O_0^b \overset{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}{\equiv}_{\mathcal{R}_1,\neg\mathbf{Exp}} O_1^b}{O_0^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_2} O_1^b} \quad \mathit{I\text{-}BisCd}\ \dfrac{O_2^b :_0 F_{\phi_{pw}} \qquad O_1^b \overset{\neg\mathbf{Cl}\wedge\neg\phi_{pw}}{\equiv}_{\mathcal{R}_2,\neg\phi_{pw}} O_2^b}{O_1^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_0 O_2^b}}{O_0^b \overset{G_{\neg\mathbf{Cl}}\wedge G_{\neg\phi_{pw}}}{\sim}_{\varepsilon_2} O_2^b}$$

Fig. 4. Proof Tree for OEKE

Theorem 1. *Let us consider the OEKE protocol, where Password is a finite dictionnary of size N equipped with the uniform distribution. Let A be a (k,t)-adversary against the security of OEKE within a time bound t, with less than $N_u + N_s$ interactions with the parties and asking q_H hash queries and q_E Encryption/Decryption queries. Then we have:*

$$Adv_{oeke}(A) \leq \frac{N_u+N_s}{N} + \frac{N_u+N_s}{q-1} + \frac{q_E^2}{2(q-1)} + \frac{3N_s q_E}{q-1} + \frac{q_H^2}{2^{l+1}} + 4q_H \times \varepsilon(\mathbf{1}_k, t)$$

We stayed careful of putting realistic hypothesis for elements of the proof, as for functions in ROM and ICM. We obtained the optimal term $\frac{N_u+N_s}{N}$.

N.B.: In 2003, the autors of the paper [3] recognized that their results of the reductions proof were not optimal. For technical reasons, they used a collision-resistant hash function H_1. After we began our article, in the paper [9], they proved the security of OEKE using the tool CryptoVerif. The boundary was improved relative to the former proof since they reached the optimal result $\frac{N_u+N_s}{N}$. As in these papers, we obtained the optimal term but using a new kind of analysis under CIL.

Moreover, the logic CIL is sufficiently developed: it can be used easily and efficiently to construct computational proofs.

References

1. Steven M. BELLOVIN, Michael MERRITT, *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionnary Attacks*. In Proc. IEEE Computer Society Symposium on Research in Security and Privacy, pages 72-84, 1992.
2. Mihir BELLARE, Philip ROGAWAY, *The AuthA Protocol for Password-Based Authenticated Key Exchange*. Unpublished contribution to IEEE P1363, 2000.

3. Emmanuel BRESSON, Olivier CHEVASSUT, David POINTCHEVAL, *Security Proofs for an Efficient Password-Based Key Exchange*. In Proceedings of CCS'03, pages 241-250. ACM Press, 2003.
4. Mihir BELLARE, Phillip ROGAWAY, *Code-Based Game-Playing Proofs and the Security of Triple Encryption*. In EUROCRYPT 2006, LNCS volume 4004, 2006.
5. Victor SHOUP, *Sequences of games: A Tool for Taming Complexity in Security Proofs*. manuscript, 2004.
6. Shai HALEVI, *A plausible approach to computer-aid cryptographic proofs*. manuscript, 2005.
7. Gilles BARTHE, Marion DAUBIGNARD, Bruce KAPRON, Yassine LAKHNECH, *Computational Indistinguishability Logic*. In Proceedings of CCS'10, pages 375-386. ACM Press, 2010.
8. Marion DAUBIGNARD, *Formal Methods for Concrete Security Proofs*. PhD thesis, 2012.
9. Bruno BLANCHET, *Automatically Verified Mechanized Proof of One-Encryption Key Exchange*. In CSF'12, 2012.

A Oracle systems

A.1 Oracle systems and adversaries

An oracle system is a stateful system that provides oracle access to adversaries.

Definition 1. *An oracle system O is given by:*

- *sets M_o of oracle memories and N_o of oracles,*
- *for each $o \in N_o$, a query domain $In(o)$, an answer domain $Out(o)$ and an implementation $O_o : In(o) \times M_o \to D(Out(o) \times M_0)$,*
- *a distinguished initial memory $\bar{m_o} \in M_o$, and distinguished oracles o_I for initialization and o_F for finalization, such that $In(o_I) = Out(o_F) = 1$. We let $Res = In(o_F)$.*

Two oracle systems O and O' are *compatible* iff they have the same sets of oracle names, and the query and the answer domains of each oracle name coincide in both oracle systems. When building a compatible oracle system from another one, it is thus sufficient to provide its set of memories, its initial memory and the implementation of its oracles.

Adversaries interact with oracle systems by making queries and receiving answers. An exchange for an oracle system O is a triple (o,q,a) where $o \in N_o$, $q \in In(o)$ and $a \in Out(o)$. We let Xch be the set of exchanges. Initial and final exchanges are defined in the obvious way, by requiring that o is an initialization and finalization oracle respectively (the sets of these exchanges are denoted by Xch_I and Xch_F respectively). The sets Que of queries and Ans of answers are respectively defined as $\{(o,q) \mid (o,q,a) \in Xch\}$ and $\{(o,a) \mid (o,q,a) \in Xch\}$.

Definition 2. *An adversary A for an oracle system O is given by a set M_a of adversary memories, an initial memory $\bar{m_a} \in M_a$ and functions for querying and updating $A : M_a \to D(Que \times M_a)$ and $A_{\downarrow} : Xch \times M_a \to D(m_a)$.*

Informally, the interaction between an oracle system and an adversary proceeds in three successive phases: the initialization oracle sets the initial memory distributions of the oracle system and of the adversary. Then, A performs computations, updates its state and submits queries to O. In turn, O performs computations, updates its state, and replies to A, which updates its state. Finally, A outputs a result by calling the finalization oracle.

A.2 Semantics

Definition 3. *A transition system S consists of:*

- *a (countable non-empty) set M of memories (states) with a distinguished initial memory $\bar{m}$,*
- *a set $\sum$ of actions with distinguished subsets of $\sum_I$ and $\sum_F$ of initialization and finalization actions,*
- *a (partial) transition function $step : M \rightharpoonup D(\sum \times M)$.*

A partial execution sequence of S is a sequence of ζ of the form $m_0 \xrightarrow{x_1} m_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} m_k$ such that $Pr[\text{step}(m_{k-1}) = (a_k, m_k)] > 0$ for $i = 1..k$ and $x_i = (o_i, q_i, a_i)$. If $k = 1$ then ζ is a step. If $m_0 = \bar{m}$, $x_1 \in \sum_I$ and $x_k \in \sum_F$ then ζ is an execution sequence of length k. A probabilistic transition system S induces a sub-distribution on executions, denoted S, such that the probability of a finite execution sequence ζ is $Pr[S = \zeta] = \prod_{i=1}^{k} Pr[\text{step}(m_{i-1}) = (a_i, m_i)]$. A transition system is of height $k \in \mathbb{N}$ if all its executions have length at most k: in this case, S is a distribution.

Definition 4. *Let O be an oracle system and A be an O-adversary. The composition $A \mid O$ is a transition system such that $M = M_a \times M_o$, the initial memory is $(\bar{m_a}, \bar{m_o})$, the set of actions is $\sum = Xch$, $\sum_I = Xch_I$ and $\sum_F = Xch_F$, and*

$$step_{A|O}(m_a, m_o) = ((o,q), m'_a) \leftarrow A(m_a);\ (a, m'_o) \leftarrow O_o(q, m_o)\ ; m''_a \leftarrow A_{\downarrow}((o,q,a), m'_a);\ return\ ((o,q,a),(m''_a, m'_o))$$

An adversary is called k-bounded if $A \mid O$ is of height k. This means that A calls the finalization oracle after less than k interactions with O. $A \mid O$ may be ill-defined for unbounded adversaries, since $\text{step}_{A|O}(m_a, m_o)$ may be a sub-distribution. Throughout the paper, we only consider bounded adversaries, i.e. that are k-bounded for some k.

A.3 Events

Security properties abstract away from the state of adversaries and are modeled using traces. Informally, a trace τ is an execution sequence η from which the adversary memories have been erased.

Definition 5. *Let O be an oracle system.*

- *A partial trace is a sequence τ of the form $m_0 \xrightarrow{x_1} m_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} m_k$ where $m_0..m_k \in M_o$ and $x_1..x_k \in Xch$ such that $Pr[O_{o_i}(q_i, m_{i-1}) = (a_i, m_i)] > 0$ for $i = 1..k$ and $x_i = (o_i, q_i, a_i)$. A trace is a partial trace τ such that $m_0 = \bar{m_o}$, $x_1 = (o_I, \_, \_)$ and $x_k = (o_F, \_, \_)$.*
- *An O-event $\mathbf{E}$ is a predicate over O-traces, whereas an extended O-event $\mathbf{E}$ is a predicate over partial O-traces.*

The probability of an (extended) event is derived directly from the definition of $A \mid O$: since each execution sequence η induces a trace $\mathcal{T}(\eta)$ simply by erasing the adversary memory at each step, one can define for each trace τ, the set $\mathcal{T}^{-1}(\tau)$ of execution sequences that are erased to τ, and for every (generalized) event $\mathbf{E}$, the probability: $Pr[A \mid O : \mathbf{E}] = Pr[A \mid O : \mathcal{T}^{-1}(\mathbf{E})] = \sum_{\{\eta \in Exec(A|O) | \mathbf{E}(\mathcal{T}(\eta)) = \mathbf{True}\}} Pr[A \mid O : \eta]$.

Constructions and proofs in CIL use several common operations on (extended) events and traces. First, one can define the conjunction, the disjunction, etc, of events. Moreover, one can define for every predicate P over $Xch \times M_o \times M_o$ the events "eventually P" F_P and "always P" G_P that correspond to P being satisfied by one step and all steps of the trace respectively.

Reduction-based arguments require that adversaries can partially simulate behaviors. In some cases, adversaries must test whether a predicate $\varphi \subseteq Xch \times M_o \times M_o$ holds for given values. Since the adversary has no access to the oracle memory, we say that φ is *testable* iff for all x, m_1, m'_1, m_2, m'_2, we have $\varphi(x, m_1, m'_1)$ iff $\varphi(x, m_2, m'_2)$ (that is φ depends only on the exchange).

Given two traces τ and τ', we write $\tau \mathcal{R} \tau'$ iff for every $i \in [1,k]$, we have $m_i \mathcal{R} m'_i$, where: $\tau = m_0 \xrightarrow{x_1} m_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} m_k$ and $\tau' = m'_0 \xrightarrow{x_1} m'_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} m'_k$.

Moreover, we say that two events $\mathbf{E}$ and $\mathbf{E'}$ are $\mathcal{R}$-compatible, written $\mathbf{E}\mathcal{R}\mathbf{E'}$, iff $\mathbf{E}(\tau)$ is equivalent to $\mathbf{E'}(\tau')$ for every traces τ and τ' such that $\tau \mathcal{R} \tau'$.

B Computational Indistinguishability Logic

B.1 Statements and Rules

As cryptographic proofs rely on assumptions, CIL manipulates sequents of the form $\Delta \Rightarrow \omega$, where Δ is a set of statements (the assumptions) and ω is a statement (the conclusion). Validity extends to sequents $\Delta \Rightarrow \omega$ in the usual manner. Given a set Δ of statements, $\models \Delta$ iff $\models \psi$ for every $\psi \in \Delta$. Then $\Delta \models \omega$ iff $\models \Delta$ implies $\models \omega$. For clarity and brevity, our presentation of CIL omits hypotheses and the standard structural and logical rules for sequent calculi.

Theorem 2. *Every sequent $\Delta \Rightarrow \varphi$ provable in CIL is also valid, i.e. $\Delta \models \varphi$.*

Judgments CIL considers negligibility statements of the form $O :_\varphi \mathbf{E}$, where $\mathbf{E}$ is an event. A statement $O :_\varphi \mathbf{E}$ is valid, written $\models O :_\varphi \mathbf{E}$, iff for every (k,t)-adversary A, $Pr(A \mid O : \mathbf{E}) \leq \varepsilon(k,t)$.

We also consider indistinguishability statements of the form $O \sim_\varepsilon O'$, where O and O' are compatible oracle systems which expect a boolean as result. A statement $O \sim_\varepsilon O'$ is valid, written $\models O \sim_\varepsilon O'$, iff for every (k,t)-adversary A,

$$|\ Pr[A \mid O : R = \mathbf{True}] - Pr[A \mid O' : R = \mathbf{True}]\ | \leq \varepsilon(k,t)$$

where $R = \mathbf{True}$ is shorthand for $F_{\lambda(o,q,a).\, o=o_F \wedge q=\mathbf{True}}$.

Therefore, we formalize the indistinguishability of distributions yielded by systems under condition, the latter being written as an event of systems. Let $\mathbf{E}$ be an event of O and O'. A statement $O \overset{\mathbf{E}}{\sim}_\varepsilon O'$ is valid, written $\models O \overset{\mathbf{E}}{\sim}_\varepsilon O'$, iff for every (k,t)-adversary A,

$$| \, Pr[A \mid O : R = \mathbf{True} \wedge \mathbf{E}] - Pr[A \mid O' : R = \mathbf{True} \wedge \mathbf{E}] \, | \leq \varepsilon(k,t)$$

As cryptographic proofs rely on assumptions, CIL manipulates sequents of the form $\Delta \Rightarrow \omega$, where Δ is a set of statements (the assumptions) and ω is a statement (the conclusion). Validity extends to sequents $\Delta \Rightarrow \omega$ in the usual manner. Given a set Δ of statements, $\models \Delta$ iff $\models \psi$ for every $\psi \in \Delta$. Then $\Delta \models \omega$ iff $\models \Delta$ implies $\models \omega$.

Rules On Figures (5), (6) and (7), we expose rules that support equational reasoning and consequence in Hoare logic, rules that were extended rules found during the conception of the proofs in this article, and rules that are used mainly in the proofs in this article. Let O, O' and O" be compatible oracle systems, $\mathbf{E}$, $\mathbf{E}_1$ and $\mathbf{E}_2$ be events of O, O' and O", and φ, φ_1 and φ_2 be step-predicates.

$$\frac{O \sim_{\varepsilon_i} \mathbf{E}_i (i \in I) \qquad \mathbf{E} \Rightarrow \bigvee_{i \in I} \mathbf{E}_i}{O :_{\sum_{i \in I} \varepsilon_i} \mathbf{E}} \; \mathit{UR} \qquad \frac{}{O :_\varepsilon F_\varphi} \; \mathit{Fail} \qquad \frac{O :_\varepsilon F_{\neg\varphi} \qquad O \equiv_{\mathcal{R},\varphi} O'}{O \sim_\varepsilon O'} \; \mathit{I\text{-}Bis}$$

$$\frac{O \leq_{\det,\gamma} O' \qquad O :_\varepsilon \mathbf{E} \circ \pi}{O' :_\varepsilon \mathbf{E}} \; \mathit{B\text{-}Det\text{-}Left} \quad \frac{O :_\varepsilon \mathbf{E} \circ C}{C[O] :_{\varepsilon'} \mathbf{E}} \; \mathit{B\text{-}Sub} \quad \frac{O :_\varepsilon \mathbf{E}_1 \wedge G_\varphi \quad O \equiv_{\mathcal{R},\varphi} O' \quad \mathbf{E}_1 \mathcal{R} \mathbf{E}_2}{O' :_\varepsilon \mathbf{E}_2 \wedge G_\varphi} \; \mathit{B\text{-}BisG}$$

$$\frac{O :_{\varepsilon_2} \mathbf{E}_2 \quad O' :_{\varepsilon_1} F_{\neg\varphi} \quad O \equiv_{\mathcal{R},\varphi} O' \quad \mathbf{E}_1 \mathcal{R} \mathbf{E}_2}{O' :_{\varepsilon_1 + \varepsilon_2} \mathbf{E}_1} \; \mathit{UpToBad}$$

Fig. 5. Classic rules

$$\frac{O \overset{\mathbf{E}_2}{\sim}_{\varepsilon_1} O' \qquad \mathbf{E}_2 \Rightarrow \mathbf{E}_1 \qquad O :_{\varepsilon_2} \mathbf{E}_1 \wedge \neg \mathbf{E}_2 \qquad O' :_{\varepsilon_2} \mathbf{E}_1 \wedge \neg \mathbf{E}_2}{O \overset{\mathbf{E}_1}{\sim}_{\varepsilon_1 + \varepsilon_2} O'} \; \mathit{URCd} \quad \frac{}{O :_{\varepsilon'} F_{\varphi'}} \; \mathit{Fail2} \quad \frac{O \overset{\mathbf{E}_1}{\sim}_{\varepsilon_1} O' \qquad O' \overset{\mathbf{E}_2}{\sim}_{\varepsilon_2} O"}{O \overset{\mathbf{E}_1 \vee \mathbf{E}_2}{\sim}_{\varepsilon_1 + \varepsilon_2} O"} \; \mathit{TrCd}$$

Fig. 6. Extended rules

$$\frac{O :_{\varepsilon_1} F_{\varphi_1} \wedge G_{\varphi_2} \quad O :_{\varepsilon_2} F_{\neg\varphi_2} \quad O \equiv_{\mathcal{R},\varphi_2} O'}{O' :_{\varepsilon_1 + \varepsilon_2} F_{\varphi_1}} \; \mathit{B\text{-}BisG2} \quad \frac{O' :_\varepsilon F_{\neg\varphi_2} \wedge G_{\varphi_1} \qquad O \overset{\varphi_1}{\equiv}_{\mathcal{R},\varphi_2} O'}{O \overset{G_{\varphi_1}}{\sim}_\varepsilon O'} \; \mathit{I\text{-}BisCd}$$

$$\frac{O \overset{\mathbf{E}_1 \wedge \mathbf{E}_2}{\sim}_{\varepsilon_2} O' \qquad O :_{\varepsilon_1} \neg \mathbf{E}_1 \wedge \mathbf{E}_2 \qquad O' :_{\varepsilon_1} \neg \mathbf{E}_1 \wedge \mathbf{E}_2}{O \overset{\mathbf{E}_2}{\sim}_{\varepsilon_1 + \varepsilon_2} O'} \; \mathit{FTr}$$

Fig. 7. Rules used in the proof (extended rules)

More precisely, CIL features a rule to compute an upper-bound on the probability of an event from the number of oracle calls, and from the probability that a single oracle call triggers that event. Let φ be a predicate on $Xch \times$

$M_o \times M_o$ and define, for every $o \in N_o$, the probability ε_o as $\max_{\substack{q \in Que, m \in M_o, \\ a \in Ans, m' \in M_o}} Pr[O_o(q,m) = (a,m') \wedge \varphi((o,q,a),m,m')]$.

For every $o \in N_o$, let k_o be the maximal number of queries to o and let $\varepsilon = \sum_{o \in N_o} k_o \varepsilon_o$. CIL features the rule $\dfrac{}{O :_\varepsilon F_\varphi}$ *Fail*. But sometimes, this upper-bound is not enough convenient for the proof. We introduce another rule which keeps all the previous oracles calls triggerring the event when considering a single oracle call. CIL features the rule $\dfrac{}{O :_{\varepsilon'} F_\varphi}$ *Fail2*, where $\varepsilon' = \varepsilon \times \frac{(\sum_{o \in I} k_o)^2}{2}$ such that:

- k_o is the maximal number of queries of the oracle o and n is the cardinal of the set N_o
- I is the family of oracles that can ensure that the step-predicate φ can be satisfied: o can be an oracle in $N_o \setminus I$ such that $\varepsilon_o(k_{o_1}, \cdots, k_{o_n}) = 0$ or an oracle in I such that $\exists \varepsilon$, $\varepsilon_o(k_{o_1}, \cdots, k_{o_n}) = \varepsilon \times \sum_{o' \in I} k_{o'}$

B.2 Contexts

Informally, a context C is an intermediary between an oracle system O and adversaries. One can compose a O-context C with O to obtain a new oracle system $C[O]$ and with a $C[O]$-adversary to obtain a new O-adversary $C \parallel A$. Moreover, one can show that the systems $C \parallel A \mid O$ and $A \mid C[O]$ coincide in a precise mathematical sense. Despite its seemingly naivety, the relationship captures many reduction arguments used in cryptographic proofs and yields CIL rules that allow proving many schemes.

The definition of contexts is very similar to that of oracle systems, except that procedures are implemented by two functions: one that transfers calls from the adversary to the oracles and another one that tranfers answers from the oracles to the adversary (possibly after some computations).

Definition 6. *An O-context C is given by:*

- *sets M_c of context memories, an initial memory $\bar{m}_c$ and N_c of procedures*
- *for every $c \in N_c$, a query domain $In(c)$, an answer domain $Out(c)$ and two functions $C_{\overrightarrow{c}} : In(c) \times M_c \to D(Que \times M_c)$ and $C_{\overleftarrow{c}} : In(c) \times Xch \times M_c \to D(Out(c) \times M_c)$.*
- *distinguished initialization and finalization procedures c_I and c_F such that $In(c_I) = Out(c_F) = 1$, and for all x and m_c, $range(C_{\overrightarrow{c_I}}(x, m_c))(\lambda((o, \_), \_).\, o = o_I)$ and $range(C_{\overrightarrow{c_F}}(x, m_c))(\lambda((o, \_), \_).\, o = o_F)$. We let $Res_c = In(c_F)$.*

An indistinguishability context is an O-context C such that $Res_c = Res$ and $C_{\overrightarrow{c_F}}(r, m) = \delta_{((r, o_F), m)}$ for all r and m.

The sets Que_c of context queries, Ans_c of context answers and Xch_c of context exchanges are defined similarly to oracle systems. An O-context can be composed with the oracle system O or with any O-adversary A, yielding a new oracle system $C[O]$ or a new adversary $C \parallel A$. We begin by defining the composition of a context and an oracle system.

Definition 7. *The application of an O-context C to O defines an oracle system $C[O]$ such that:*

- *the set of memories is $M_c \times M_o$ and the initial memory is $(\bar{m}_c, \bar{m}_o)$*
- *the oracles are the procedures of C and their query and answer domains are given by C. The initialization and finalization oracles are the initialization and finalization procedures of C*
- *the implementation of an oracle c is:*
  $\lambda(q_c, (m_c, m_o)).\ ((o, q_o), m'_c) \leftarrow C_{\overrightarrow{c}}(q_c, m_c)\ ;\ (a_o, m'_o) \leftarrow O_o(q_o, m_o);\ (a_c, m''_c) \leftarrow C_{\overleftarrow{c}}(q_c, (o, q_o, a_o), m'_c)\ ;$
  $return\ (a_c, (m''_c, m'_o))$
  where $\cdot \leftarrow \cdot$ notation is used for monadic composition and "return" is used for returning the result of the function.

The composition of an adversary with a context is slightly more subtle and requires that the new adversary stores the current query in its state.

Definition 8. *The application of an O-context C to a $C[O]$-adversary A defines an O-adversary $C \parallel A$ such that:*

- *the set of memories is $M_c \times M_a \times Que_c$ and the initial memory is $(\bar{m}_c, \bar{m}_a, \_)$*
- *the transition function is:*
  $\lambda(m_c, m_a, \_).\ ((c, q_c), m'_a) \leftarrow A(m_a)\ ;\ ((o, q), m'_c) \leftarrow C_{\overrightarrow{c}}(q_c, m_c);\ return\ ((o, q), (m'_c, m'_a, (o, q)))$
- *the update function is:*
  $\lambda((m_c, m_a, (o_c, q_c)), (o_o, q_o, a_o)).\ (a_c, m'_c) \leftarrow C_{\overleftarrow{c}}(q_c, (o_o, q_o, a_o), m_c)\ ;\ return\ (m'_c, A_{\downarrow}((o_c, q_c, a_c), m_a), \_)$

Context CDH used in the proofs

CDH assumption in G

Let $G = \langle g \rangle$ be a finite cyclic group of order a l-bit prime number q, where the operation is denoted multiplicatively. We give an oracle system CDH such that:

- the memories map the variable g to the values in G and the variables α and β to the values $[1..(q-1)]$;
- for one such variable g, the initialization oracle draws uniformly at random values for α and β and outputs (g^α, g^β);
- the finalization oracle takes as input an element of G (in addition to a memory).

Bounding the number of calls of the adversary to the oracles is irrevelant. Let $\mathbf{1}_k$ be the function mapping o_I and o_F to $\mathbf{1}$. Given a negligible function ε, the $\varepsilon - CDH$ assumption holds for the group G iff for all $(\mathbf{1}_k, t)$-adversary, we have $\varepsilon - CDH \vdash$ oracle $CDH :_{\varepsilon(\mathbf{1}_k, t)} R = \mathbf{1}$.

Notation: Given g, $x \leftarrow \mathbb{Z}_q^*$ and $y \leftarrow \mathbb{Z}_q^*$, let $CDH(g^x, g^y) = g^{xy}$.

Formalization of CDH assumption: We define an oracle system CDH to capture the game played by an adversary to find the Diffie-Hellman instance (A, B). We implement this oracle as follows:

$\mathrm{Imp}_{CDH}(o_I)(g) =$
    $\alpha_0 \leftarrow \mathbb{Z}_q^*$ $\beta_0 \leftarrow \mathbb{Z}_q^*$; $A := g^{\alpha_0}$; $B := g^{\beta_0}$;
    return (A, B)

$\mathrm{Imp}_{CDH}(o_F)(x) =$
    if $x = CDH(A, B)$ then return $\mathbf{1}$
    else return $\mathbf{0}$
    endif

Context of CDH assumption

For this part, we write the game O_4^1 as a context C of CDH. We simulate the oracles using the random self-reducibility of the Diffie-Hellman problem, given one CDH instance (A, B).

$C_{\overrightarrow{c_I}}(x)$:
    return $(o_I, \mathbf{1})$

$C_{\overleftarrow{c_I}}(x, (o, q, (A, B)))$:
    $pw \leftarrow Password$
    $L_{H_0} := []$; $L_{H_1} := []$; $L_{H_2} := []$; $L_{H_3} := []$; $L_E := []$; $L_{pw} := []$; $L_O := []$;
    $L_A := []$; $L_B := []$; $var_X := \perp$; $var_\theta := \perp$; $var_\varphi := \perp$; $var_{sk} := \perp$;
    $b := 1$
    return $\mathbf{1}$

$C_{\overrightarrow{E}}(pw, x)$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{E}}((pw, x), (o, q, a))$:
    if $(pw, x, \_, \_) \notin L_E$ then $y \leftarrow \bar{G}$; $L_E := L_E.(pw, x, y, \perp)$ endif
    return y such that $(pw, x, y, \_) \in L_E$

$C_{\overrightarrow{D}}(pw, y)$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{D}}((pw, y), (o, q, B))$:
    if $(pw, \_, y, \_) \notin L_E$ then $\phi \leftarrow \mathbb{Z}_q^*$; $x = g^\phi$; $L_E := L_E.(pw, x, y, \phi)$ endif
    return x such that $(pw, x, y, \_) \in L_E$

$C_{\overrightarrow{H_0}}(x)$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{H_0}}(x, (o, q, a))$:
    if $x \notin L_{H_0}$ then $y \leftarrow U(l_0)$; $L_{H_0} := L_{H_0}.(x, y)$ endif
    return $L_{H_0}(x)$

$C_{\overrightarrow{H_1}}(x)$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{H_1}}(x, (o, q, a))$:
    if $x \notin L_{H_1}$ then $y \leftarrow U(l_1)$; $L_{H_1} := L_{H_1}.(x, y)$ endif
    return $L_{H_1}(x)$

$C_{\overrightarrow{U_1}}(u, i)$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{U_1}}((u, i), (o, q, A))$:
    $\alpha \leftarrow \mathbb{Z}_q^*$; $X = A^\alpha$; $var_\theta[(u, i)] = (\alpha, X)$; $var_X[(u, i)] = X$; $L_A := L_A.(\alpha, X)$
    return (u, X)

$C_{\overrightarrow{S_1}}((s, j), (u, X))$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{S_1}}((s, j), (u, X), (o, q, B))$:
    $Y^\star \leftarrow \bar{G}$; $\beta \leftarrow \mathbb{Z}_q^*$; $Y = B^\beta$; $var_\varphi[(s, j)] = (\beta, Y, Y^\star)$; $L_B := L_B.(\beta, Y)$; $var_X[(s, j)] = X$
    return $(s, Y^\star)$

$C_{\overrightarrow{U_2}}((u, i), (s, Y^\star))$:
    return $(\perp, \mathbf{1})$

$C_{\overleftarrow{U_2}}((u, i), (s, Y^\star), (o, q, a))$:
    if $var_\theta[(u, i)] != \perp$ then $Y \leftarrow \bar{G}$; $(\_, Y, Y^\star) = var_\varphi[(u, i)]$;
        $(\alpha, X) = var_\theta[(u, i)]$; $K_u = Y^\alpha$
        $Auth = H_1(u \parallel s \parallel X \parallel Y \parallel K_u)$; $var_{sk}[(u, i)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_u)$
    endif
    return $Auth$

$C_{\overrightarrow{S_2}}((s,j),u,Auth)$:
  return $(\perp,\mathbf{1})$

$C_{\overleftarrow{S_2}}((s,j),u,Auth,(o,q,B))$:
  if $var_\varphi[(s,j)]! = \perp$ then $(\beta,Y,Y^\star) = var_\varphi[(s,j)]$; $X = var_X[(s,j)]$; $K_s = X^\beta$
    $H' = H_1(u \parallel s \parallel X \parallel Y \parallel K_s)$
    if $H' = Auth$ then $var_{sk}[(s,j)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_s)$ endif
  endif
  return $\mathbf{1}$

$C_{\overrightarrow{Exec}}((u,i),(s,j))$:
  return $(\perp,\mathbf{1})$

$C_{\overleftarrow{Exec}}((u,i),(s,j),(o,q,(A,B)))$:
  $\alpha \leftarrow \mathbb{Z}_q^*$; $X = A^\alpha$; $\beta \leftarrow \mathbb{Z}_q^*$; $Y = B^\beta$; $Y^\star = E(pw,Y)$
  $Auth = H_1(u \parallel s \parallel X \parallel Y \parallel K_u)$; $var_{sk}[(u,i)] = H_0(u \parallel s \parallel X \parallel Y \parallel K_u)$
  return $((u,X),(s,Y^\star),Auth)$

$C_{\overrightarrow{Reveal}}(p,k)$:
  return $(\perp,\mathbf{1})$

$C_{\overleftarrow{Reveal}}((p,k),(o,q,a))$:
  if $var_{sk}[(p,k)]! = \perp$ then $sk := var_{sk}[(p,k)]$ endif
  return sk

$C_{\overrightarrow{Test^1}}(p,k)$:
  return $(\perp,\mathbf{1})$

$C_{\overleftarrow{Test^1}}((p,k),(o,q,a))$:
  if $var_{sk}[(p,k)]! = \perp$ then $sk := var_{sk}[(p,k)]$ endif
  return sk

$C_{\overrightarrow{cF}}(x)$:
  $u \parallel s \parallel X \parallel Y \parallel K \leftarrow L_{H_1}$
  if $X = (A,\alpha) \wedge Y = (B,\beta)$ then $o_F(K^{\alpha^{-1}\beta^{-1}})$ endif
  return $\mathbf{1}$

B.3 Bisimulation

Game-based proofs often proceed by transforming an oracle system into an equivalent one, or in case of imperfect simulation into a system that is equivalent up to some bad event. We reason in terms of probabilistic transition systems, using a mild extension of the standard notion of bisimulation. More specifically, we define the notion of bisimulation-up-to, where two probabilistic transition systems are bisimilar until the failure of a condition on their transitions. The definition of bisimulation is recovered by considering bisimulation-up-to the constant predicate **True**.

Let O and O' be two compatible oracle systems. For every oracle name, we let $\hat{M}$ be $M_o + M'_o$ and for every $o \in N_o$, we let $\hat{O}_o$ be the disjoint sum of O_o and O'_o, i.e. $\hat{O}_o : In(o) \times \hat{M} \to D(Out(o) \times \hat{M})$. We write $m \xrightarrow{(x,y)}_{>0} m'$ iff $Pr[\hat{O}_o(q,m_i) = (a,m'_i)] > 0$.

Definition 9. *Let $\varphi \subseteq Xch \times \hat{M} \times \hat{M}$ and let $\mathcal{R} \subseteq \hat{M} \times \hat{M}$ be an equivalence relation. O and O' are bisimilar-up-to φ, written $O \equiv_{\mathcal{R},\varphi} O'$, iff $\bar{m}\mathcal{R}\bar{m}'$, and for all $m_1 \xrightarrow{(o,q,a)}_{>0} m'_1$ and $m_2 \xrightarrow{(o,q,a)}_{>0} m'_2$ such that $m_1\mathcal{R}m_2$:*

- *Stability: if $m'_1\mathcal{R}m'_2$ then $\varphi((o,q,a),m_1,m'_1) \Leftrightarrow \varphi((o,q,a),m_2,m'_2)$;*
- *Compatibility: if $\varphi((o,q,a),m_1,m'_1)$ then $Pr[\hat{O}_o(q,m_1) \in (a,C)] = Pr[\hat{O}_o(q,m_2) \in (a,C)]$ where C is the equivalence class of m'_1 under $\mathcal{R}$.*

Bisimulations are closely related to obversational equivalence and relational Hoare logic, and allow to justify proofs by simulations. Besides, bisimulations-up-to subsume the Fundamental Lemma of Victor Shoup. Then, we introduce an extension of this concept, taking account of a particular equivalence relation included in a more restricted set of memories.

Definition 10. *Let $\varphi' \subseteq \hat{M}$ and let $\hat{M}_{\varphi'} = \{m \in \hat{M} \mid \varphi'(m)\}$. Let $\varphi \subseteq Xch \times \hat{M} \times \hat{M}$ and let $\mathcal{R} \subseteq \hat{M}_{\varphi'} \times \hat{M}_{\varphi'}$ be an equivalence relation.*

O and O' are bisimilar-up-to φ, written $O \overset{\varphi'}{\equiv}_{\mathcal{R},\varphi} O'$, iff for all $\bar{m},\bar{m}',m_1,m_2,m'_1,m'_2$ in $\hat{M}_{\varphi'}$ such that $\bar{m}\mathcal{R}\bar{m}'$, and for $m_1 \xrightarrow{(o,q,a)}_{>0} m'_1$ and $m_2 \xrightarrow{(o,q,a)}_{>0} m'_2$ such that $m_1\mathcal{R}m_2$:

- *Stability: if $m'_1\mathcal{R}m'_2$ then $\varphi((o,q,a),m_1,m'_1) \Leftrightarrow \varphi((o,q,a),m_2,m'_2)$;*
- *Compatibility: if $\varphi((o,q,a),m_1,m'_1)$ then $Pr[\hat{O}_o(q,m_1) \in (a,C)] = Pr[\hat{O}_o(q,m_2) \in (a,C)]$ where C is the equivalence class of m'_1 under $\mathcal{R}$.*

B.4 Determinization

Bisimulation is stronger than language equivalence, and can not always be used to hope from one game to another. In particular, bisimulation can not be used for eager/lazy sampling, or for extending the internal state of the oracle system. The goal of this section is to introduce a general construction, inspired from the subset construction for determinizing automata, to justify such transitions. We consider two oracles systems O and O' and assume that states $m' \in M_{o'}$ can be seen as pairs $(m, m") \in M_o \times M_{o"}$. There are two ways to compute the probability to end up $(m, m")$ for a fixed $m"$ knowing that the step starts with a state of first component m. The first is to perform the exchange in O and then draw $m"$ according to a distribution γ. The second is to look at all possible $m"$ which γ map to m and then to perform the exchange in O'. Imposing the equality between these two ways of computing probabilities is going to compel the same equality to hold for steps, which in turn propagates to traces.

Definition 11. *Let O and O' be compatible oracle systems. O determinizes O' by $\gamma : M_o \to D(M_o)$, written $O \leq_{det,\gamma} O'$, iff $M_o \times M_o" = M'_o$ and there exists $\bar{m}_o"$ such that $(\bar{m}_o, \bar{m}_o") = \bar{m}'_o$, and $\gamma(\bar{m}_o) = \delta_{\bar{m}_o"}$, and $Pr[\gamma(m_2 = m_2")]p_1 = \sum_{m_1" \in M_o"} Pr[\gamma(m_1 = m_1")]p_2(m_1")$ for all $m_1, m_2 \in M_o$, $m_1", m_2" \in M_o"$, where $p_1 = Pr[O(o_c, q, m_1) = (a, m_2)]$ and $p_2(m_1") = Pr[O'(o_c, q, (m_1, m_1")) = (a, (m_2, m_2"))]$.*

We define a projection function π from O'-traces to O-traces by extending the projection from $M_o \times M_o"$ to M_o.

C Proofs for extended rules

C.1 Proof of the rule Fail2

Lemma 1. *Rule Fail2 defined as follows is sound:* $\dfrac{}{O :_{\varepsilon'} F_\varphi}$ *Fail2* *where* $\varepsilon' = \varepsilon \times \dfrac{(\sum_{o \in I} k_o)^2}{2}$ *and*

- *k_o is the maximal number of queries of the oracle o and n is the cardinal of the set N_o*
- *I is the family of oracles that can ensure that the step-predicate φ can be satisfied: o can be an oracle in $N_o \setminus I$ such that $\varepsilon_o(k_{o_1}, \cdots, k_{o_n}) = 0$ or an oracle in I such that $\exists \varepsilon$, $\varepsilon_o(k_{o_1}, \cdots, k_{o_n}) = \varepsilon \times \sum_{o' \in I} k_{o'}$*

Proof Let A be a (k, t)-adversary for oracle system O. Let φ be a step-predicate in $Xch \times \hat{M} \times \hat{M}$. We denote by T the set of traces satisfying F_φ. We recall that the event "eventually φ", written F_φ, means φ being satisfied at *one* step of a trace. Let I be the family of oracles o that can ensure that the step-predicate φ can be satisfied, $I \subseteq N_o$. We define n as the cardinal of the set N_o and for one oracle $o \in N_o$, k_o is the maximal number of its queries.

Let the trace τ in T be the sequence of the form $m_0 \xrightarrow{x_1} m_1 \xrightarrow{x_2} \cdots \xrightarrow{x_l} m_l$ where $m_0, \cdots, m_l \in M_o$ and $x_1, \cdots, x_l \in Xch$ such that $Pr[O_{o_i}(q_i, m_{i-1}) = (a_i, m_i)] > 0$ for $i = 1, \cdots, l$ and $x_i = (o_i, q_i, a_i)$. Therefore, there exists one m_{i_0} such that φ becomes satisfied, where $i_0 \in [1, \cdots, l]$.

We write two hypothesis:

- let o be an oracle in $N_o \setminus I$ such that $\varepsilon_o(k_{o_1}, \cdots, k_{o_n}) = 0$
- let o be an oracle in I such that $\exists \varepsilon$, $\varepsilon_o(k_{o_1}, \cdots, k_{o_n}) = \max_{\{\tau \in T | k_o \text{ queries}\}} Pr[O_o(q, m_{l-1}) = (a, m_l)] = \varepsilon \times \sum_{o' \in I} k_{o'}$ s.t. we denote ε as the maximal number common to all oracles in I

First, we divide traces of set T in subgroups using equivalence relation. Two traces are related iff φ is true *for the first time* at step i for a query to oracle o. Classes are denoted $C(i, o, j)$, where $j = \sum_{o' \in I} k_{o'}$ is the number of good queries (i.e. the queries to oracles in I), and realize a partition of T.

Second, we let $\mathcal{T}$ be the projection mapping sequences of steps to partial traces (see for more details Section 2.4). Then, by definition, the probability that a system yields a trace τ is the sum of the probabilities that the system yields execution η projecting to τ, which we write $Pr[A \mid O : \tau] = \sum_{\{\eta \in Exec(A|O) | \mathcal{T}(\eta) = \tau\}} Pr[A \mid O : \eta]$.

Let $\tau \in C(i, o, j)$. We define Pref(η, i) as the prefix of length i of partial execution η, and $\eta[i]$ its i-th step. Then, we have:

$$\sum_{\tau \in C(i,o,j)} Pr[A \mid O : \tau] = \sum_{\{\tau \in C(i,o,j) \mid \mathcal{T}(\eta)=\tau\}} Pr[A \mid O : \eta] \leq \sum_{\{\tau \in C(i,o,j) \mid \mathcal{T}(\eta)=\tau\}} Pr[A \mid O : \mathrm{Pref}(\eta, i)]$$
$$= \sum_{\{\tau \in C(i,o,j) \mid \mathcal{T}(\eta)=\tau\}} Pr[A \mid O : \mathrm{Pref}(\eta, i-1)].Pr[A \mid O : \eta[i]]$$
$$= \sum_{\{\tau \in C(i,o,j) \mid \mathcal{T}(\eta)=\tau \mid \mathcal{T}(\eta[i])=((o,q,a),m,m')\}} Pr[A \mid O : \mathrm{Pref}(\eta, i-1)].Pr[O_o(q,m) = (a,m')]$$
either
$$\leq \sum_{\{\tau \in C(i,o,j) \mid \mathcal{T}(\eta)=\tau \mid \mathcal{T}(\eta[i])=((o,q,a),m,m')\}} Pr[A \mid O : \mathrm{Pref}(\eta, i-1)] \times j.\varepsilon \leq j.\varepsilon \text{ if } o \in I$$
or
$$\leq \sum_{\{\tau \in C(i,o,j) \mid \mathcal{T}(\eta)=\tau \mid \mathcal{T}(\eta[i])=((o,q,a),m,m')\}} Pr[A \mid O : \mathrm{Pref}(\eta, i-1)] \times 0 = 0 \text{ if } o \notin I$$

Then, we use the fact that equivalence class forms a partition to conclude:

$$Pr[A \mid O : F_\varphi] = \sum_{\tau \in T} Pr[A \mid O : \tau] \sum_{i,o,j} \sum_{\tau \in C(i,o,j)} Pr[A \mid O : \tau] \leq \sum_{o \in I, j} j.\varepsilon = \sum_{o \in I} \left(\sum_{o' \in I} k_{o'} \right) .\varepsilon \leq \varepsilon \times \frac{(\sum_{o \in I} k_o)^2}{2}$$

C.2 Proof of the rule I-BisCd

Lemma 2. *We consider two compatible oracle systems O and O'. Let φ_1 and φ_2 be two step-predicates in $\hat{M}$ and $Xch \times \hat{M} \times \hat{M}$ respectively. The following rule is sound:*

$$\frac{O' :_\varepsilon F_{\neg\varphi_2} \wedge G_{\varphi_1} \qquad O \overset{\varphi_1}{\equiv}_{\mathcal{R},\varphi_2} O'}{O \overset{G_{\varphi_1}}{\sim}_\varepsilon O'} \text{ I-BisCd}$$

Proof We introduce the equivalence relation $\mathcal{R}$ such that for two states m and m' in $\hat{M}_{\varphi_1}$, we have $m\mathcal{R}m'$ and $\varphi_1(m) \wedge \varphi_1(m')$, where the step-predicate φ_1 is in $\hat{M}$ (i.e. φ_1 steps in over the memories but not over the actions in Xch). We recall that $R = \mathbf{True} \wedge G_{\varphi_1} \wedge G_{\varphi_2}$ is a compatible event. We decompose the set of traces created by $A \mid O$ and $A \mid O'$ and verifying $G_{\varphi_1} \wedge G_{\varphi_2}$ into distinct classes of equivalence of a finite number of executions $\sigma_1, \cdots, \sigma_m$, resulting in $Pr[A \mid O : R = \mathbf{True} \wedge G_{\varphi_1} \wedge G_{\varphi_2}] = \sum_{i=1}^m Pr[A \mid O : C_O(\sigma_i)] = \sum_{i=1}^m Pr[A \mid O' : C_{O'}(\sigma_i)] = Pr[A \mid O' : R = \mathbf{True} \wedge G_{\varphi_1} \wedge G_{\varphi_2}]$. Then, we conclude the rule I-BisCd since:

$$Pr[A \mid O : R = \mathbf{True} \wedge G_{\varphi_1}] - Pr[A \mid O' : R = \mathbf{True} \wedge G_{\varphi_1}]$$
$$= Pr[A \mid O : R = \mathbf{True} \wedge G_{\varphi_1} \wedge F_{\neg\varphi_2}] - Pr[A \mid O' : R = \mathbf{True} \wedge G_{\varphi_1} \wedge F_{\neg\varphi_2}]$$
$$\leq max(Pr[A \mid O : R = \mathbf{True} \wedge G_{\varphi_1} \wedge F_{\neg\varphi_2}], Pr[A \mid O' : R = \mathbf{True} \wedge G_{\varphi_1} \wedge F_{\neg\varphi_2}])$$

C.3 Proof of the rule B-BisG2

Lemma 3. *We consider two compatible oracle systems O and O'. Let φ_1 and φ_2 be two step-predicates in $Xch \times \hat{M} \times \hat{M}$. The following rule is sound:*

$$\frac{O :_{\varepsilon_1} F_{\varphi_1} \wedge G_{\varphi_2} \quad O :_{\varepsilon_2} F_{\neg\varphi_2} \quad O \equiv_{\mathcal{R},\varphi_2} O'}{O' :_{\varepsilon_1+\varepsilon_2} F_{\varphi_1}} \text{ B-BisG2}$$

Proof Let φ_1 and φ_2 be step-predicates in $Xch \times \hat{M} \times \hat{M}$. The rule B-BisG2 is obtained from the combination of the rule B-BisG and a variation of this latter rule:

$$\frac{O :_{\varepsilon_1} F_{\varphi_1} \wedge G_{\varphi_2} \quad O \equiv_{\mathcal{R},\varphi_2} O'}{O' :_{\varepsilon_1} F_{\varphi_1} \wedge G_{\varphi_2}} \text{ B-BisG} \quad \frac{O :_{\varepsilon_2} \mathbf{True} \wedge F_{\neg\varphi_2} \quad O \equiv_{\mathcal{R},\varphi_2} O'}{O' :_{\varepsilon_2} \mathbf{True} \wedge F_{\neg\varphi_2}} \text{ B-BisG-variation}$$

We are allowed to conclude since $O' :_{\varepsilon_1} F_{\varphi_1} \wedge G_{\varphi_2}$ and $O' :_{\varepsilon_2} F_{\neg\varphi_2}$.