# RAPPORT TECHNIQUE EVA

## Outils CPV et CPV2


**Auteur** : Jean Goubault-Larrecq
Laboratoire Spécification et Vérification, CNRS UMR 8643, ENS Cachan,
61, avenue du président-Wilson, F-94235 Cachan Cedex
goubault@lsv.ens-cachan.fr


**Date** : 17 mai 2002

**Rapport EVA numro** : 8

**Version** : 1


**TRUSTED LOGIC S.A.**
5 rue du Bailliage
78000 Versailles, France
www.trusted-logic.fr

**Laboratoire Spécification Vérification**
CNRS UMR 8643, ENS Cachan
61, avenue du président-Wilson
94235 Cachan Cedex, France
www.lsv.ens-cachan.fr

**Laboratoire Verimag**
CNRS UMR 5104,
Univ. Joseph Fourier, INPG
2 av. de Vignate,
38610 Gières, France
www-verimag.imag.fr

**Adresse :** Laboratoire Spécification et Vérification,
CNRS UMR 8643, ENS Cachan
61, Avenue du Président Wilson, 94230 Cachan, France
goubault@lsv.ens-cachan.fr

**Résumé :** Le but de cette note est de présenter rapidement les outils CPV et CPV2 de vérification de protocoles cryptographiques. L'outil CPV est une évolution d'un prototype antérieur à EVA, qui a évolué dans le cadre d'EVA. CPV2 est une refonte totale de CPV, en cours de développement, fondée sur des techniques nouvelles et couvrant la quasi-totalité du langage EVA.

# Outils CPV et CPV2

Jean Goubault-Larrecq
Laboratoire Spécification et Vérification, CNRS UMR 8643, ENS Cachan,
61, avenue du président-Wilson, F-94235 Cachan Cedex
goubault@lsv.ens-cachan.fr

17 mai 2002

Le but de cette note est de présenter rapidement les outils CPV et CPV2 de vérification de protocoles cryptographiques. L'outil CPV est une évolution d'un prototype antérieur à EVA, qui a évolué dans le cadre d'EVA. CPV2 est une refonte totale de CPV, en cours de développement, fondée sur des techniques nouvelles et couvrant la quasi-totalité du langage EVA.

# 1 L'outil CPV

## 1.1 Principes de base

L'outil CPV est basé sur des idées présentées dans [6]. Il s'agit d'un outil qui ne vérifie que des propriétés de secret, c'est-à-dire du type `*A *G secret (M@s.A)` dans la syntaxe EVA. D'autre part, l'analyse est approchée: soit l'analyseur retourne que les propriétés de secret sont vérifiées de façon certaine, soit l'analyseur retourne un signal avertissant de la possibilité d'une attaque, c'est-à-dire de la divulgation d'un secret. Il est relativement rare que l'analyseur annonce une attaque alors qu'il n'en existait aucune.

L'outil CPV est fondé sur une approximation supérieure des ensembles de messages connus de l'intrus par automates d'arbres. Il gère des hypothèses de sécurité relativement particulières mais courantes en pratique: typiquement, on se place dans un état initial maximal où l'intrus est incapable de déduire les clés privées (symétriques ou asymétriques) à long terme, ainsi que les nonces non encore créés au démarrage du protocole. Cet état initial maximal est infini mais représentable par automate d'arbres.

CPV gère d'autre part naturellement les principaux en multi-session parallèle, qu'il approxime comme des règles de déduction supplémentaires pour l'intrus, et non comme des processus. La technique, évoquée rapidement dans [6], est voisine de celle de [4], et ressemble à l'approche des strand spaces [12].

## 1.2 Sortie graphique

Voici quelques exemples d'analyses de protocoles. En figure 1, on a affiché les résultats affichés par CPV sur le protocole d'Otway-Rees et celui de Yahalom, tels que décrits dans [3].

On remarquera que chaque branche d'exécution d'Otway-Rees mène à une attaque potentielle: que la clé `Kab` soit potentiellement divulguée correspond à une attaque réelle, traditionnellement réalisée en faisant tourner deux sessions en parallèle, dans laquelle un des participants honnêtes joue les deux rôles du protocole; ici l'attaque est trouvée grâce au fait que les identités de `A` et de `B` ne sont pas supposées distinctes, ce qui permet de simuler un participant jouant les deux rôles en même temps. D'un autre côté, la divulgation potentielle de la clé `sk me/S` (la clé privée partagée entre `A` et `S`, ou entre `B` et `S`, selon la valeur de l'identité `me`) correspond plus probablement à un excès de prudence du vérificateur; l'approximation, relativement brutale, du serveur de clés `S` opérant en multi-session parallèle semble en être responsable.

L'exécution symbolique du protocole de Yahalom est plus intéressante (à droite). Il y a essentiellement deux branches possibles. Sur celle de gauche, on remarque que, d'une part, le secret des clés à long terme `sk me/S` est préservé — mais ici le serveur `S` n'est pas comme ci-dessus en multi-session parallèle. D'autre part, il y a une attaque contre `B` (voir le rectangle du bas de la branche gauche), qui est réelle, et qui exploite une faiblesse dans les garanties

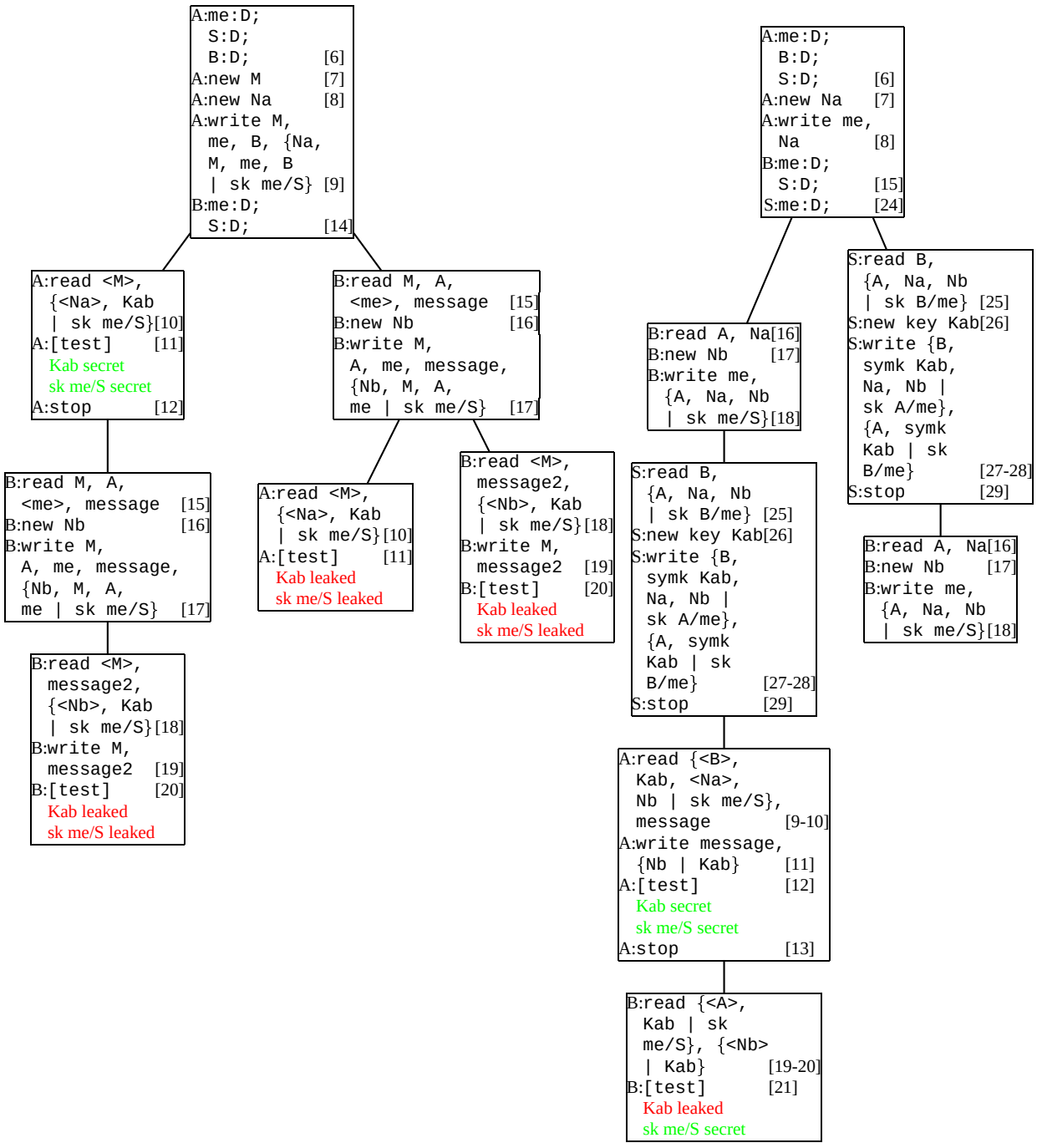


Figure 1: Les protocoles d'Otway-Rees (gauche) et de Yahalom (droite)

de fraîcheur du protocole. Finalement, la clé Kab que A récupère dans le protocole est, elle, secrète: le protocole est donc *sûr* pour ce qui est du rôle A. Il est à noter que la branche droite s'interrompt brutalement sans que le protocole termine: il s'agit d'une collection d'exécutions où l'intrus a cherché à tromper les participants, mais ne réussit pas à produire des messages de la forme attendue par les participants A ou B; en somme, les intrusions le long de cette branche sont détectées.

L'analyse termine en 4,78 s. pour le protocole d'Otway-Rees, 1,31 s. pour celui de Yahalom. Le protocole de Gong est analysé en figure 2, en 1,8 s. Il est à noter que le serveur y tourne en multi-session parallèle, montrant que l'approximation brutale des principaux en multi-session parallèle est suffisante ici. Ce protocole est d'autre part remarquable par le fait que quoi que l'intrus fasse, A et B dérouleront le protocole dans l'ordre spécifié, contrairement aux protocoles ci-dessus, qui exhibent une structure branchante d'exécutions.

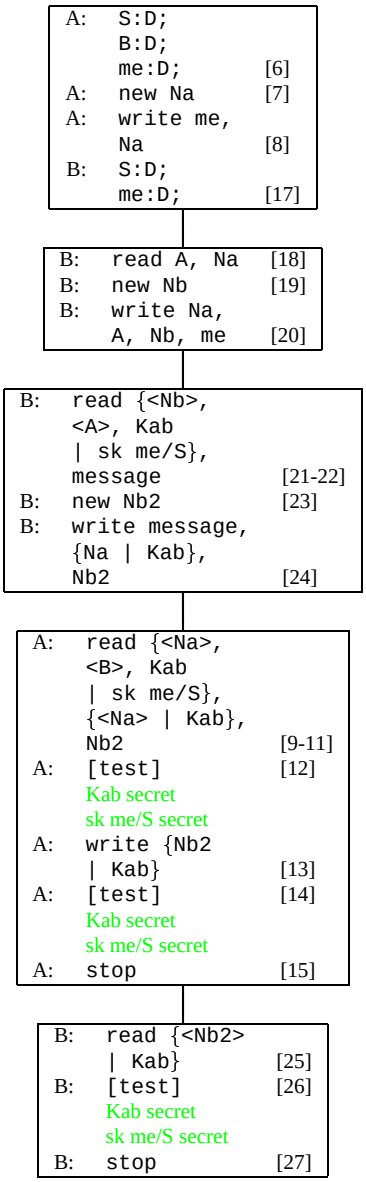


Figure 2: Le protocole de Gong

### 1.3 État d'avancement et connexion avec EVA

L'outil CPV est terminé. Il n'est cependant pas du tout connecté à EVA. La sémantique et le modèle de sécurité sous-jacents à CPV sont cependant proches de ceux d'EVA. Il a été décidé de ne pas poursuivre le développement et l'interfaçage de CPV avec EVA, l'outil CPV2 devant être fondé sur de nouvelles idées.

## 2 L'outil CPV2

### 2.1 Principes de base

Si CPV est fondé sur les automates d'arbres, CPV2 est fondé sur une retranscription des protocoles en logique du premier ordre. Bien que ceci ait l'air très différent, il se trouve que l'outil scientifique naturel pour coder les protocoles cryptographiques dans le style de CPV est celui des automates d'arbres *bidirectionnels alternants*, ou bien de façon équivalente une certaine forme de contraintes ensemblistes définies, ou encore de façon équivalente des formules de la *classe monadique*. Cette connexion entre logique du premier ordre, et plus spécifiquement la classe monadique, et les contraintes ensemblistes est exposée dans [2].

Cette connexion est étendue au cas des automates modulo une théorie équationnelle, ce qui est l'une des nécessités posée par la clause `axiom` d'EVA, dans [10] pour le cas de plusieurs symboles associatifs et commutatifs, et dans [7, 8] pour le cas de l'ordre supérieur, qui devrait permettre de modéliser certaines constructions impliquant des lieurs (on pense au $\nu$ du spi-calcul [1]).

La vision des protocoles dans CPV2 est présentée en figure 3.

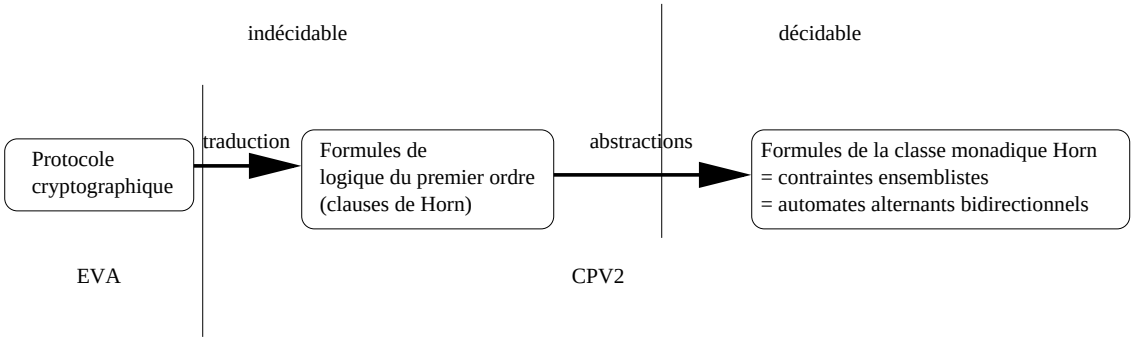


Figure 3: L'architecture de CPV2

Les approches par traduction en logique du premier ordre est originellement due à C. Meadows, et plus tard à C. Weidenbach ou D. Bolignano; on se reportera à [9] pour un panorama.

L'intérêt d'une telle architecture est double: d'une part la traduction en logique du premier ordre permet une meilleure documentation du processus de traduction — ceci est dans l'optique de pouvoir *expliquer* la vérification d'un protocole (le E de "EVA"). D'autre part elle permet de modulariser les abstractions (la flèche de droite de la figure 3) en composant des abstractions génériques pour les programmes logiques, à commencer par des approximations cartésiennes [5], mais comprenant aussi des approches par partitionnement de prédicats dans les programmes logiques (une approche similaire à celle du partitionnement dynamique en interprétation abstraite), ainsi que de transformations par *magic sets* [11]. L'utilité de ces dernières reste cependant à évaluer dans le contexte de la vérification des protocoles cryptographiques.

Dans tous les cas, l'abstraction est telle que toute réponse positive de l'outil sur l'abstraction (à droite de la figure 3) entraîne la validité des propriétés de sécurités souhaitées sur le modèle au premier ordre complet et donc sur le protocole cryptographique EVA (à gauche de la figure 3).

## 2.2 État d'avancement et connexion avec EVA

Le code du moteur de CPV2 est terminé à 70%. Étant une seconde mouture, il travaille directement sur la syntaxe abstraite PEVA d'EVA. Les sorties ne sont pas encore codées.

Il est prévu que CPV2 sache vérifier un fragment positif de la logique d'EVA, suffisant pour exprimer les propriétés de sécurité (`claim` en LAEVA) les plus courantes, dont le secret et deux versions d'authentification. Les hypothèses de sécurité (`assume` en LAEVA) ne seront pas toutes traitées. Dans un premier temps, les hypothèses standard de CPV seront faites. Il est ensuite prévu d'intégrer les hypothèses de la forme "initialement, ... est secret" (`assume secret (X@s.A)`), ce qui devrait être suffisant pour la plupart des utilisations pratiques.

CPV2 sait en revanche traiter tous les aspects liés à l'exécution de protocoles dans le modèle EVA: nombre arbitraire de principaux, en mono ou multi-session, gestion des algorithmes de chiffrement et des inverses de clés correspondants à chaque algorithme, analyse de cas et branchements conditionnels (`switch`) notamment. Il sait de plus traiter certaines constructions admises dans la syntaxe abstraite PEVA d'EVA et qui ne sont pas encore descriptibles dans la syntaxe concrète LAEVA, notamment la présence de boucles.

# References


[1] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*. ACM Press, 1997.

[2] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Set constraints are the monadic class. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 75–83. IEEE Computer Society Press, 1993.

[3] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society*, 426(1871):233–271, 1989.

[4] Mourad Debbabi, M. Mejri, Nadia Tawbi, and I. Yahmadi. Formal automatic verification of authentication cryptographic protocols. In *1st IEEE International Conference on Formal Engineering Methods (ICFEM'97)*. IEEE, 1997.

[5] Thom Frühwirth, Ehud Shapiro, Moshe Y. Vardi, and Eyal Yardeni. Logic programs as types for logic programs. In *LICS'91*, 1991.

[6] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification (extended abstract). In *Proceedings of the Workshop on Formal Methods in Parallel Programming, Theory and Applications, 15th IPDPS*, pages 977–984. Springer-Verlag LNCS 1800, 2000.

[7] Jean Goubault-Larrecq. Higher-order automata, pushdown systems, and set constraints. Research Report LSV-01-9, LSV, 2001. Submitted.

[8] Jean Goubault-Larrecq. Higher-order positive set constraints. In *CSL'02*. Springer-Verlag, 2002. To appear.

[9] Jean Goubault-Larrecq. Sécurité, modélisation et analyse de protocoles cryptographiques. *Phœbus, la revue de la sûreté de fonctionnement*, 20, 2002. Numéro spécial sur la sécurité des systèmes d'information.

[10] Jean Goubault-Larrecq and Kumar Neeraj Verma. Alternating two-way AC-tree automata. In *CSL'02*. Springer-Verlag, 2002. To appear.

[11] Ulf Nilsson. Abstract interpretation: A kind of magic. *TCS*, 142(1):125–139, 1995.

[12] F. Javier Thayer Fábrega and Jonathan C. Herzog. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.