UNIVERSITÀ DEGLI STUDI DI BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI

Corso di Laurea in Scienze dell'Informazione

UN APPROCCIO ALGEBRICO PROBABILISTICO ALL'ANALISI DI PROPRIETÀ DI SICUREZZA DI SISTEMI CRITTOGRAFICI

Tesi di Laurea di:

Angelo Troina

Relatore:

Prof. Roberto Gorrieri

Correlatore:

Dott. Alessandro Aldini

Insegnamento: *Metodi per il Trattamento dell'Informazione*

Sessione II

Anno Accademico 2001/2002

Indice

1 INTRODUZIONE 4

2 INFORMATION FLOW 12

2.1 UN LINGUAGGIO NON-DETERMINISTICO 15

2.1.1 Non-deterministic Process Algebra 16

2.1.2 La semantica formale 18

2.2 NON-DEDUCIBILITY ON COMPOSITION 21

3 IL MODELLO PROBABILISTICO 26

3.1 UN LINGUAGGIO PROBABILISTICO 27

3.1.1 *GRTS* 28

3.1.2 Probabilistic Process Algebra 30

3.1.3 La semantica formale 38

3.2 BISIMULAZIONE DEBOLE PROBABILISTICA 42

3.2.1 Un esempio 46

3.3 PROPRIETÀ DI SISTEMI PROBABILISTICI 47

3.3.1 Probabilistic Non-deducibility on Composition 49

3.3.2 Rilevazione di flussi di informazione probabilistici 52

3.3.3 Estensione conservativa 57

3.3.4 Misura del flusso di informazione 59

4 MODELLO FORMALE PER LA CRITTOGRAFIA.....64

4.1 ESPRESSIONI 66

4.2 EQUIVALENZA 67

4.2.1 Alcune considerazioni.....70

4.3 EQUIVALENZA PROBABILISTICA DI ESPRESSIONI 71

4.3.1 Pattern Probabilistico 79

5 IL MODELLO CRITTOGRAFICO-PROBABILISTICO 88

5.1 UN LINGUAGGIO CRITTOGRAFICO-PROBABILISTICO.....89

5.1.1 Crypto Probabilistic Process Algebra.....90

5.1.2 La semantica formale.....95

5.1.3 Alcuni esempi.....98

5.2 PROPRIETÀ DI SICUREZZA DI SISTEMI CRITTOGRAFICI 102

5.2.1 Crypto Probabilistic Non-deducibility on Composition.....106

5.2.2 Proprietà di Autenticazione 112

6 UN CASO DI STUDIO 116

6.1 INTRODUZIONE 116

6.2 UN PROTOCOLLO PROBABILISTICO DI NON-REPUDIATION 117

6.3 ANALISI DELLA SICUREZZA DEL PROTOCOLLO.....121

6.3.1 Proprietà di non-repudiation.....124

7 CONCLUSIONI.....130

RIFERIMENTI BIBLIOGRAFICI 132

1 Introduzione

I sistemi computerizzati sono ormai divenuti indispensabili per molteplici tipi di attività in ambito accademico, commerciale, industriale o governativo. Permettono di gestire informazioni in maniera estremamente rapida, migliorano le comunicazioni e ne riducono i costi, permettono di offrire servizi sempre più efficienti e veloci.

Questi cambiamenti hanno rivoluzionato, negli ultimi vent'anni, il modo di condurre gli affari e di svolgere molteplici altre attività. Tuttavia hanno anche introdotto il rischio di attacchi di tipo informatico che possono rivelarsi fatali per le organizzazioni, conducendo a perdite di tempo, denaro, reputazione e informazioni private.

La maggior parte del software, presente oggi sul mercato, ha raggiunto un livello di tale semplicità d'uso che l'installazione e l'utilizzo è consentito anche a persone con scarse conoscenze tecniche, mentre risulta difficile configurare e operare in modo sicuro su molti di questi prodotti. Una causa della presenza, sui sistemi computerizzati, di un gran numero di vulnerabilità è proprio il gap esistente tra le conoscenze necessarie per operare su un sistema e quelle necessarie per renderlo sicuro. Tali vulnerabilità sono gli elementi utilizzati da coloro che intendono portare attacchi a sistemi computerizzati.

Un altro aspetto importante è la variazione del profilo di tali persone. Difatti mentre negli anni '80, appartenevano a questa categoria solamente pochi esperti in materia, che sfruttavano per le loro azioni strumenti creati da essi stessi, adesso la situazione è notevolmente cambiata. Chiunque può,

mediante l'utilizzo di strumenti che sono facilmente reperibili nella rete mondiale (Internet), effettuare diverse tipologie di attacchi a volte senza neanche conoscerne le conseguenze, ma solo per puro divertimento.

Un contesto di questo genere ha contribuito notevolmente ad aumentare l'interesse per le caratteristiche di sicurezza di sistemi. E' divenuto dunque importante definire con precisione alcune proprietà di sicurezza per ottenere una verifica formale della correttezza di un protocollo o di un sistema.

Nella prima parte di quest'opera analizzeremo, in particolare, le proprietà di sicurezza comunemente classificate come proprietà di *information flow*, il cui obiettivo è quello di controllare il tipo di informazione che può passare legalmente tra differenti entità. Inizialmente questo tipo di proprietà di sicurezza è stato introdotto per assicurare la segretezza delle informazioni, in particolare vengono usate per verificare se le politiche di controllo di accesso ai dati sono sufficienti a garantire la privatezza dell'informazione.

Si consideri, ad esempio la politica di controllo degli accessi *Mandatory Access Control* (MAC in breve), in cui alcune regole sono stabilite dal sistema. Un esempio molto comune di MAC è la *sicurezza multilivello* [35] in cui ogni oggetto (ad esempio dei dati) ed ogni soggetto (ad esempio un utente) è relegato ad un livello di sicurezza; un soggetto può quindi utilizzare l'informazione di un oggetto solamente se il livello di sicurezza in cui si trova è maggiore del livello di sicurezza in cui si trova l'oggetto. Questa politica può essere implementata attraverso due semplici regole di controllo degli accessi: *No Read Up* (un soggetto non può leggere dati da un oggetto che si trovi ad un livello di sicurezza maggiore) e *No Write Down* (un soggetto non può scrivere dati su un oggetto che si trovi ad un livello di sicurezza inferiore). Tuttavia neppure la più rigida politica MAC è in grado di assicurare una completa sicurezza contro potenziali perdite di informazione. Potrebbe infatti essere possibile trasmettere informazioni in maniera *indiretta* attraverso

effetti collaterali del sistema. Se ad esempio una risorsa di memoria finita (come potrebbe essere un hard disk) è condivisa da due livelli, "high" e "low", si potrebbero trasmettere dati dal livello "high" al livello "low" utilizzando il messaggio di errore che si ottiene in caso di "risorsa piena". Se si vuole trasmettere un'informazione binaria dal livello "high" è sufficiente riempire o svuotare la risorsa condivisa per trasmettere il bit "1" o il bit "0". Simultaneamente infatti dal livello "low" un altro soggetto può tentare di scrivere sulla risorsa condivisa decodificando un messaggio di errore con un "1" e una scrittura effettuata con successo con uno "0". Trasmissioni indirette di questo tipo, che sfruttano un canale di comunicazione coperto (*covert channel*), non violano le due regole per il controllo degli accessi della sicurezza multilivello. Diviene spesso necessario, di conseguenza, integrare una politica di controllo degli accessi di tipo MAC con un'analisi per la ricerca di covert channels.

Ponendo alcune regole di information flow diviene quindi possibile controllare l'esistenza di canali di comunicazione coperti che possono dare origine a trasferimenti indiretti di informazione.

L'intuizione comune alla base delle proprietà di information flow è strettamente legata alla classica nozione della *Non-Interference* [37], in cui gli utenti di basso livello non devono essere in grado di dedurre alcuna informazione riguardo l'attività degli utenti di alto livello. In particolare, ogni possibile interazione effettuata dal sistema con l'ambiente esterno (immaginando che tali interazioni vengano modellate attraverso azioni di alto livello), non devono alterare il comportamento osservabile, da un punto di vista di basso livello, del sistema isolato. Per raggiungere tale scopo è necessario definire un equivalenza semantica che permetta di stabilire se il comportamento del sistema che interagisce con l'esterno (potenzialmente insicuro) sia uguale al comportamento del sistema isolato (sicuro poiché non interagisce con alcun utente ostile).

In particolare, presenteremo un algebra di processi non-deterministica che permette di modellare sistemi multilivello, quindi,

utilizzando la nozione di bisimulazione debole [19] come equivalenza semantica per il confronto del comportamento osservabile di sistemi, verificheremo la loro sicurezza attraverso la proprietà di *Non-deducibility on Composition* (*NDC*). Questa proprietà, strettamente legata alla nozione della non-interference, si basa sull'idea che il comportamento osservabile di un sistema isolato non deve essere alterato quando si considerano tutte le possibili interazioni del sistema con agenti di alto livello dell'ambiente esterno.

In molti casi, tuttavia, l'analisi del comportamento di un sistema dovrebbe tenere in considerazione anche aspetti addizionali, come ad esempio la distribuzione di probabilità delle azioni osservabili che un sistema può effettuare, oppure il tempo impiegato per eseguirle. In particolare, analizzare il comportamento probabilistico di un sistema permette di individuare canali di comunicazione coperti probabilistici, che non vengono rilevati considerando solamente il suo comportamento non-deterministico. Tali trasferimenti indiretti di informazione vengono infatti catturati osservando una variazione nelle frequenze delle azioni eseguite dal sistema.

L'approccio probabilistico all'analisi del comportamento di un sistema permette inoltre di superare la prospettiva tradizionale in cui i sistemi vengono classificati come sicuri o come insicuri in maniera binaria. In particolare, assegnando una misura probabilistica ad ogni flusso di informazione che potrebbe rivelarsi, è possibile stabilire il livello di sicurezza di un sistema. In pratica, per considerare il sistema sufficientemente sicuro, si controlla che la probabilità di osservare un flusso di informazione insicuro sia inferiore ad una soglia prestabilita. Inoltre, modellando il comportamento probabilistico di un sistema, è possibile sia analizzarne le proprietà di sicurezza e i flussi di informazione, sia effettuarne una valutazione delle prestazioni.

Nella seconda parte di quest'opera analizzeremo quindi un'algebra di processi [14, 1] che permette di modellare il comportamento di sistemi

probabilistici. Tale formalismo è un'estensione naturale e conservativa del modello non-deterministico. In particolare, se un sistema non dà origine a flussi di informazione nel modello probabilistico, la controparte non-deterministica del sistema è altrettanto sicura.

Il contributo fondamentale apportato dal nostro studio è quindi l'estensione di questo potente modello probabilistico con alcuni operatori che permettano di gestire operazioni crittografiche e di manipolare messaggi. La nostra idea è, infatti, che un approccio classico allo studio della sicurezza, come quello utilizzato per l'analisi dei flussi di informazione in sistemi multilivello [35], possa essere usato proficuamente anche per l'analisi di proprietà di sicurezza di protocolli crittografici.

Definiremo, quindi, un nuovo linguaggio che permetta di modellare il comportamento probabilistico di sistemi e di specificare proprietà di sicurezza di protocolli crittografici. In particolare, attraverso il nostro nuovo formalismo sarà possibile definire e modellare protocolli crittografici che utilizzano algoritmi stocastici e, di conseguenza, si comportano in maniera probabilistica. Algoritmi di questo tipo, il cui uso si sta largamente diffondendo negli ultimi anni, permettono infatti di trasformare problemi con complessità esponenziale in problemi con complessità polinomiale.

Nella nostra ottica, una trattazione astratta delle operazioni di crittografia è più che adeguata per la definizione, l'analisi e l'implementazione di sistemi crittografici. Abbiamo deciso, ad esempio, di ignorare i dettagli delle funzioni di cifratura utilizzate, modellando con una descrizione di più alto livello gli obiettivi che ci siamo prefissati. In letteratura si possono trovare diversi modelli che trattano le operazioni di crittografia in maniera puramente formale.

L'espressione $\{M\}_K$ rappresenta, ad esempio, un messaggio cifrato in cui M è il testo in chiaro e K è la chiave di cifratura utilizzata. Da questo punto di vista $\{M\}_K$, M e K rappresentano espressioni formali più

che stringhe di bit. Varie funzioni possono poi essere applicate a questo tipo di espressioni generandone altre. Una di queste funzioni potrebbe essere quella di decifratura, che, prese in input le espressioni formali $\{M\}_K$ e K, restituisce l'espressione M. In particolare, non dovrebbe esserci alcun modo di ricavare M o K dall'espressione $\{M\}_K$ presa singolarmente.

Quest'ultima considerazione ha senso solamente se si assume crittografia perfetta, ovvero se si considera nulla la probabilità di decrittare un testo cifrato di cui non si conosce la chiave. Nei protocolli reali è tuttavia impossibile utilizzare algoritmi di cifratura perfetti. Un agente disonesto può, infatti, portare attacchi a qualsiasi algoritmo di cifratura reale, e, sebbene con probabilità estremamente basse, potrebbe riuscire a decrittare un messaggio senza conoscere la chiave utilizzata per cifrarlo. In letteratura non esistono approcci formali all'analisi delle proprietà di sicurezza di protocolli crittografici che non assumano crittografia perfetta. Noi abbiamo invece deciso di modellare uno scenario in cui siano contemplati anche possibili attacchi agli algoritmi di cifratura utilizzati. Abbandoneremo dunque l'assunzione di crittografia perfetta e definiremo formalmente alcune operazioni di crittografia che ci permettano di modellare la possibilità di decrittare un messaggio $\{M\}_K$ senza conoscere la chiave K.

Per rendere la trattazione più semplice e chiara abbiamo deciso di limitarci alla definizione di uno strumento formale che permette di modellare algoritmi di cifratura simmetrici.

Unità fondamentale del modello per la gestione delle operazioni di crittografia sono le *espressioni*, queste rappresentano i messaggi trasmessi nei protocolli di sicurezza. Le espressioni sono costruite, partendo dall'insieme dei bit $\{0, 1\}$ e da un insieme di chiavi, attraverso operazioni di accoppiamento e cifratura. Mostreremo quindi una relazione di equivalenza tra espressioni, che permette di stabilire se dei dati possono apparire uguali ad un nemico che non ha conoscenza delle chiavi di cifratura utilizzate. Infine, assumendo crittografia imperfetta e

modellando possibili attacchi all'algoritmo di cifratura, definiremo un'equivalenza probabilistica che ci permetta di confrontare le espressioni analizzando le informazioni contenute in esse e la probabilità con cui un nemico può ricavarle. In particolare, stimeremo la probabilità massima con cui un nemico può rompere un blocco crittato, attraverso attacchi all'algoritmo di cifratura, analizzando nella maniera ottima la conoscenza di cui dispone.

Definiremo quindi nel contesto della nuova algebra di processi la proprietà della *Non-deducibility on Composition* per l'analisi probabilistica della sicurezza di sistemi crittografici. In particolare, poiché non si assume crittografia perfetta, fissata una soglia ε, diremo che un sistema è sufficientemente sicuro se la probabilità di rilevare un comportamento scorretto è minore o uguale ad ε. Estenderemo poi la proprietà *NDC* per sistemi crittografici definendo la proprietà di autenticazione dei messaggi, una delle principali proprietà di sicurezza di protocolli.

Attraverso alcuni esempi mostreremo quindi come utilizzare in pratica tali proprietà e soprattutto mostreremo la naturalezza con cui gli operatori crittografici si adattano al modello probabilistico (soprattutto in assenza di crittografia perfetta).

Infine mostreremo come caso di studio un protocollo probabilistico di non-repudiation e definiremo una versione riadattata della proprietà di non-repudiation.

2 Information Flow

Una sempre più ampia diffusione di sistemi distribuiti, in cui risorse e dati sono condivisi tra utenti localizzati in varie parti del mondo, ha contribuito notevolmente ad aumentare l'interesse per le caratteristiche di sicurezza di sistemi. In contesti di questo genere è possibile che, all'interno di una rete, un utente ottenga da una fonte non affidabile alcuni programmi "*maliziosi*" e li esegua nel proprio sistema producendo risultati imprevedibili. Inoltre in alcuni casi è possibile che sistemi totalmente sicuri al loro interno risultino invece insicuri durante attività critiche come l'e-commerce o l'home banking a causa di alcune debolezze nel meccanismo di gestione delle connessioni remote o delle transazioni. E' divenuto dunque importante definire con precisione alcune proprietà di sicurezza per ottenere una verifica formale della correttezza di un protocollo o di un sistema. Negli ultimi anni sono quindi state proposte svariate definizioni formali per la descrizione di proprietà di sicurezza (si vedano ad esempio [18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34]).

Le proprietà di sicurezza che andremo ad analizzare in questo capitolo sono quelle comunemente classificate come proprietà di *information flow*, il cui obiettivo è quello di controllare il tipo di informazione che può passare legalmente tra differenti entità. Inizialmente questo tipo di proprietà di sicurezza è stato introdotto per assicurare la segretezza delle informazioni, in particolare vengono usate

per verificare se le politiche di controllo di accesso ai dati sono sufficienti a garantire la privatezza dell'informazione. Infatti, sebbene le politiche di controllo degli accessi siano tecniche per la sicurezza di sistemi altamente studiate, non è un problema di facile risoluzione trovare una politica di controllo degli accessi che garantisca che sia impossibile qualsiasi perdita di informazione. Uno dei problemi principali è quindi quello di limitare e possibilmente evitare qualsiasi danno prodotto da applicazioni *maliziose*, quali ad esempio i *Trojan Horses*, che cercano di causare la perdita di informazioni segrete.

Si consideri, ad esempio, un classico *Discretionary Access Control* (DAC in breve), in cui ogni *soggetto* (ad esempio un agente attivo che potrebbe essere un utente) decide le proprietà di accesso ai suoi *oggetti* (come ad esempio agenti passivi che potrebbero essere files). La gestione dei files in UNIX è un esempio classico di DAC. Questo tipo di politica permette quindi all'utente di controllare i diritti di accesso ai suoi files e proprio questa flessibilità causa una perdita di sicurezza. Se ad esempio un utente esegue un Trojan Horse possono venire modificate (dallo stesso utente e inconsapevolmente) le proprietà di accesso ai suoi dati, rendendo ad esempio informazioni private visibili ad ogni altro utente. Da questo punto di vista possiamo affermare che la politica DAC non offre alcuna garanzia contro attacchi "interni".

Un approccio diverso al problema è dato dalla politica *Mandatory Access Control* (MAC in breve), in cui alcune regole di accesso sono stabilite dal sistema. Con una politica di questo tipo sebbene un utente si trovi ad eseguire un Trojan Horse le sue azioni sono comunque limitate dal momento che non è in grado di modificare le regole MAC. Un esempio molto comune di MAC è la *sicurezza multilivello* [35] in cui ogni oggetto ed ogni soggetto è relegato ad un livello di sicurezza; l'informazione può passare da un particolare oggetto ad un particolare soggetto solamente se il livello di sicurezza in cui si trova il soggetto è maggiore del livello di sicurezza in cui si trova l'oggetto. In questo modo un Trojan Horse che si trovi ad operare ad un certo livello non è in grado

di rendere l'informazione disponibile ai livelli più bassi è la sua azione è limitata a quel particolare livello di sicurezza. Questa politica può essere implementata attraverso due semplici regole di controllo degli accessi: *No Read Up* (un soggetto non può leggere dati da un oggetto che si trovi ad un livello di sicurezza maggiore) e *No Write Down* (un soggetto non può scrivere dati su un oggetto che si trovi ad un livello di sicurezza inferiore).

Tuttavia neppure adoperare la più rigida politica MAC è in grado di garantire una completa sicurezza contro l'azione di un Trojan Horse. Potrebbe infatti essere ancora possibile trasmettere informazioni in maniera indiretta attraverso effetti collaterali del sistema. Se ad esempio una risorsa di memoria finita (come potrebbe essere un hard disk) è condivisa da due livelli, "high" e "low", si potrebbero trasmettere dati dal livello "high" al livello "low" utilizzando il messaggio di errore che si ottiene in caso di "risorsa piena". Se si vuole trasmettere un'informazione binaria dal livello "high" è sufficiente riempire o svuotare la risorsa condivisa per trasmettere il bit "1" o il bit "0". Simultaneamente infatti dal livello "low" un altro soggetto può tentare di scrivere sulla risorsa condivisa decodificando un messaggio di errore con un "1" e una scrittura effettuata con successo con uno "0". E' chiaro che trasmissioni indirette di questo tipo, sfruttando un canale di comunicazione coperto (*covert channel*), non violano le due regole per il controllo degli accessi della sicurezza multilivello (si veda la fig. 2.1). Diviene spesso necessario, di conseguenza, integrare una politica di controllo degli accessi di tipo MAC con un'analisi per la ricerca di covert channels (si veda ad esempio [36]).

L'esistenza dei covert channels ha condotto ad uno studio di carattere generale sull' information flow. L'idea di base è quella di riuscire a controllare direttamente l'intero flusso dell'informazione in un sistema piuttosto che le politiche di accesso dei soggetti agli oggetti [37]. Ponendo alcune regole di information flow diviene quindi possibile controllare sia trasferimenti di informazione diretti sia trasferimenti di

informazione indiretti, poiché, da questo punto di vista, entrambi rappresentano "flussi di informazione indesiderati".

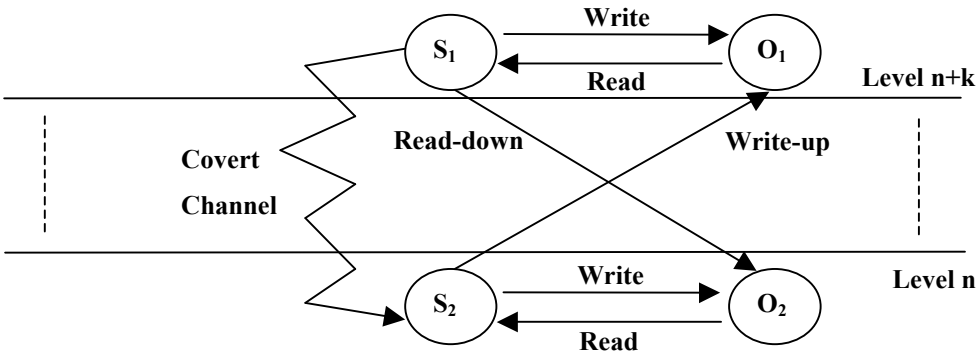


Fig. 2.1. Flussi di informazione nella sicurezza multilivello.

In letteratura si possono trovare svariate definizioni di sicurezza che si rifanno all'idea dell'information flow, ognuna delle quali sviluppata su un particolare modello (si vedano ad esempio [37, 38, 39, 40, 41, 42, 43]). Nel prossimo paragrafo presentiamo il linguaggio formale che sarà usato per specificare ed analizzare proprietà di sicurezza di sistemi concorrenti non-deterministici. Nel secondo paragrafo mostreremo nello specifico la proprietà di sicurezza *BNDC* in grado di rilevare l'esistenza di flussi di informazione tra gruppi di utenti. In particolare tale proprietà controlla che il comportamento del sistema composto con tutti i possibili agenti di alto livello sia equivalente, dal punto di vista degli utenti di basso livello, al comportamento del sistema isolato.

2.1 Un linguaggio non-deterministico

In questa sezione presentiamo il linguaggio formale che useremo per verificare le proprietà di sicurezza di sistemi non-deterministici. Nel primo paragrafo specifichiamo la sintassi del linguaggio, nel secondo invece presentiamo la sua semantica operazionale e definiamo una relazione di equivalenza. Il linguaggio, chiamato *Non-deterministic Process Algebra* (*NDPA* in breve), è sostanzialmente una variante dei

linguaggi *CSP* (Hoare, [12]) e *CCS* (Milner, [19]) in cui l'insieme delle azioni osservabili è partizionato in due livelli. Come avviene per la descrizione della semantica operazionale del *CCS* il modello utilizzato per specificare la semantica operazionale di *NDPA* è il *labelled transition system* (*LTS*) [44], in cui gli stati sono rappresentati dai termini dell'algebra. Per astrarre dai comportamenti che sono indistinguibili per un osservatore esterno, utilizzeremo un equivalenza semantica tra i termini, in modo che due sistemi siano indistinguibili se e solo se sono equivalenti. Le proprietà di sicurezza legate all'information flow sono infatti tutte basate sull'analisi del comportamento osservabile di un sistema.

2.1.1 Non-deterministic Process Algebra

Non-deterministic Process Algebra (*NDPA* in breve) è un'estensione dei linguaggi *CCS* [12] e *CSP* [12], che sono i formalismi maggiormente utilizzati per specificare sistemi concorrenti. Le componenti di base per rappresentare un sistema concorrente sono le sue *azioni*, ovvero le attività atomiche che esso svolge. Nel nostro calcolo l'insieme delle azioni è partizionato in azioni di input ed azioni di output, in cui un'azione di output può sincronizzare con una o più azioni di input (come vedremo nel prossimo capitolo questa assunzione renderà più semplice l'estensione del linguaggio non-deterministico al modello probabilistico). Per poter rappresentare sistemi multilivello (e quindi analizzare il flusso di informazione tra soggetti di livelli separati), differentemente dal *CSP* e dal *CCS*, in *NDPA* le azioni appartengono a due diversi livelli di sicurezza e quindi l'insieme di tutte le azioni osservabili è ulteriormente partizionato in azioni di alto livello ed azioni di basso livello[1].

Formalmente indichiamo con **AType** l'insieme dei tipi delle azioni e

[1] Si noti che con questo modello si possono rappresentare solamente sistemi a due livelli. Questo non costituisce però una reale limitazione poiché è sempre possibile trattare sistemi multilivello raggruppando in coppie i vari livelli fino ad ottenerne due soli.

rappresentiamo i suoi elementi con i simboli a, b, … . Nell'insieme **AType** è incluso il tipo di azione τ, che rappresenta un'azione interna non osservabile. Definiamo poi l'insieme delle azioni di input $I = \{a^* \mid a \in \mathbf{AType}\text{-}\{\tau\}\}$ e l'insieme delle azioni di output $O = \{a \mid a \in \mathbf{AType}\text{-}\{\tau\}\}$. Denotiamo quindi con $\mathbf{Act} = I \cup O \cup \{\tau\}$ l'insieme di tutte le azioni e rappresentiamo i suoi elementi con i simboli π, π', … . Per ottenere una partizione in due livelli dell'insieme delle azioni visibili, definiamo due insiemi disgiunti $\mathbf{AType_H}$ e $\mathbf{AType_L}$, di azioni di alto e basso livello rispettivamente, che costituiscono una copertura dell'insieme $\mathbf{AType}\text{-}\{\tau\}$. Le azioni $a^* \in I$ e $a \in O$ sono azioni di alto (basso) livello se $a \in \mathbf{AType_H}$ ($a \in \mathbf{AType_L}$). Nel seguito di quest'opera rappresenteremo con i simboli h, h',… tipi di azioni di alto livello e con i simboli l, l',… tipi di azioni di basso livello.

Sia quindi **Const** un insieme di termini costanti i cui elementi sono rappresentati dai simboli A, B, … . Denotiamo con $\mathcal{L}_{nd}$ l'insieme dei termini del nostro calcolo, che rappresenteremo con i simboli P_{nd}, Q_{nd}, …[2], generati dalla seguente grammatica:

$$P ::= \underline{0} \mid \pi.P \mid P + P \mid P \|_S P \mid P \setminus L \mid P / L \mid A$$

in cui $\pi \in \mathbf{Act}$, $S, L \subseteq \mathbf{AType}\text{-}\{\tau\}$. Inoltre per ogni termine costante A deve esserci una definizione corrispondente: $A \stackrel{def}{=} P$ in cui P è un termine guardato e chiuso [19].

Intuitivamente $\underline{0}$ rappresenta un termine che non può effettuare nessuna azione, potrebbe indicare un processo terminato o che si trovi in uno stato di stallo; $\pi.P$ può eseguire l'azione π e poi si comporta come il processo P; $P + Q$ (come l'analogo operatore del *CCS*) sceglie in maniera non-deterministica di comportarsi come il processo P o come il processo Q; $P \|_S Q$ (come l'analogo operatore del *CSP*) è la composizione

[2] Ometteremo in genere l'estensione $_{nd}$ quando non possono sorgere ambiguità.

parallela dei processi P e Q, in cui i processi P e Q eseguono localmente le azioni di tipo $a \notin S$ e sono forzati a sincronizzare sulle azioni di tipo $a \in S$. In particolare due azioni di tipo a possono sincronizzarsi solamente se sono due azioni di input a^* (in questo caso il risultato della sincronizzazione è un'azione di input a^*), oppure se una è un'azione di output a e l'altra è un'azione di input a^* (producendo come risultato un'azione di output a). In particolare l'operatore di composizione parallela permette che più azioni di input sincronizzino con una sola azione di output. La politica di sincronizzazione utilizzata è quindi *asimmetrica* e permette che un'azione di output sia inviata in broadcast a tutti i processi concorrenti. $P \setminus L$ può eseguire tutte le azioni del processo P eccetto quelle che appartengono all'insieme L, tale operatore potrebbe essere facilmente ottenuto attraverso l'operatore di composizione parallela con il processo $\underline{0}$, infatti $P \setminus L = P \|_L \underline{0}$; P / L trasforma le azioni il cui tipo $a \in L$ in azioni interne τ, infine attraverso il termine costante A è possibile definire sistemi ricorsivi.

Chiamiamo $\mathcal{G}_{nd}$ l'insieme dei termini guardati e chiusi [19] di *NDPA* ($\mathcal{G}_{nd} \subset \mathcal{L}_{nd}$). Sia quindi $sort(P)$ con $P \in \mathcal{G}_{nd}$, l'insieme delle azioni che occorrono sintatticamente come azioni dell'operatore di prefisso nella definizione di P. Rappresentiamo poi l'insieme degli agenti di alto livello con $\mathcal{G}_{Hnd} = \{P \in \mathcal{G}_{nd} \mid sort(P) \subseteq \mathbf{AType_H}\}$. Infine definiamo una funzione t:**Act**$\rightarrow$**AType** che mappa ogni azione nel suo tipo corrispondente:

$$t(\pi) = \begin{cases} \tau & se\ \pi = \tau \\ a & se\ \pi = a\ \ con\ a \in O \\ a & se\ \pi = a^*\ \ con\ a^* \in I \end{cases}$$

2.1.2 La semantica formale

La semantica operazionale di *NDPA* è data dal *LTS* ($\mathcal{G}_{nd}$, **Act**, $\rightarrow$) in cui gli stati sono rappresentati dai termini dell'algebra e la relazione delle transizioni $\rightarrow \subseteq \mathcal{G}_{nd} \times \mathbf{Act} \times \mathcal{G}_{nd}$ è la minima relazione che soddisfa le

regole in tab. 2.1.

Prefisso	$\dfrac{-}{\pi.P \xrightarrow{\pi} P}$
Composizione Alternativa	$\dfrac{P \xrightarrow{\pi} P'}{P+Q \xrightarrow{\pi} P'}$ $\qquad$ $\dfrac{Q \xrightarrow{\pi} Q'}{P+Q \xrightarrow{\pi} Q'}$
Composizione Parallela	$\dfrac{P \xrightarrow{\pi} P'}{P \parallel_S Q \xrightarrow{\pi} P' \parallel_S Q}\, t(\pi) \notin S$ $\dfrac{Q \xrightarrow{\pi} Q'}{P \parallel_S Q \xrightarrow{\pi} P \parallel_S Q'}\, t(\pi) \notin S$ $\dfrac{P \xrightarrow{a} P' \;\; Q \xrightarrow{a^*} Q'}{P \parallel_S Q \xrightarrow{a} P' \parallel_S Q'}\, a \in S$ $\dfrac{P \xrightarrow{a^*} P' \;\; Q \xrightarrow{a} Q'}{P \parallel_S Q \xrightarrow{a} P' \parallel_S Q'}\, a \in S$ $\dfrac{P \xrightarrow{a^*} P' \;\; Q \xrightarrow{a^*} Q'}{P \parallel_S Q \xrightarrow{a^*} P' \parallel_S Q'}\, a \in S$
Restrizione	$\dfrac{P \xrightarrow{\pi} P'}{P \setminus L \xrightarrow{\pi} P'} \quad t(\pi) \notin L$
Mascheramento	$\dfrac{P \xrightarrow{\pi} P'}{P / L \xrightarrow{\pi} P'} \quad t(\pi) \notin L$ $\qquad$ $\dfrac{P \xrightarrow{\pi} P'}{P / L \xrightarrow{\tau} P'} \quad t(\pi) \in L$
Costante	$\dfrac{P \xrightarrow{\pi} P'}{A \xrightarrow{\pi} P'} \quad A \overset{def}{=} P$

Tab. 2.1. Le regole operazionali di *NDPA*.

Introduciamo ora il concetto di comportamento osservabile: due sistemi dovrebbero avere la stessa semantica se e solo se non possono essere distinti da un osservatore esterno. Per raggiungere tale obiettivo si definisce una relazione di equivalenza tra stati/termini di *NDPA* tale che due processi sono equivalenti se sono indistinguibili.

E' possibile definire diverse equivalenze di questo tipo in base alle diverse capacità che si accordano all'osservatore. In particolare noi

tratteremo un'*equivalenza osservazionale* basata sul concetto della *bisimulazione debole* [19]. Questa scelta è dovuta al fatto che, come mostrato in [20], la bisimulazione debole riesce a superare alcune mancanze mostrate da altri tipi di equivalenza. L'*equivalenza per tracce* non è, ad esempio, in grado di rilevare flussi di informazione causati da possibili stati di stallo; l'*equivalenza failure/testing*, pur rilevando possibili stati di stallo pericolosi per la sicurezza del sistema, si rivela inefficace nel trattare sistemi divergenti (che contengono cioè cicli di azioni di alto livello o cicli di azioni interne τ).

Equivalenza osservazionale

La bisimulazione è un concetto basato sull'idea di una mutua simulazione passo-passo. Ad esempio se P esegue una certa azione e passa in P', Q deve essere in grado di simulare questo singolo passo eseguendo la stessa azione e passando nello stato Q' che ancora bisimula P', e viceversa.

Una bisimulazione debole è una bisimulazione che non tiene in considerazione le azioni interne τ, quando P simula un'azione di Q può quindi eseguire un numero arbitrario di azioni τ prima dell'azione.

Per definire un comportamento di questo tipo utilizziamo la seguente notazione:

Se $w = \pi_1 \ldots \pi_n \in \mathbf{Act}^*$ è una sequenza di azioni diremo che $P \overset{w}{\Rightarrow} P'$ se e solo se esistono $P_1, P_2, \ldots, P_{n-1} \in \mathcal{G}_{nd}$ tali che: $P \xrightarrow{\pi_1} P_1 \xrightarrow{\pi_2} \ldots$ $\ldots P_{n-1} \xrightarrow{\pi_n} P'$. Dato l'insieme $A \subseteq \mathbf{Act}^*$ diremo che $P \overset{A}{\Rightarrow} P'$ se esiste una sequenza di azioni $w \in A$ tale che $P \overset{w}{\Rightarrow} P'$. Diciamo che P' è *raggiungibile* da P se esiste $w \in \mathbf{Act}^*$ tale che $P \overset{w}{\Rightarrow} P'$, scriveremo $P \Rightarrow P'$. Chiamiamo $Der(P) = \{P' \mid P \Rightarrow P'\}$ l'insieme dei termini raggiungibili da P. Data un'azione $\pi \in \mathbf{Act}$ usiamo l'abbreviazione $P \overset{\pi}{\Rightarrow} P'$ per rappresentare $P \overset{\tau^*\pi}{\Rightarrow} P'$. L'espressione $P \overset{\hat{\pi}}{\Rightarrow} P'$ sta per $P \overset{\pi}{\Rightarrow} P'$ se $\pi \in \mathbf{Act}\text{-}\{\tau\}$, e per $P \overset{\tau^*}{\Rightarrow} P'$

se $\pi = \tau$. Si noti che $P \stackrel{\hat{\tau}}{\Rightarrow} P'$ indica che possono essere eseguite 0 o più azioni τ, mentre $P \stackrel{\tau}{\Rightarrow} P'$ richiede che sia eseguita almeno un'azione interna τ.

Definizione 2.1 Una relazione $\mathcal{R} \subseteq \mathcal{G}_{nd} \times \mathcal{G}_{nd}$ è una *bisimulazione debole* se e solo se per ogni $(P, Q) \in \mathcal{R}$ e per ogni $\pi \in \mathbf{Act}$:

- $P \xrightarrow{\pi} P' \quad \Rightarrow \quad \exists\, Q' \in \mathcal{G}_{nd} : Q \stackrel{\hat{\pi}}{\Rightarrow} Q' \ \wedge\ (P', Q') \in \mathcal{R}$
- $Q \xrightarrow{\pi} Q' \quad \Rightarrow \quad \exists\, P' \in \mathcal{G}_{nd} : P \stackrel{\hat{\pi}}{\Rightarrow} P' \ \wedge\ (P', Q') \in \mathcal{R}$

Due termini $P, Q \in \mathcal{G}_{nd}$ sono equivalenti per bisimulazione debole, notazione $P \approx_B Q$, se e solo se esiste una bisimulazione debole che contiene la coppia (P, Q).

2.2 Non-deducibility on Composition

In questa sezione presentiamo una delle più importanti proprietà di information flow. L'intuizione comune alla base di queste proprietà è strettamente legata alla classica nozione della *Non-Interference* (in breve *NI*) [37], ovvero utenti di basso livello non devono essere in grado di dedurre alcuna informazione riguardo l'attività degli utenti di alto livello.

In [20] Focardi e Gorrieri classificano alcune proprietà di sicurezza basate sull'idea della non-interference, definendole in un modello simile a quello di *NDPA*. Tra queste, nella classe delle proprietà basate sulla bisimulazione debole, troviamo ad esempio la *BSNNI* e *BNNI*. Tali proprietà, formulate sull'idea che per ogni esecuzione del sistema che contiene azioni di alto livello esiste un'altra esecuzione senza azioni di alto livello ma con la stessa sottosequenza di azioni di basso livello, non sono però in grado di individuare alcune situazioni di stallo dovute ad attività di alto livello del sistema. Problemi di questo tipo vengono risolti dalla proprietà chiamata *Non Deducibility on*

Composition (*NDC* in breve), che è una delle proprietà più affermate e studiate. Questa proprietà si basa sull'idea che il comportamento osservabile di basso livello di un sistema isolato P non viene alterato quando si considerano tutte le possibili interazioni di P con agenti di alto livello dell'ambiente esterno. Introduciamo quindi la nozione dell'*NDC* nel contesto della bisimulazione e definiamo la proprietà di sicurezza *Bisimulation-based Non Deducibility on Composition* (*BNDC* in breve). Un sistema P è *BNDC* se, per ogni processo Π di alto livello, un utente di basso livello non è in grado di distinguere il sistema P dal sistema composto $((P \|_S \Pi) / S) \setminus \mathbf{AType_H}$. In altre parole, un sistema P è *BNDC* se ciò che un utente di basso livello riesce ad osservare dal sistema non viene in nessun modo modificato componendo il sistema P con qualsiasi processo di alto livello. Formalmente:

Definizione 2.2 *Bisimulation-based Non-deducibility on Composition*

$P \in BNDC \quad \Leftrightarrow \quad P \setminus \mathbf{AType_H} \approx_B ((P \|_S \Pi) / S) \setminus \mathbf{AType_H}$

$\forall \Pi \in \mathcal{G}_{Hnd}, S \subseteq \mathbf{AType_H}$

Questa definizione è compatibile con la definizione di *BNDC* data in [20], in cui l'unica differenza è dovuta all'utilizzo di un operatore di composizione parallela simile a quello del *CCS*. In particolare, dal momento che noi utilizziamo un operatore di composizione parallela *CSP*-like, dovremo in un primo momento mascherare le azioni di alto livello dell'insieme di sincronizzazione S, e quindi eliminiamo dal sistema le rimanenti azioni di alto livello.

Negli esempi che seguono assumiamo che un utente di basso livello che osserva il sistema conosca come il sistema è definito.

Esempio 2.1. Si consideri il sistema $P = l.\underline{0} + h.l'.\underline{0}$ che, in cui un utente di basso livello può osservare un'azione l oppure, se è stata eseguita prima l'azione di alto livello h, un'azione l'. Analizzando il sistema P,

assumendo che gli utenti di basso livello conoscano come il sistema P sia definito, si può facilmente notare che il sistema è insicuro. Infatti in base all'osservazione dell'azione l o dell'azione l' un utente di basso livello è in grado di stabilire se il sistema ha eseguito o meno l'azione di alto livello h. Formalmente si consideri il processo di alto livello $\Pi = h^*.\underline{0}$, avremo $((P\|_{\{h\}}\Pi) /\{h\}) \setminus \mathbf{AType_H} = l.\underline{0} + \tau.l'.\underline{0}$ mentre $P \setminus \mathbf{AType_H} = l.\underline{0}$; per cui $P \setminus \mathbf{AType_H} \not\approx_B ((P\|_{\{h\}}\Pi) /\{h\}) \setminus \mathbf{AType_H}$ e di conseguenza $P \notin BNDC$.

Esempio 2.2. Si consideri il sistema $P = l.P + h.\underline{0}$ in cui un possibile stallo può essere causato dall'esecuzione dell'azione di alto livello h. Conoscendo questo tipo di comportamento un utente di basso livello è in grado di stabilire che il sistema P non ha eseguito azioni di alto livello finché è in grado di osservare l'azione di basso livello l. Tale condizione è più che sufficiente per creare un flusso di informazione dal livello alto al livello basso. Formalmente abbiamo che $P \notin BNDC$, infatti, se consideriamo ancora il processo di alto livello $\Pi = h^*.\underline{0}$, avremo che:
$P \setminus \mathbf{AType_H} = l.P \not\approx_B l.P + \tau.\underline{0} = ((P\|_{\{h\}}\Pi) /\{h\}) \setminus \mathbf{AType_H}$.

Esempio 2.3. Si consideri quindi il sistema $P = l.P + h.l.P$. In questo caso si ha che $P \in BNDC$. Infatti da un punto di vista di basso livello si riesce ad osservare solamente l'esecuzione ripetuta dell'azione l, che non è sufficiente per ricavare informazioni sul comportamento del livello alto del sistema. Nessun processo di alto livello che interagisce con il sistema P è infatti in grado di modificare il comportamento osservabile di basso livello del sistema. Informalmente possiamo quindi osservare che:
$P \setminus \mathbf{AType_H} \approx_B ((P\|_h \Pi) /\{h\}) \setminus \mathbf{AType_H} \quad \forall \Pi \in \mathcal{G}_{Hnd}$.

Esempio 2.4. Si consideri il sistema $P = l.\underline{0} + h.h.l.\underline{0}$, in cui il processo di alto livello $\Pi = h^*.\underline{0}$ composto al sistema P genera uno stallo dopo l'esecuzione della prima azione h di P. Un utente di basso livello,

conoscendo come P è definito e conoscendo il processo Π con cui P è stato composto, non osservando alcuna azione l poiché il sistema è in stallo, è in grado di capire che il sistema P ha eseguito l'azione di alto livello h. Tale flusso di informazione viene catturato dalla proprietà *BNDC* infatti: $P \setminus \mathbf{AType_H} = l.\underline{0} \not\approx_B l.\underline{0} + \tau.\underline{0} = ((P||_{\{h\}} \Pi) / \{h\}) \setminus \mathbf{AType_H}$.

Usare in pratica la proprietà *BNDC* potrebbe non essere sempre un'operazione semplice da effettuare a causa della quantificazione universale sui processi di alto livello presente nella definizione. In [50] Martinelli dimostra che la proprietà *BNDC* è decidibile su processi a stati finiti (categoria in cui rientra la totalità dei sistemi reali). Nello stesso lavoro mostra però che se due sistemi sono *BNDC* la loro composizione può non essere *BNDC*. Non è quindi possibile dimostrare che un sistema complesso è *BNDC* analizzando i sotto-sistemi che lo compongono. In [20] Focardi e Gorrieri introducono quindi due nuove proprietà di sicurezza, più forti della *BNDC*, che trascendono la quantificazione universale sui processi di alto livello (rendendo queste proprietà più semplici da verificare) e che sono inoltre componibili, ovvero vengono preservate dagli operatori di composizione e restrizione (rendendo più semplice la verifica di queste proprietà su sistemi complessi). Tuttavia queste nuove proprietà (chiamate *SBSNNI* e *SBNDC*), essendo più restrittive della *BNDC*, portano talvolta a classificare insicuri dei sistemi che in realtà possono non dare mai luogo a flussi di informazione. Nel corso di quest'opera ci limiteremo all'utilizzo della *BNDC*, dal momento che la definizione delle proprietà *SBSNNI* e *SBNDC* non può essere utilizzata nel contesto dei protocolli crittografici, il cui studio è uno degli obiettivi principali di questo lavoro.

3 Il modello probabilistico

Le proprietà di sicurezza basate su un modello non-deterministico, come ad esempio la proprietà *BNDC* che abbiamo definito nel capitolo precedente, vengono usate per individuare flussi di informazione logici. In molti casi, tuttavia, l'osservazione del comportamento di un sistema dovrebbe tenere in considerazione aspetti addizionali, come ad esempio la distribuzione di probabilità delle azioni osservabili effettuate dal sistema. Analizzare il comportamento probabilistico di un sistema permette infatti di individuare canali di comunicazione coperti probabilistici che non vengono rilevati considerando solamente il comportamento non-deterministico. L'approccio probabilistico all'analisi del comportamento di un sistema permette inoltre di superare la prospettiva tradizionale in cui i sistemi vengono classificati come sicuri o come insicuri in maniera binaria. In particolare, assegnando un valore probabilistico ad ogni possibile flusso di informazione che potrebbe rivelarsi, è possibile stabilire il livello di sicurezza di un sistema. In pratica, per considerare il sistema sufficientemente sicuro, si controlla che la probabilità di osservare un flusso di informazione insicuro sia inferiore ad una certa soglia prestabilita. Inoltre modellando il comportamento probabilistico di un sistema è possibile sia analizzarne le proprietà di sicurezza e i flussi di informazione, sia effettuarne una valutazione delle prestazioni. Sottolineiamo infine che il modello probabilistico preserva le caratteristiche non-deterministiche del sistema.

In particolare, la *non deducibility on composition probabilistica* (la proprietà di information flow che definiremo nel modello probabilistico) è un'estensione conservativa della *BNDC* definita nel contesto non-deterministico.

Il resto di questo capitolo è suddiviso in tre paragrafi. Nel primo definiremo il modello probabilistico che utilizzeremo per specificare il comportamento probabilistico dei sistemi. Introdurremo quindi la sintassi e la semantica formale dell'algebra di processi probabilistica. Nel secondo paragrafo definiremo poi un'estensione probabilistica della bisimulazione debole. Infine nel terzo paragrafo specifichiamo la versione probabilistica della proprietà *BNDC*.

3.1 Un linguaggio probabilistico

In questa sezione viene presentato il linguaggio formale probabilistico che utilizzeremo per analizzare il comportamento probabilistico di sistemi. Questo linguaggio, che è una variante dell'algebra di processi proposta in [14, 16, 15], si ottiene estendendo il linguaggio mostrato nella sezione **2.1** nel seguente modo:

1. Si aggiunge un'informazione probabilistica agli operatori.
2. Si definisce un modello di probabilità sulla base del modello generativo-reattivo proposto in [13].

Il modello probabilistico che verrà utilizzato è quindi una variante dell'approccio generativo-reattivo [13], ed è costruito utilizzando un tipo di sincronizzazione *asimmetrica* in cui un processo che effettua un'azione generativa può sincronizzarsi solo con un processo che esegue un'azione reattiva, mentre processi che si comportano entrambi in maniera reattiva possono sincronizzarsi tra loro. L'integrazione con il modello generativo-reattivo si ottiene in maniera estremamente naturale considerando le azioni di output come azioni di tipo generativo, le azioni di input come azioni di tipo reattivo e quindi imponendo che azioni di tipo generativo si sincronizzino solamente con azioni di tipo reattivo. Un

processo che si comporta in maniera generativa decide autonomamente, sulla base di una distribuzione di probabilità, quale azione sarà eseguita. Un processo che si comporta in maniera reattiva reagisce invece internamente ad un'azione generativa o ad un'azione reattiva di tipo a eseguita da un componente dall'ambiente in cui si trova in base alla distribuzione di probabilità associata alle azioni reattive di tipo di a che può eseguire. Dal momento che la scelta del tipo di azione che verrà eseguita è completamente non-deterministica e dipende dall'ambiente, le azioni reattive sono considerate come una sorta di azioni incomplete, che debbono sincronizzarsi con azioni generative di altri sistemi per ottenere un *sistema chiuso*. Si dice quindi che un sistema è *pienamente specificato* quando dà origine ad un sistema di transizioni probabilistico puramente generativo, che non ammette cioè azioni reattive. Un sistema pienamente specificato è quindi un sistema puramente probabilistico da cui, estraendo le azioni dall'albero delle transizioni, si può ricavare una catena di Markov che potrà poi essere facilmente analizzata per ottenere informazioni sulle prestazioni del sistema.

3.1.1 *GRTS*

Un sistema di transizioni generative-reattive (*GRTS*) [14, 15] è costituito da transizioni etichettate con un'azione (di tipo generativo o reattivo) e corredate da una misura di probabilità.

In accordo con la scelta di considerare le azioni di input come azioni reattive e le azioni di output come azioni generative, denotiamo con $\mathbf{RAct} = I$ e $\mathbf{GAct} = O \cup \{\tau\}$ rispettivamente gli insiemi di azioni reattive e generative. Si noti che l'azione τ è considerata un'azione generativa dal momento che rappresenta una transizione interna autonomamente eseguita da un processo, senza alcuna interazione con l'ambiente esterno. L'insieme delle azioni diviene quindi $\mathbf{Act} = \mathbf{RAct} \cup \mathbf{GAct}$ e gli elementi di tale insieme verranno rappresentati nella forma $\pi, \pi', \ldots$.

Tutte le transizioni possibili in un determinato stato del *GRTS* sono raggruppate in *bundle*. Per ogni stato si ha un solo bundle

generativo, che contiene tutte le transizioni etichettate da azioni di tipo generativo che è possibile effettuare in quello stato, e diversi bundle reattivi, ciascuno dei quali è riferito ad un particolare tipo di azione a e contiene quindi tutte le transizioni etichettate con l'azione a^*.

Un bundle dà origine ad una scelta probabilistica tra le transizioni che lo compongono; la scelta tra i vari bundle avviene invece in maniera non-deterministica.

Definizione 3.1. Un sistema di transizioni generative-reattive (*GRTS*) è una terna del tipo (**S**, **AType**, T) in cui:

- **S** è un insieme di stati.
- **AType** è un insieme di *tipi* di azioni.
- $T \in \mathcal{M}$ (**S** × **Act** ×]0, 1] × **S**) è un multiinsieme[3] di transizioni probabilistiche tale che:

  1. $\forall\, s \in \mathbf{S},\ \forall\, a^* \in \mathbf{RAct},\ \sum\{|\ p \mid \exists\, t \in \mathbf{S} : (s, a^*, p, t) \in \mathbf{T}\ |\} \in \{0, 1\}$

  2. $\forall\, s \in \mathbf{S},\ \sum\{|\ p \mid \exists\ a \in \mathbf{GAct}, t \in \mathbf{S} : (s, a, p, t) \in \mathbf{T}\ |\} \in \{0, 1\}$

In cui la condizione **1.** regola la distribuzione delle probabilità tra le transizioni dei bundle di tipo reattivo, mentre la condizione **2.** vincola, per ogni stato, il bundle di tipo generativo. Entrambe le condizioni stabiliscono infatti che per ogni bundle *non vuoto* la somma delle probabilità delle transizioni che lo compongono vale 1 (in caso contrario la sommatoria su multiinsiemi vuoti è definita uguale a 0).
Un *GRTS con radice* è una tupla del tipo (**S**, **AType**, **T**, s_0), in cui $s_0 \in \mathbf{S}$ è lo stato iniziale e (**S**, **AType**, **T**) è un *GRTS*.

Nell'esempio in fig. 3.1 viene mostrato un *GRTS* composto da un bundle generativo che contiene le tre transizioni g, g', g'', da un bundle reattivo contenente transizioni etichettate da azioni di tipo a che ammette due transizioni a^*, e un ultimo bundle reattivo di tipo b contenente

[3] Utilizziamo i simboli "{|" e "|}" per rappresentare le parentesi associate ad un multiinsieme e la notazione $\mathcal{M}(S)$ per indicare una collezione di multiinsiemi sull'insieme S.

un'unica transizione b^*. Le transizioni che appartengono ad uno stesso bundle sono rappresentate graficamente raggruppandole con un arco. La probabilità di una transizione può quindi essere omessa se uguale ad 1.

Sottolineiamo infine che un sistema pienamente specificato (ovvero un sistema che non contiene bundle di tipo reattivo) è *performance closed*, nel senso che dà origine ad un sistema di transizioni totalmente probabilistico che non permette scelte non-deterministiche.

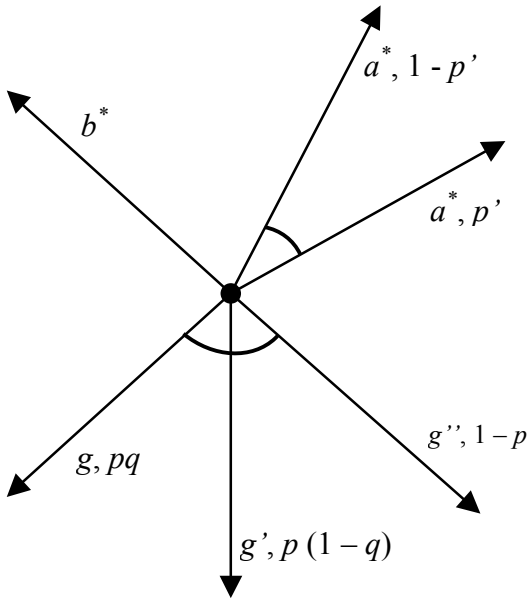


Fig. 3.1. Esempio di *GRTS*.

3.1.2 Probabilistic Process Algebra

In questa sezione viene presentata la sintassi del linguaggio che chiameremo *Probabilistic Process Algebra* (*PPA* in breve), che come abbiamo detto è una variante dell' algebra di processi proposta in [14, 16, 15], in cui non si utilizza l'operatore di ridenominazione. Sia quindi $\mathcal{L}$ l'insieme dei termini del calcolo rappresentati da P, Q, ... generato dalla seguente grammatica:

$$P ::= \underline{0} \mid \pi.P \mid P +^{p} P \mid P \|_{S}^{p} P \mid P \setminus L \mid P/_{a}^{p} \mid A$$

In cui $p \in]0, 1[$ è una probabilità, $S,L \subseteq$ **AType**-$\{\tau\}$, $a \in$ **AType**-$\{\tau\}$.

Indichiamo con $\mathcal{G}$ l'insieme dei termini di $\mathcal{L}$ guardati e chiusi [19], e con $\mathcal{G}_H = \{P \in \mathcal{G} \mid sort(P) \subseteq \mathbf{AType_H}\}$ l'insieme dei processi di alto livello, che eseguono cioè solamente azioni di alto livello. Sottolineiamo che da qualsiasi termine $P \in \mathcal{L}$ è possibile derivare il corrispondente termine, definito nel contesto non-deterministico, $P_{nd} \in \mathcal{L}_{nd}$ semplicemente rimovendo i parametri probabilistici dagli operatori in P.

Vediamo ora nel dettaglio i vari operatori del linguaggio:

0 indica un termine che non può effettuare nessuna altra transizione. Potrebbe rappresentare un processo terminato o che si trovi in uno stato di stallo[4].

$\pi.P$ è l'operatore di prefisso che esegue l'azione π con probabilità 1 e quindi si comporta come il processo P.

$P +^p Q$ rappresenta l'operatore di composizione alternativa. Questo operatore dà origine ad una scelta probabilistica tra le azioni generative di P e Q o tra le loro azioni reattive dello stesso tipo. In particolare distinguiamo i seguenti casi:

1. Se sia P sia Q possono eseguire azioni generative allora il termine $P +^p Q$ esegue un'azione generativa di P con probabilità p oppure un'azione generativa di Q con probabilità $1 - p$.
2. Se uno dei due processi P e Q non è in grado di eseguire un'azione generativa l'operatore $P +^p Q$ esegue un'azione generativa dell'altro processo con probabilità 1.
3. Se prendiamo invece in considerazione azioni reattive di un tipo a si avrà che il termine $P +^p Q$, se sia P sia Q possono eseguire azioni a^*, esegue un'azione reattiva a^* tra quelle di P con

[4] Generalmente abbrevieremo i termini $P.\underline{0}$ omettendo il processo $\underline{0}$ e scrivendo semplicemente P.

probabilità p oppure una tra quelle di Q con probabilità $1 - p$.

4. La scelta tra azioni reattive e generative o la scelta tra azioni reattive di tipi diversi è puramente non-deterministica.

Nella fig. 3.2 sono mostrati alcuni esempi. In fig. 3.2 (a) mostriamo i *GRTS* generati dai termini $a +^p b$ e $b^* +^p b^*$ che rappresentano una scelta puramente probabilistica fatta in base alla probabilità p.
In fig. 3.2 (b) vengono mostrati i *GRTS* generati dai due termini $a +^p b^*$ e $a^* +^p b^*$ che rappresentano invece scelte puramente non-deterministiche in cui il valore di p viene quindi omesso. Infine nella fig. 3.2 (c) viene mostrato un modello misto in cui vengono effettuate scelte probabilistiche e scelte non-deterministiche insieme; il termine che genera il *GRTS* in fig. 3.2 (c) è $(a +^{p'} b^*) +^p (b +^{p''} b^*)$ in cui le due probabilità p' e p'' non vengono prese in considerazione perché si trovano in un contesto in cui viene effettuata una scelta non-deterministica.

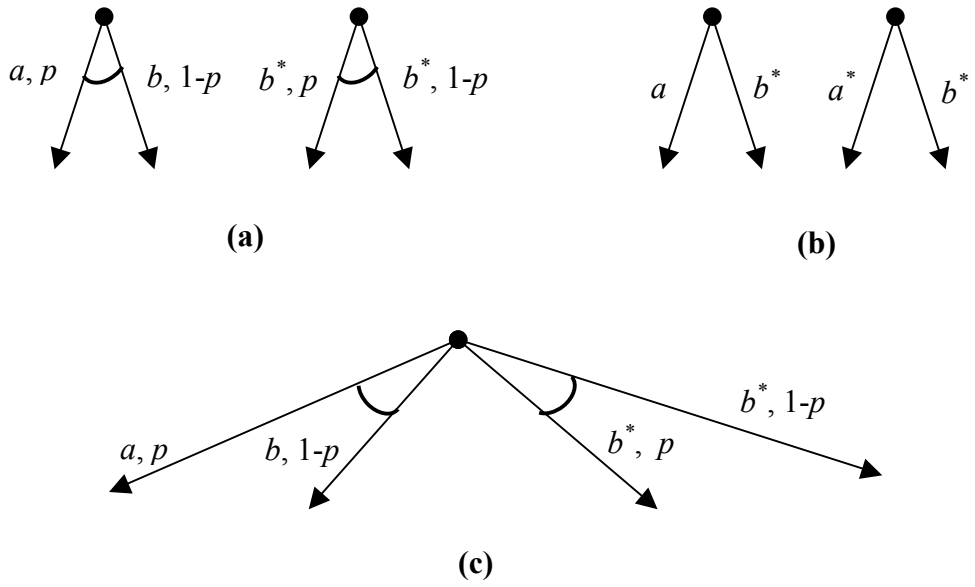


Fig. 3.2. Esempi di *GRTS* derivati dall'operatore di composizione alternativa.

$\boldsymbol{P} \|_S^p \boldsymbol{Q}$ rappresenta l'operatore di composizione parallela, è basato su una politica di sincronizzazione simile a quella usata nel CSP, in cui, dato l'insieme $S \subseteq \mathbf{Atype}\text{-}\{\tau\}$, i termini P e Q sono forzati a sincronizzarsi sulle azioni il cui tipo si trova in S e ad eseguire localmente le altre azioni. Due azioni di tipo a possono sincronizzarsi solamente se sono entrambe due azioni reattive a^* (producendo come risultato un'azione reattiva a^*), oppure se una delle due azioni è un'azione generativa a e l'altra è un'azione reattiva a^* (producendo come risultato un'azione generativa a). In pratica, quindi, le azioni generative di P (Q) che possono essere effettuate da $P \|_S^p Q$ sono quelle il cui tipo a non è contenuto in S, oppure quelle il cui tipo a è contenuto in S e per le quali il processo Q (P) può effettuare un'azione reattiva a^*.
La scelta tra le azioni generative di tipo $a \notin S$, eseguibili da P e Q viene fatta in base alla probabilità p con lo stesso meccanismo visto per l'operatore di composizione alternativa. Lo stesso avviene se si considerano azioni reattive di un tipo $a \notin S$, $P \|_S^p Q$ può effettuare tutte le azioni reattive a^* eseguibili da P e da Q e la scelta tra i due viene fatta ancora una volta in base alla probabilità p. Distinguiamo poi i seguenti casi:

1. Se l'operatore $P \|_S^p Q$ effettua delle restrizioni sulle azioni generative eseguibili da P (Q), in accordo con il modello generativo [13], le probabilità di eseguire le azioni rimanenti di P (Q) debbono essere ridistribuite proporzionalmente in modo che la somma totale delle loro nuove probabilità sia ancora uguale ad 1.
2. Nel caso in cui per le azioni generative a di P (Q) avvenga sincronizzazione, la loro probabilità viene distribuita tra le diverse azioni a che si ottengono sincronizzandosi con le varie azioni reattive a^* eseguibili da Q (P), in rapporto alla probabilità con cui queste vengono scelte in Q (P).

3. Nel caso in cui avvenga sincronizzazione per azioni reattive di un tipo $a \in S$ e se sia P sia Q possono eseguire azioni reattive a^*, la scelta delle due azioni a^* (una di P e una di Q), che per la sincronizzazione danno origine all'azione a^* eseguita da $P \|_S^p Q$, è fatta in base alla probabilità che tali azioni vengano scelte *localmente* da P e da Q.

Nella fig. 3.3 mostriamo quindi alcuni *GRTS* derivati da operazioni di composizione parallela.

1. In fig. 3.3 (a) viene rappresentato graficamente il *GRTS* ottenuto dal termine $((a +^q b) +^{q'} c^*) \|_{\{c\}}^p c$ che può eseguire, senza sincronizzazione, le due azioni generative a e b del processo a sinistra, oppure, sincronizzando, l'azione generativa c del processo di destra (si noti che l'azione generativa c del termine di destra, pur essendo di un tipo presente in S**,** è eseguibile poiché il termine a sinistra può eseguire un'azione c^* e quindi sincronizzarsi). Come nel caso della composizione alternativa poiché entrambi i processi possono effettuare azioni generative avremo che il termine di sinistra eseguirà un'azione generativa con probabilità p mentre il termine di destra ha probabilità 1- p di eseguire una sua azione generativa. Poiché le azioni generative eseguibili dal termine di sinistra $((a +^q b) +^{q'} c^*)$ sono a, con probabilità q, e b, con probabilità 1- q (mentre non viene considerata la probabilità q' perché come abbiamo visto per la composizione alternativa si trova nel contesto di una scelta non-deterministica), il sistema che si ottiene dopo la composizione parallela eseguirà un'azione a con una probabilità $p \cdot q$, un'azione b con probabilità $p \cdot (1-q)$ o un'azione c con probabilità $1-p$.
2. In fig. 3.3 (b) è invece rappresentato graficamente il *GRTS* derivato dal termine $(a +^r b) \|_{\{a,b\}}^p (b^*.c +^q b^*.d)$ in cui solo il processo dalla parte sinistra può eseguire azioni generative e

quindi il parametro p non viene in questo caso considerato. Inoltre poiché il processo di sinistra non può effettuare l'azione a (non riesce a sincronizzare col processo di destra che non effettua mai azioni a^*) potrà essere eseguita solamente l'azione generativa b con probabilità 1, e quindi neanche il parametro r è considerato. Tale azione permette poi la sincronizzazione con una delle due azioni b^* del processo di destra scelta in base alla probabilità q. In pratica potrebbe essere eseguita un'azione b che porta il sistema nella forma $\underline{0} \parallel^{p}_{\{b\}} c$ con probabilità q, oppure un'azione b che porta il sistema nello stato $\underline{0} \parallel^{p}_{\{b\}} d$ con probabilità 1-q.

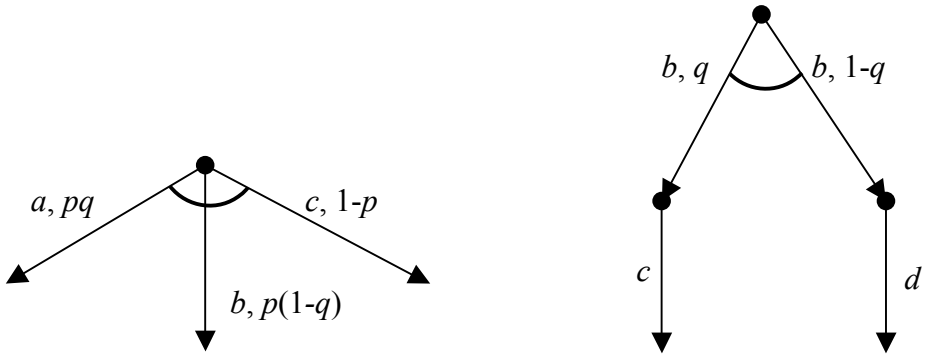


Fig. 3.3. Esempi di *GRTS* derivati dall'operatore di composizione parallela.

$\boldsymbol{P} \setminus \boldsymbol{L}$ in cui $L \subseteq$ **Atype**-$\{\tau\}$, è l'operatore di restrizione. Questo operatore previene l'esecuzione di tutte le azioni il cui tipo è in L. Questo operatore era stato introdotto da Aldini in [1] per rendere più semplice la definizione di alcune proprietà di sicurezza. In effetti, come abbiamo visto nella sezione **2.1.1**, tale operatore potrebbe essere facilmente ottenuto attraverso l'operatore di composizione parallela con il processo $\underline{0}$, infatti $P \setminus L = P \parallel^{p}_{L} \underline{0}$ per qualsiasi valore di $p \in]0,1[$.

$\boldsymbol{P}/^{p}_{a}$ in cui $a \in$ **AType** e p è ancora una probabilità, indica l'operatore di

mascheramento. Questo operatore trasforma le azioni generative e reattive del tipo a nell'azione generativa interna τ, in accordo con le seguenti regole:

Se si prendono in considerazione le azioni generative eseguibili da $P/_a^p$ si distinguono i seguenti casi:

1. Se P può eseguire sia azioni generative sia azioni reattive a^*, allora $P/_a^p$ esegue un'azione generativa τ (ottenuta mascherando un'azione reattiva a^* di P) con probabilità p oppure esegue una delle altre azioni generative possibili con probabilità 1-p. Attraverso questa regola l'operatore di mascheramento non introduce non-determinismo tra le azioni generative preservando la definizione di *GRTS*.
2. Se P non può eseguire azioni generative o se P non esegue mai azioni reattive a^* il parametro p non viene considerato.

Se si prendono in considerazione le azioni reattive, $P/_a^p$ può eseguire le stesse azioni reattive di tipo $b \neq a$ eseguibili da P (e con la stessa distribuzione di probabilità).

Il parametro probabilistico p rappresenta quindi la probabilità che le azioni τ ottenute mascherando le azioni reattive a^* di P siano eseguite rispettando le azioni generative permesse da P prima del mascheramento. Se ad esempio consideriamo il termine $P \triangleq a^* +^q b$ in cui viene effettuata una scelta puramente non-deterministica (il parametro q è irrilevante) e gli applichiamo il mascheramento di azioni di tipo a, otteniamo il termine $P/_a^p \triangleq \tau +^p b$ in cui si ha una scelta probabilistica, effettuata in accordo alle probabilità p e 1-p, tra l'azione generativa interna τ, ottenuta mascherando l'azione reattiva a^* , e l'azione generativa b. In questo modo il parametro probabilistico p garantisce che l'operatore $P/_a^p$ non introduca non-determinismo tra azioni generative, ed in pratica non viene usato quando il mascheramento è effettuato su azioni generative dal

momento che la scelta tra azioni generative è già probabilistica.
La ragione per cui si è deciso di trasformare azioni reattive a^* in azioni generative τ è legata al fatto che uno degli obiettivi dell'operatore di mascheramento è quello di ottenere sistemi pienamente specificati partendo da *sistemi aperti* (ovvero sistemi che permettono transizioni reattive). In particolare, quando un sistema è pienamente specificato, le scelte non-deterministiche dovute all'interazione con l'ambiente sono state risolte e il parametro p le trasforma quindi in scelte puramente probabilistiche. Formalmente il mascheramento di un'azione reattiva a^* del processo P corrisponde all'esecuzione di una sincronizzazione tra l'azione a^* ed un'azione generativa esterna a che dà origine ad un'azione interna τ, e la cui distribuzione di probabilità dipende dal parametro p. Analizzando le proprietà di sicurezza che utilizzano l'operatore di mascheramento mostreremo che il comportamento di basso livello di un sistema sicuro non dipende dalla scelta del parametro p.

Se si considera ad esempio il processo $P \triangleq h^*.P +^q l.P$, in cui la scelta è puramente non-deterministica, e si mascherano le sue azioni di alto livello, si ottiene il processo $P/_h^p = \tau.P +^p l.P$ in cui la scelta è puramente probabilistica. Come abbiamo detto il mascheramento dell'azione reattiva h^* viene visto come una sincronizzazione tra l'azione h^* ed un'azione generativa esterna h che dà origine ad un'azione generativa interna τ. Come vedremo nel seguito il processo P è sicuro (nel senso che non dà origine a flussi di informazione) se la distribuzione di probabilità delle azioni mascherate non è in grado di modificare il comportamento probabilistico del processo dal punto di vista delle azioni di basso livello. Ciò significa che la probabilità di esecuzione dell'azione generativa esterna h (che si sincronizza con l'azione reattiva interna h^*) non deve essere rilevante, nel senso che non deve alterare il comportamento probabilistico del sistema da un punto di vista di basso livello. In pratica si controlla che, in qualsiasi modo venga effettuata dall'ambiente la scelta non-deterministica in $P \triangleq h^*.P +^q l.P$, il

comportamento probabilistico di basso livello del processo (definito dall'esecuzione dell'azione l con una certa probabilità) rimanga inalterato.

A definisce infine un processo costante. Questo operatore viene utilizzato per specificare sistemi ricorsivi. Generalmente quando si definisce un sistema attraverso un'algebra di processi viene infatti specificato anche un insieme di termini costanti nella forma $A \triangleq P$, in cui P è un termine guardato e chiuso [19].

Di seguito specifichiamo alcune assunzioni:

- Per evitare ambiguità introduciamo la seguente relazione di precedenza tra gli operatori: prefisso > restrizione = mascheramento > composizione alternativa > composizione parallela.
- Inoltre consideriamo il parametro $p = ½$ quando viene omesso.
- Stabiliamo che l'insieme S, nell'operatore di composizione parallela, è uguale all'insieme vuoto se non viene specificato.
- Utilizziamo l'abbreviazione $P/^{p_1 \cdots p_n}_{a_1 \ldots a_n}$ per rappresentare l'espressione $P/^{p_1}_{a_1} \ldots /^{p_n}_{a_n}$ che maschera azioni di tipi diversi.
- Sia $L = \{a_1, \ldots, a_n\} \subseteq$ **AType** - $\{\tau\}$ e $a_1, \ldots, a_n$ una sequenza di azioni. Quando ogni termine in $Der(P)$ non permette azioni reattive il cui tipo è in L, la formula P/L sta per $P/_{a_1 \ldots a_n}$. In questo caso, infatti, i parametri probabilistici per l'operatore di mascheramento sono ininfluenti e in particolare: $P/_{a_1 \ldots a_n} = P/^{p_1 \cdots p_n}_{a_1 \ldots a_n}$ $\forall p_1, \ldots, p_n \in]0,1[$.

3.1.3 La semantica formale

La semantica formale di questo calcolo effettua una mappatura dei termini del calcolo in *GRTS* in cui ogni transizione è caratterizzata (ed

etichettata) da un'azione e da una probabilità.
Nella definizione delle regole di semantica operazionale utilizziamo le seguenti notazioni:

- una transizione $P \xrightarrow{\pi,p} P'$ indica che il processo P esegue un'azione π con probabilità p e poi si comporta come il processo P'
- $P \xrightarrow{\pi} P'$ indica invece che $\exists\, p : P \xrightarrow{\pi,p} P'$, ovvero che il processo P può eseguire un'azione π per poi comportarsi come il processo P'
- $P \xrightarrow{\pi}$ indica invece che $\exists\ P' : P \xrightarrow{\pi} P'$, ovvero che il processo P può eseguire un'azione π
- $P! \xrightarrow{\pi}$ indica invece che $\nexists\ P' : P \xrightarrow{\pi} P'$, ovvero che il processo P non esegue mai l'azione π
- $P \xrightarrow{G}$, dato $G \subseteq \mathbf{AType}$, indica che $\exists\, a \in G : P \xrightarrow{a}$, ovvero che il processo P può eseguire un'azione generativa il cui tipo è incluso nell'insieme G
- $P! \xrightarrow{G}$, dato $G \subseteq \mathbf{AType}$, indica che $\forall a \in G\ P! \xrightarrow{a}$, ovvero che il processo P non esegue mai un'azione generativa il cui tipo è incluso nell'insieme G.

Siano poi gli insiemi $G_{S,P}$, $G_L \subseteq \mathbf{AType}$, con S, $L \subseteq \mathbf{AType} - \{\tau\}$ e $P \in \mathcal{G}$, definiti nel seguente modo:

$$G_{S,P} = \{a \in \mathbf{AType} \mid a \notin S \vee (a \in S \wedge P \xrightarrow{a^*})\}$$
$$G_L = \{a \in \mathbf{AType} \mid a \notin L\}$$

In particolare $G_{S,Q}$ ($G_{S,P}$) è l'insieme dei tipi delle transizioni generative di P (Q) eseguibili da $P \parallel_S^p Q$,mentre G_L è l'insieme dei tipi delle transizioni generative di P eseguibili da $P \setminus L$. Poiché si effettua una restrizione su un insieme di azioni eseguibili occorre anche ridistribuire

le probabilità delle transizioni generative di P (Q) eseguibili da $P \|_S^p Q$ e $P \setminus L$ in modo che la somma totale delle probabilità sia uguale ad 1 [13]. A questo scopo, nella definizione delle regole semantiche, viene introdotta la funzione $\nu_P(G) : \mathcal{P}(\mathbf{AType}) \rightarrow]0,1]$, in cui $P \in \mathcal{G}$, definita nel seguente modo: $\nu_P(G) = \sum \{| p \mid \exists P' \in \mathcal{G}, a \in G : P \xrightarrow{a,p} P' |\}$; questa funzione calcola la somma delle probabilità delle transizioni generative il cui tipo è incluso in G eseguibili da P. In particolare $\nu_P(G_{S,Q})$ ($\nu_Q(G_{S,P})$) calcola la somma delle probabilità delle transizioni generative di P (Q) eseguibili da $P \|_S^p Q$, e viene usata per normalizzare le probabilità delle transizioni generative di P (Q). Analogamente $\nu_P(G_L)$ calcola la somma delle probabilità delle transizioni generative di P eseguibili da $P \setminus L$, e ancora può essere usata per normalizzare le probabilità delle transizioni generative di P.

La semantica formale dell'algebra di processi è data dal sistema di transizioni generative-reattive ($\mathcal{G}$, **AType**, T) i cui stati sono i termini del calcolo e la relazione delle transizioni T è il minimo multiinsieme che soddisfa le regole operazionali specificate nel seguito. Le regole contrassegnate dal simbolo **(L)** fanno riferimento a transizioni del processo di sinistra P. Per semplicità abbiamo omesso le regole simmetriche che tengono in considerazione le transizioni del processo di destra Q; queste si possono facilmente ottenere scambiando i ruoli dei termini P e Q e sostituendo p con 1-p nell'etichetta delle transizioni derivate. Vediamo adesso nel dettaglio, per ogni operatore del linguaggio, le regole di semantica operazionale che permettono la mappatura dei termini del nostro linguaggio in *GRTS*.

Prefisso

$$\frac{-}{\pi.P \xrightarrow{\pi,1} P}$$

Composizione alternativa

$$\textbf{(L)}\ \frac{P \xrightarrow{a^*,q} P' \quad Q \xrightarrow{a^*}}{P +^p Q \xrightarrow{a^*,p\cdot q} P'} \qquad \textbf{(L)}\ \frac{P \xrightarrow{a^*,q} P' \quad Q\,! \xrightarrow{a^*}}{P +^p Q \xrightarrow{a^*,q} P'}$$

$$\textbf{(L)}\ \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{\textbf{GAct}}}{P +^p Q \xrightarrow{a,p\cdot q} P'} \qquad \textbf{(L)}\ \frac{P \xrightarrow{a,q} P' \quad Q\,! \xrightarrow{\textbf{GAct}}}{P +^p Q \xrightarrow{a,q} P'}$$

Composizione parallela

$$\textbf{(L)}\ \frac{P \xrightarrow{a^*,q} P' \quad Q \xrightarrow{a^*}}{P \|^p_S Q \xrightarrow{a^*,p\cdot q} P' \|^p_S Q}\ a \notin S \qquad \textbf{(L)}\ \frac{P \xrightarrow{a^*,q} P' \quad Q\,! \xrightarrow{a^*}}{P \|^p_S Q \xrightarrow{a^*,q} P' \|^p_S Q}\ a \notin S$$

$$\frac{P \xrightarrow{a^*,q} P' \quad Q \xrightarrow{a^*,q'} Q'}{P \|^p_S Q \xrightarrow{a^*,q\cdot q'} P' \|^p_S Q'}\ a \in S$$

$$\textbf{(L)}\ \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,p\cdot q/\nu_P(G_{S,Q})} P' \|^p_S Q}\ a \notin S$$

$$\textbf{(L)}\ \frac{P \xrightarrow{a,q} P' \quad Q\,! \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,q/\nu_P(G_{S,Q})} P' \|^p_S Q}\ a \notin S$$

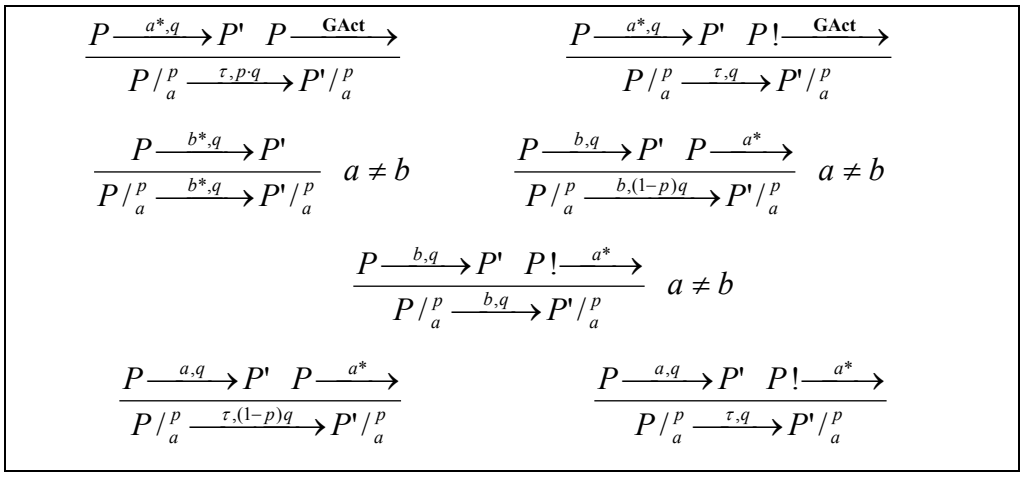
$$\textbf{(L)}\ \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a^*,q'} Q' \quad Q \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,p\cdot q\cdot q'/\nu_P(G_{S,Q})} P' \|^p_S Q'}\ a \in S$$

$$\textbf{(L)}\ \frac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a^*,q'} Q' \quad Q\,! \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,q\cdot q'/\nu_P(G_{S,Q})} P' \|^p_S Q'}\ a \in S$$

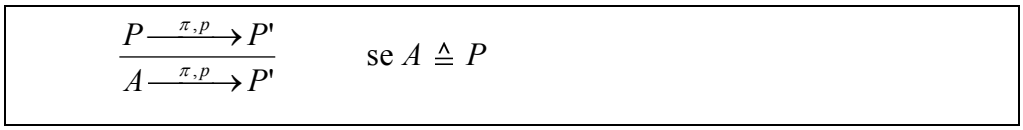
Restrizione

$$\frac{P \xrightarrow{a^*,q} P'}{P \setminus L \xrightarrow{a^*,q} P' \setminus L}\ a \notin L \qquad \frac{P \xrightarrow{a,q} P'}{P \setminus L \xrightarrow{a,q/\nu_P(G_L)} P' \setminus L}\ a \notin L$$

Mascheramento



Costante



Teorema. Se P è un processo di $\mathcal{G}$ allora la semantica operazionale di P è un *GRTS* [14, 15].

3.2 Bisimulazione debole probabilistica

In questa sezione definiamo un'equivalenza di bisimulazione debole probabilistica, come proposta in [45], nel contesto dei sistemi di transizioni generative-reattive. L'idea di base consiste nel sostituire le classiche transizioni deboli $s \stackrel{\hat{\pi}}{\Rightarrow} s'$ (mostrate nel paragrafo **2.1.2**) con la probabilità di raggiungere dallo stato s una classe di stati equivalenti a s' attraverso una sequenza di transizioni etichettate con stringhe nella forma $\tau^* \hat{\pi} \tau^*$. Prima di introdurre questo nuovo tipo di equivalenza mostriamo alcune nozioni di base sulla teoria della probabilità (si veda ad esempio [46]).

Definizione 3.2. Si definisce *spazio delle probabilità* la tupla $(\Omega, \mathcal{F}, P)$ in cui:

- Ω, lo *spazio degli esempi*, è un insieme non vuoto di elementi chiamati *sample points*.
- $\mathcal{F}$, il *campo sigma*, è una collezione non vuota di sottoinsiemi di Ω, chiamati *eventi*, chiusa rispetto alla complementazione ed all'unione contabile.
- P, la *misura di probabilità*, assegna un numero reale $P(F)$ ad ogni evento F del campo sigma tale che:

$$(i) \quad P(F) \geq 0 \qquad \forall F \in \mathcal{F}$$

$$(ii) \quad P(\Omega) = 1$$

$$(iii) \quad \bigcap_{i=1}^{n} F_i = \varnothing \quad \Rightarrow \quad P(\bigcup_{i=1}^{n} F_i) = \sum_{i=1}^{n} P(F_i)$$

$$(iv) \quad \bigcap_{i=1}^{\infty} F_i = \varnothing \quad \Rightarrow \quad P(\bigcup_{i=1}^{\infty} F_i) = \sum_{i=1}^{\infty} P(F_i)$$

Si consideri ora il *GRTS* con radice $(S, \mathbf{AType}, T, s_0)$. Definiamo quindi la funzione Pr: $S \times \mathbf{Act} \times S \rightarrow [0, 1]$ come:

$$Pr(s, \pi, s') = \sum \{| p \mid (s, \pi, p, s') \in T |\}$$

e, dato $C \subseteq S$,

$$Pr(s, \pi, C) = \sum_{s' \in C} Pr(s, \pi, s').$$

Un *frammento di esecuzione* è una sequenza finita σ del seguente tipo: $\sigma = s_0 \xrightarrow{\pi_1, p_1} s_1 \xrightarrow{\pi_1, p_2} \ldots s_k$ tale che $(s_{i-1}, \pi_i, p_i, s_i) \in T$ per ogni $i \in \{1, \ldots, k\}$. Chiamiamo quindi $tr(\sigma)$ la sequenza di azioni $\pi_1, \pi_2, \ldots, \pi_k$ e diciamo che σ è *massimale* se e solo se l'ultimo stato di σ, ovvero lo stato s_k, è terminale.
Ora indichiamo con $Pr(\sigma) = Pr(s_0, \pi_1, s_1) \cdot \ldots \cdot Pr(s_{k-1}, \pi_\kappa, s_k)$ la probabilità del frammento di esecuzione σ.

Per *esecuzione* intendiamo o un frammento di esecuzione massimale oppure una sequenza infinita $\sigma' = s_0 \xrightarrow{\pi_1, p_1} s_1 \xrightarrow{\pi_1, p_2} \ldots$ in

cui $(s_{i-1}, \pi_i, p_i, s_i) \in T$ per ogni $i \in \mathrm{N}$.

Denotiamo con $tr_k(\sigma')$ la sequenza di azioni $\pi_1, \pi_2, \ldots, \pi_k$.
Infine denotiamo con $\sigma\uparrow$ l'insieme delle esecuzioni $\{| \sigma' \mid \sigma <_{prefix} \sigma' |\}$ in cui $<_{prefix}$ è l'usuale relazione di prefisso su sequenze.

A questo punto, dato uno stato iniziale $s \in S$, definiamo uno spazio delle probabilità sull'insieme delle esecuzioni che iniziano dallo stato s. Siano quindi $Exec(s)$ l'insieme di tutte le esecuzioni che iniziano con lo stato s e $ExecFrag(s)$ l'insieme di tutti i frammenti di esecuzione che iniziano con lo stato s.

Chiamiamo $\sum(s)$ il più piccolo campo sigma sull'insieme $Exec(s)$ che contiene tutti gli insiemi $\sigma\uparrow$ dato $\sigma \in ExecFrag(s)$.

La misura di probabilità *Prob* è l'unica misura sul campo sigma $\sum(s)$ tale che $Prob(\sigma\uparrow) = Pr(\sigma)$.

Siano $C \subseteq \mathcal{G}$, $P \in \mathcal{G}$, e $A \subseteq \mathbf{Act}^*$, indichiamo con $Exec(A, C)$ l'insieme delle esecuzioni che, partendo dal loro stato iniziale e attraverso una sequenza di azioni in A, conducono ad uno stato in C. Formalmente:
$\sigma' \in Exec(\mathrm{A}, \mathrm{C}), \sigma' = s_0 \xrightarrow{\pi_1, p_1} s_1 \xrightarrow{\pi_2, p_2} \ldots \Leftrightarrow \exists\, k : tr_k(\sigma') \in A \wedge s_k \in C$.
Sia quindi $Exec(P, A, C) = Exec(A, C) \cap Exec(P)$. A questo punto è semplice osservare che $Exec(P, A, C)$ è misurabile nel campo $\sum(P)$, poiché: $Exec(P, A, C) = \bigcup_\sigma \sigma\uparrow$, in cui l'unione è effettuata su tutti i frammenti di esecuzione σ che partono da P, terminano con uno stato in C e tali che $tr(\sigma)$ è una sequenza dell'insieme A.

Nel seguito $\tau^*\hat{a}$ rappresenta l'insieme delle sequenze $\tau^* a$ se $a \in \mathbf{GAct} - \{\tau\}$ e l'insieme delle sequenze τ^* se $a = \tau$.

Sia $Prob(P, \tau^*\hat{a}, C)$ la probabilità di passare dal termine P ad un termine dell'insieme C attraverso sequenze nella forma $\tau^*\hat{a}$, allora l'equazione:

$$Prob(P,\tau^{*}\hat{a},C)=\begin{cases}1 & (i)\\ \sum_{Q\in\mathcal{G}} Prob(P,\tau,Q)\cdot Prob(Q,\tau^{*},C) & (ii)\\ \sum_{Q\in\mathcal{G}} Prob(P,\tau,Q)\cdot Prob(Q,\tau^{*}a,C)+Prob(P,a,C) & (iii)\end{cases}$$

(i) se $a = \tau \wedge P \in C$

(ii) se $a = \tau \wedge P \notin C$

(iii) se $a \neq \tau$

ha una soluzione minima che coincide con $Prob(Exec(P, \tau^{*}\hat{a}, C))$.

La definizione di bisimulazione debole probabilistica per *GRTS* deriva dalla definizione di una bisimulazione debole probabilistica per sistemi pienamente probabilistici data in [45], in cui la relazione di transizioni debole $\stackrel{\hat{a}}{\Rightarrow}$ data da Milner [19] è sostituita con il valore probabilistico $Prob(P, \tau^{*}\hat{a}, C)$. Sottolineiamo che gli autori di [45] definiscono, per sistemi di transizioni pienamente probabilistici, sia una bisimulazione debole sia una bisimulazione con ritardo e mostrano che, nel contesto probabilistico, le due relazioni coincidono (per questo motivo nella definizione della relazione di equivalenza seguente possiamo usare $\tau^{*}\hat{a}$ invece di $\tau^{*}\hat{a}\,\tau^{*}$).

Definizione 3.3. Una relazione $\mathcal{R} \subseteq \mathcal{G} \times \mathcal{G}$ è una *bisimulazione debole probabilistica* se e solo se ogni volta che $(P, Q) \in \mathcal{R}$, allora per ogni $C \in \mathcal{G}/\mathcal{R}$:

- $Prob(P,\tau^{*}\hat{a},C) = Prob(Q,\tau^{*}\hat{a},C) \quad \forall\, a \in \mathbf{GAct}$
- $Prob(P, a^{*},C) = Prob(Q, a^{*},C) \quad \forall\, a^{*} \in \mathbf{RAct}$

Due termini P e Q sono equivalenti per bisimulazione debole probabilistica, indicato con $P \approx_{PB} Q$, se esiste una bisimulazione debole probabilistica $\mathcal{R}$ che contiene la coppia (P, Q).

Ricordiamo inoltre che in [45] è descritto un algoritmo che computa le classi di equivalenza di una bisimulazione debole probabilistica in tempo $O(n^3)$ e spazio $O(n^2)$, in cui n è il numero degli stati. Infine è facile notare che la definizione di bisimulazione debole probabilistica estende la nozione classica di bisimulazione debole, infatti se due termini sono equivalenti per bisimulazione debole probabilistica le loro controparti non-deterministiche sono equivalenti per bismulazione debole. In particolare, come vedremo nel prossimo paragrafo, le proprietà di sicurezza definite nel modello probabilistico catturano tutti i flussi di informazione che si possono rilevare analizzando il comportamento non-deterministico del sistema con l'approccio classico.

3.2.1 Un esempio

Si considerino i due *GRTS* in fig. 3.4.

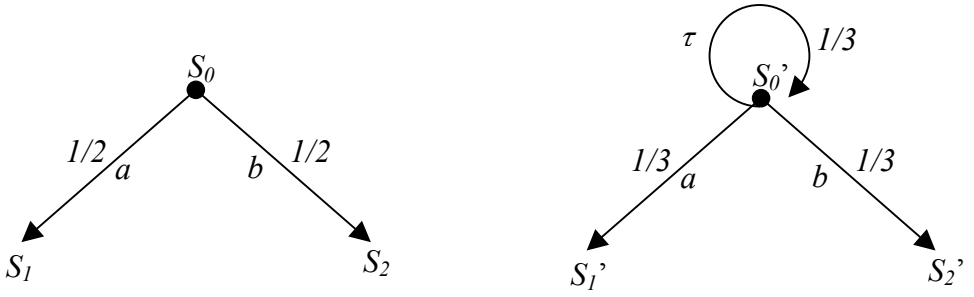


Fig 3.4. Sistemi equivalenti per bisimulazione debole probabilistica.

Se si utilizza la bisimulazione debole probabilistica $\approx_{PB}$ i due sistemi risultano equivalenti. Per dimostrare questo risultato si consideri la relazione di equivalenza $\mathcal{R}$ sull'insieme $S = \{S_0, S_1, S_2, S_0', S_1', S_2'\}$ tale che $S/\mathcal{R} = \{C_1, C_2\}$ in cui $C_1 = \{S_0, S_0'\}$ è la classe di equivalenza degli stati iniziali e $C_2 = \{S_1, S_2, S_1', S_2'\}$ è la classe di equivalenza degli stati rimanenti.

Per ogni $S \in C_1$ si ha $Prob(S, \tau^*, C_1) = 1$ e $Prob(S, \tau^* \pi, C_2) = ½$ con $\pi \in \{a, b\}$. Per ogni $S \in C_2$ si ha invece $Prob(S, \tau^*, C_2) = 1$ e $Prob(S, \tau^* \pi, C_1) = 0$ con $\pi \in \{a, b\}$. Di conseguenza $\mathcal{R}$ è una bisimulazione debole

probabilistica. E' opportuno sottolineare che dallo stato $S_0' \in C_1$ si può raggiungere lo stato $S_1' \in C_2$ ($S_2' \in C_2$) attraverso un'azione a (b) dopo avere eseguito l'azione τ un numero arbitrario di volte. Per calcolare la probabilità $Prob(S_0', \tau^* a, C_2)$ ($Prob(S_0', \tau^* b, C_2)$) si deve ridistribuire la probabilità 1/3 associata alla transizione interna τ eseguibile da S_0' tra le altre transizioni eseguibili da S_0'. Applicando la formula definita in precedenza si ottiene:
$Prob(S_0', \tau^* a, C_2) = 1/3 \cdot Prob(S_0', \tau^* a, C_2) + 1/3$ da cui si ottiene $Prob(S_0', \tau^* a, C_2) = ½$ (analogamente avviene per l'azione b).

3.3 Proprietà di sistemi probabilistici

In questa sezione introduciamo la proprietà di *Non deducibility on Composition* estendendo la teoria della non-interference descritta nella sezione **2.2** al modello probabilistico specificato in questo capitolo. In particolare baseremo tale proprietà di sicurezza sulla bisimulazione debole probabilistica definita nel paragrafo precedente. Prima di fare ciò individuiamo però un sottoinsieme dei termini di *PPA* sul quale formalizzeremo la nostra analisi.

Definiamo un operatore ausiliario $P//_a$ che si comporta come l'operatore di mascheramento $P/_a^{1/2}$ e che trasforma ogni azione a^* e a in azioni generative τ_{a^*} e τ_a (invece che in semplici azioni τ), mantenendo così una traccia delle azione osservabili originarie da cui derivano. Sia $\mathbf{GAct}_\tau = \{ \tau_a \mid a \in \mathbf{AType} - \{\tau\} \} \cup \{ \tau_{a^*} \mid a \in \mathbf{AType} - \{\tau\} \}$ e ridefiniamo **GAct** come $\mathbf{GAct} = \mathbf{AType} \cup \mathbf{GAct}_\tau$. Utilizziamo quindi l'abbreviazione $P//$ per rappresentare l'espressione $P/_{a\ b\ \ldots}$ che maschera, in ordine alfabetico, tutte le azioni osservabili, di qualsiasi tipo, eseguibili da ciascun termine $P' \in Der(P)$. Sia quindi $\mathbf{GAct}_\tau^\infty$ l'insieme delle sequenze infinite con elementi in $\mathbf{GAct}_\tau$. Definiamo l'insieme $Pref_w^a = \{ w' \in \mathbf{GAct}_\tau^* \mid \exists w'' \in \mathbf{GAct}_\tau^\infty , b \in \mathbf{Act}, b \neq a : w' \cdot \tau_b \cdot w'' = w \}$. Dato $L \subseteq \mathbf{GAct}_\tau^*$ e $a \in \mathbf{AType} - \{\tau\}$ assumiamo $L \cdot \tau_a = \{ w \cdot \tau_a \mid w \in L \}$ e

$Prob(P', L) = Prob(Exec(P', L, Der(P')))$.

Definizione 3.4. $P \in \mathcal{G}$ è *ben definito* $\Leftrightarrow$ $\exists\, \varepsilon \in]0, 1[$:

$\forall\, P' \in Der(P).\ \forall\, a \in \mathbf{AType}\text{-}\{\tau\}.\ \forall\, w \in \mathbf{GAct}_{\tau}^{\omega}$: $P'//^{Pref_w^a \cdot \tau_a} \Rightarrow$

$Prob(P', Pref_w^a \cdot \tau_a) > \varepsilon$.

Quindi definiamo $\mathcal{G}^w = \{\, P \in \mathcal{G} \mid P$ è *ben definito* $\}$.

E' opportuno sottolineare che la famiglia dei processi ben definiti non cambia se si sostituisce il parametro ½ nell'operatore $P//$ con un altro qualsiasi valore nell'intervallo]0, 1[. Inoltre se P è ben definito allora ogni $P' \in Der(P)$ e ogni Q tale che $P \approx_{PB} Q$ è ben definito. E' quindi facile osservare che tutti i processi che originano sistemi di transizioni finiti e tutti i processi con un insieme finito di classi di equivalenza per la bisimulazione debole probabilistica sono ben definiti.

Intuitivamente un processo è ben definito se ogni azione che può essere osservata, eseguendo una sequenza data di azioni, viene eseguita con una probabilità maggiore del valore fissato ε. Prendendo in considerazione l'insieme $\mathcal{G}^w$ dei processi ben definiti presenteremo nel prossimo paragrafo la nozione probabilistica della proprietà *BNDC* mostrata nella sezione **2.2**. Attraverso alcuni esempi analizzeremo tale proprietà enfatizzando la sua capacità di rilevare flussi di informazione indiretti che non sono catturati da proprietà classiche basate sul contesto non-deterministico. Mostreremo poi che il modello probabilistico è un'estensione conservativa dell'approccio non-deterministico alla teoria dell'information flow descritta nel capitolo **2**. In particolare se un sistema soddisfa una particolare proprietà di sicurezza nel contesto probabilistico allora il sistema soddisfa anche la corrispondente proprietà di sicurezza nel modello non-deterministico. Infine mostreremo che analizzando il comportamento probabilistico di un sistema si può ricavare una stima quantitativa del flusso di informazione, ovvero diventa possibile stimare la probabilità di osservare ogni comportamento potenzialmente insicuro

di un sistema.

3.3.1 Probabilistic Non-deducibility on Composition

Sia Seq^k_D l'insieme delle sequenze di elementi in D di lunghezza k.

Definizione 3.5. *PBNDC: Probabilistic BNDC*

$P \in \mathcal{G}^w \quad \in PBNDC \Leftrightarrow P \setminus \mathbf{AType_H} \approx_{PB} ((P \|^p_{\{h_1,\ldots,h_k\}} \Pi) /^{\mathbf{p}}_{h_1 \ldots h_k}) \setminus \mathbf{AType_H}$

$\forall$ sequenza $h_1 \ldots h_k \in \mathbf{AType_H}^*$, $\forall \Pi \in \mathcal{G}^w{}_H$, $\forall p \in]0, 1[$, $\forall \mathbf{p} \in Seq^k_{]0,1[}$.

Questa definizione stabilisce che l'osservazione probabilistica di basso livello del processo P non varia eseguendo P in parallelo con qualsiasi processo di alto livello Π. La proprietà *PBNDC* deve essere soddisfatta:

- per ogni processo di alto livello Π. Da questi dipende infatti la distribuzione di probabilità delle azioni generative di alto livello che utilizzano per sincronizzarsi con le corrispondenti azioni reattive di P.
- per ogni possibile scelta dei parametri p_i che formano la sequenza **p**. Questi sono scelti dal processo di alto livello per risolvere il non-determinismo dovuto alle azioni reattive di alto livello eseguibili da $P \|^p_{\{h_1,\ldots,h_k\}} \Pi$ (derivanti dalla sincronizzazione di azioni reattive dello stesso tipo di P e Π) e mascherate in azioni τ.
- per ogni possibile valore del parametro p che regola l'esecuzione parallela dei due processi P e Π.

La proprietà *PBNDC* è conforme alla nozione di non-interference probabilistica data da Gray in [55]. In particolare Gray definisce le nozioni di *comportamento di alto livello dell'ambiente* $\mathcal{H}$ e di *comportamento di basso livello dell'ambiente* $\mathcal{L}$ che forniscono la probabilità che l'ambiente produca azioni di input di alto (basso) livello, dato uno storico delle azioni di input di alto (basso) livello che hanno

avuto luogo fino a quel momento. Quindi controlla se, fissato $\mathcal{L}$ e modificando $\mathcal{H}$, la distribuzione di probabilità degli eventi di basso livello del sistema viene alterata o meno dal comportamento dell'ambiente descritto da $\mathcal{L}$ e $\mathcal{H}$. Nel contesto del nostro modello algebrico probabilistico, le possibili interazioni del sistema con agenti esterni di alto livello svolgono il ruolo dell'ambiente di alto livello.

Mostriamo di seguito alcuni esempi:

Esempio 3.1. Si consideri il processo $P = l.\underline{0} +^p h.h.l.\underline{0}$. Nell'esempio **2.4** abbiamo mostrato che il sistema $P_{nd} = l.\underline{0} + h.h.l.\underline{0}$, corrispondente al processo P in un contesto non-deterministico, non verificava la proprietà *BNDC*. Allo stesso modo $P \notin PBNDC$. Formalmente si consideri, ad esempio, il processo di alto livello $\Pi = h^*.\underline{0}$, otteniamo che:
$((P\|_{\{h\}}\Pi)\ /_h) \setminus \mathbf{AType_H} = l.\underline{0} +^p \tau.\underline{0} \not\approx_{PB} l.\underline{0} = P \setminus \mathbf{AType_H}$. Si noti che nell'operatore di mascheramento $/_h$ non utilizziamo alcun parametro probabilistico (dal momento che vengono mascherate solamente azioni generative è infatti inutile utilizzarlo).

Esempio 3.2. Si consideri il processo:

$$P = (\tau.(l.\underline{0} +^p h_1.\underline{0}) +^{1-p} \tau.(l.\underline{0} +^p \tau.\underline{0})) +^p h_2.l.\underline{0}$$

il cui *GRTS* è rappresentato in fig. 3.5(a) e in cui un utente di basso livello che osserva l'azione l può dedurre che l'azione di alto livello h_1 non è stata eseguita. In particolare un utente di alto livello potrebbe generare un covert channel nel seguente modo:

(*i*) previene l'esecuzione di h_2 nella componente $h_2.l.\underline{0}$

(*ii*) aspetta la risoluzione delle scelte probabilistiche tra le azioni τ

(*iii*) se viene scelta la componente $l.\underline{0} +^p h_1.\underline{0}$ con probabilità 1-p può alterare l'osservazione dal basso livello comunicando o meno l'azione h_1.

Tale comportamento insicuro è catturato dalla proprietà *PBNDC*. Si consideri infatti il processo $\Pi = h_1^*.\underline{0}$ e l'insieme di sincronizzazione

$S=\{h_1, h_2\}$. Il *GRTS* derivato dal termine $((P \|_S \Pi) / S) \setminus \mathbf{AType_H}$ è rappresentato in fig. 3.5(b). Il comportamento di questo sistema composto consiste in una scelta probabilistica tra l'azione l, eseguita con probabilità p, e l'azione τ, eseguita con probabilità $1\text{-}p$. Il *GRTS* derivato dal termine $P \setminus \mathbf{AType_H}$, rappresentato in fig. 3.5(c), è chiaramente diverso. Per cui $P \setminus \mathbf{AType_H} \not\approx_{PB} ((P \|_S \Pi) / S) \setminus \mathbf{AType_H}$ e, di conseguenza, $P \notin PBNDC$.

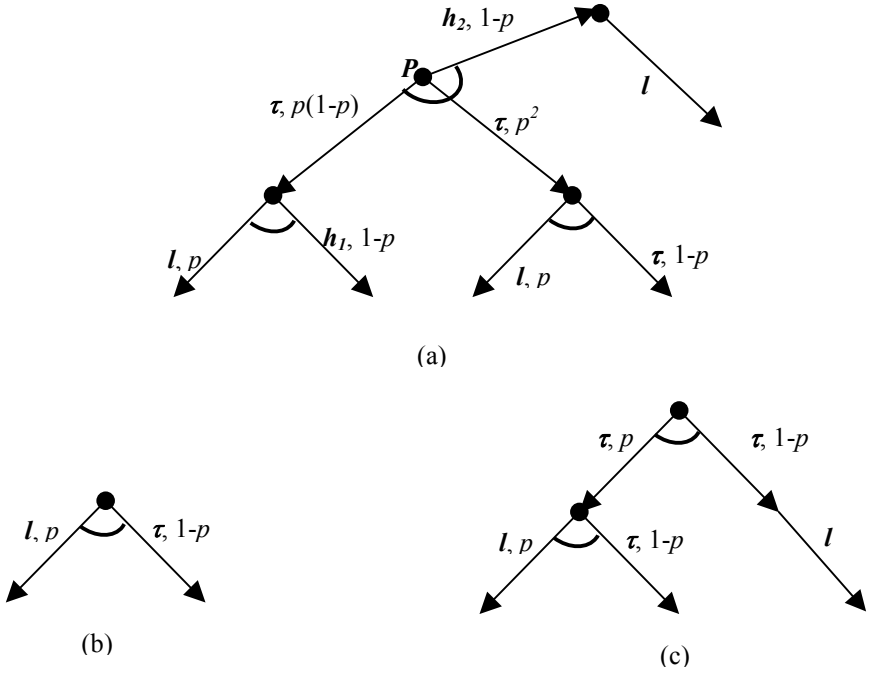


Fig. 3.5. Esempio di un sistema non *PBNDC*.

Esempio 3.3. Come esempio di un sistema sicuro si consideri il processo $P = l.P +^q h.l.P$ che può eseguire un'azione di basso livello preceduta o meno da un'azione di alto livello. Dal punto di vista di basso livello viene osservata l'azione l con probabilità 1 senza poter ottenere informazioni aggiuntive sul comportamento del livello alto. Informalmente si può osservare $((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H} \approx_{PB} P \setminus \mathbf{AType_H}$ per ogni processo di alto livello Π. Ovvero ogni possibile interazione del sistema P con processi di alto livello, non altera la distribuzione di probabilità dell'evento l e quindi del comportamento osservabile del

livello basso del sistema.

Concludiamo questo paragrafo sottolineando, come già abbiamo fatto per la proprietà *BNDC* nella sezione **2.2**, che la definizione di *PBNDC* non è semplice da verificare a causa della quantificazione universale sui processi di alto livello.

3.3.2 Rilevazione di flussi di informazione probabilistici

In questa sezione analizziamo le tipologie dei comportamenti insicuri di un sistema che possono essere rilevati nel contesto probabilistico. In particolare mostriamo attraverso alcuni esempi che la proprietà *PBNDC* è in grado di rilevare perdite di informazione indesiderate legate al comportamento probabilistico del sistema.

Esempio 3.4. Si consideri il sistema $P = (l.\underline{0} +^{p} l.l'.\underline{0}) +^{q} l.h.l'.\underline{0}$, il cui *GRTS* è rappresentato in fig. 3.6.

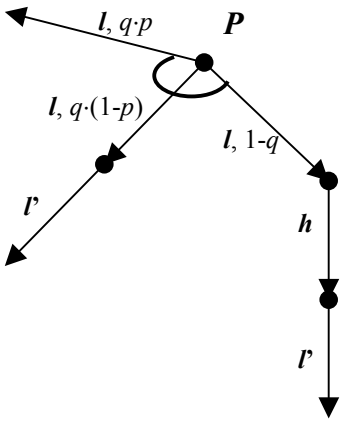


Fig. 3.6. Sistema con flusso di informazione probabilistico

L'esecuzione della sequenza *l.l'* (in confronto all'esecuzione di una singola azione *l*) può rivelare informazioni sull'esecuzione dell'azione *h* attraverso deduzioni statistiche che possono essere ricavate dalle frequenze relative dei due eventi facendo partire ripetutamente il sistema. Un agente di alto livello può cioè influenzare il comportamento del

sistema osservabile da un utente di basso livello che analizza la frequenza degli eventi di basso livello ad ogni esecuzione del sistema. Formalmente in $P \setminus \mathbf{AType_H} = (l.\underline{0} +^{p} l.l'.\underline{0}) +^{q} l.\underline{0}$ un utente di basso livello può osservare l'azione l con probabilità $q \cdot p + (1-q)$ o la sequenza di azioni $l.l'$ con probabilità $q \cdot (1-p)$. Al contrario, se $\Pi = h^*.\underline{0}$, dal termine $((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H} = (l.\underline{0} +^{p} l.l'.\underline{0}) +^{q} l.\tau.l'.\underline{0}$ un utente di basso livello può osservare una singola azione l con probabilità $q \cdot p$ e la sequenza $l.l'$ con probabilità $q \cdot (1-p) + (1-q)$. Di conseguenza $P \notin PBNDC$ poiché $P \setminus \mathbf{AType_H} \not\approx_{PB} ((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H}$.

Esempio 3.5. Si consideri il seguente processo non-deterministico:

$$P_{nd} = \tau.Q_{nd} + \tau.l_2.\underline{0}$$
$$Q_{nd} = \tau.l_1.\underline{0} + h^*.l_2.\underline{0}$$

sicuro in accordo alla proprietà *BNDC*. Intuitivamente le azioni di basso livello di P_{nd} osservabili sono date da una scelta non-deterministica tra le azioni l_1 ed l_2, e la composizione di P_{nd} con qualsiasi processo di alto livello non altera il comportamento osservabile del sistema isolato. Tuttavia il sistema è da considerarsi insicuro poiché, una volta raggiunto lo stato Q_{nd}, un agente di alto livello, eseguendo un'azione h che si sincronizza con l'azione h^*, determina l'osservazione dell'azione l_1 o dell'azione l_2. Di conseguenza un utente di basso livello che osserva l'azione l_1 può dedurre che il sistema non ha eseguito l'azione h. Si consideri ora l'estensione probabilistica del sistema P_{nd}:

$$P = \tau.Q +^{1/2} \tau.l_2.\underline{0}$$
$$Q = \tau.l_1.\underline{0} + h^*.l_2.\underline{0}$$

che genera il *GRTS* rappresentato in fig. 3.7. Nel modello probabilistico la vista del sistema di basso livello è determinata da una scelta probabilistica tra le azioni l_1 e l_2. Se viene raggiunto lo stato Q un agente di alto livello può interferire con il sistema risolvendo la scelta non-deterministica tra l'azione τ e l'azione reattiva h^*. Le probabilità di osservare le azioni l_1 e l_2 vengono quindi modificate. Viene alterata,

inoltre, l'osservazione di un utente di basso livello che analizza le frequenze relative delle azioni osservabili ad ogni esecuzione del sistema. Tale flusso di informazione probabilistico viene catturato dalla proprietà *PBNDC*, ovvero $P \notin PBNDC$. Formalmente, dato $\Pi = h.\underline{0}$, abbiamo che: $((P \parallel_{\{h\}} \Pi) \; /_h^q) \setminus \mathbf{AType_H} = \tau.(\tau.l_1.\underline{0} +^{1-q} \tau.l_2.\underline{0}) +^{1/2} \tau.l_2.\underline{0} \not\approx_{PB} \tau.\tau.l_1.\underline{0} +^{1/2} \tau.l_2.\underline{0} = P \setminus \mathbf{AType_H}$ per qualsiasi scelta del parametro $q \in]0, 1[$. In particolare in $P \setminus \mathbf{AType_H}$ un utente di basso livello osserva l'azione l_1 con probabilità ½ mentre in $((P \parallel_{\{h\}} \Pi) \; /_h^q) \setminus \mathbf{AType_H}$ può osservare la stessa azione l_1 con probabilità ½(1-q).

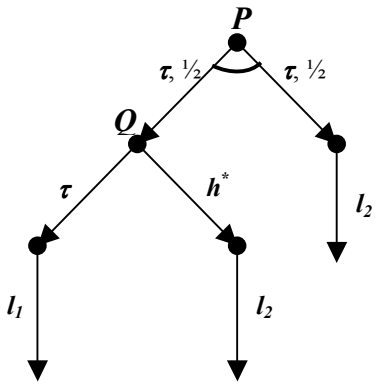


Fig. 3.7. Sistema *BNDC* ma non *PBNDC*

L'ultimo esempio proposto mostra come un sistema sicuro nel contesto non-deterministico possa causare una perdita di informazione se analizzato in un modello probabilistico. In particolare abbiamo mostrato che la controparte probabilistica di un sistema non-deterministico sicuro in accordo alla proprietà *BNDC*, può non soddisfare la *PBNDC*. Mostriamo infine un ultimo esempio interessante che sottolinea come l'analisi probabilistica del comportamento dei sistemi è in grado rivelare flussi di informazione che non vengono individuati in un contesto non-deterministico.

Esempio 3.6. Si consideri il seguente pseudo-codice ispirato da un

esempio proposto da Sabelfeld e Sands in [53, 54]:

$$l := h \quad +^{p} \quad l := rand(1)$$

in cui h è una variabile di alto livello impostata preventivamente a 0 o a 1 da un utente di alto livello (in accordo ad una data distribuzione di probabilità) e $rand(n)$ è una funzione che sceglie casualmente un numero naturale nell'insieme $\{0, \ldots, n\}$. L'output osservabile dal basso livello (costituito dal valore della variabile l) è quindi letto direttamente dalla variabile di alto livello h con probabilità p oppure assegnato casualmente con probabilità 1-p. Il comportamento di un sistema di questo tipo è rappresentato graficamente nella fig. 3.8.

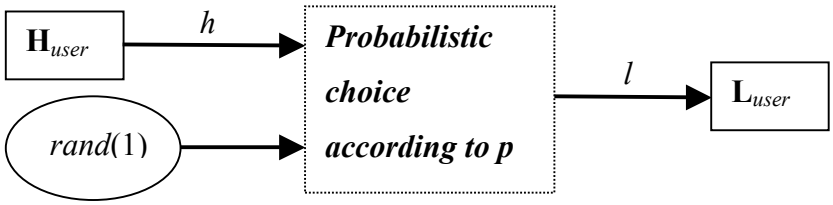


Fig. 3.8. Sistema che nasconde un covert channel probabilistico.

E' facile osservare che se si considerano solamente le azioni che il sistema è in grado di eseguire non si riescono ad individuare flussi di informazione. Tuttavia il valore finale di l potrebbe rivelare h se l'utente di basso livello osserva le frequenze relative dei risultati di esecuzioni ripetute.

Si consideri ad esempio il seguente processo:

$$P = (l_0.P +^{1/2} l_1.P) +^{p} h.(l_0.P +^{q} l_1.P)$$

in cui un utente di basso livello osserva le azioni l_0 e l_1 con una distribuzione di probabilità che dipende dal comportamento di alto livello dell'ambiente. In particolare, se non viene effettuata alcuna sincronizzazione con l'azione di alto livello h, il sistema sceglie tra le due azioni l_0 e l_1 con la stessa probabilità ½ (situazione che modella l'assegnamento $l := rand(1)$). Al contrario, se il sistema interagisce con

un agente di alto livello esterno attraverso l'azione h, la distribuzione di probabilità dei due eventi di basso livello è guidata dal parametro q (situazione che modella l'assegnamento $l := h$). Intuitivamente q è il parametro della distribuzione di probabilità dei due possibili valori (0 e 1) che h può assumere, mentre p è il parametro che guida la scelta probabilistica tra le due alternative che possono produrre l'output al basso livello. La versione non-deterministica del processo P è sicura poiché l'esecuzione di qualsiasi azione di alto livello non altera il comportamento osservabile di basso livello del sistema. Dato il termine $P_{nd} = (l_0.P_{nd} + l_1.P_{nd}) + h.(l_0.P_{nd} + l_1.P_{nd})$ si ha quindi che $P_{nd} \in BNDC$. Tuttavia nel contesto probabilistico non possiamo affermare che P è sicuro. La probabilità di osservare un'azione l_0 rispetto ad un'azione l_1 viene infatti modificata in base al comportamento dell'agente di alto livello che interagisce con il sistema. Formalmente dato $\Pi = h^*.\underline{0}$, abbiamo che: $((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H} = (l_0.P +^{1/2} l_1.P) +^p \tau.(l_0.P +^q l_1.P) \not\approx_{PB} l_0.P +^{1/2} l_1.P = P \setminus \mathbf{AType_H}$, fatta eccezione per il caso $q = ½$ (vedi fig. 3.9). Dunque $P \in PBNDC$ solo quando $q = ½$ poiché in questo caso l'utente di alto livello simula il comportamento della funzione *rand*(1) e l'utente di basso livello non può dedurre nulla, riguardo il comportamento di alto livello, osservando la distribuzione di probabilità delle due azioni l_1 e l_2.

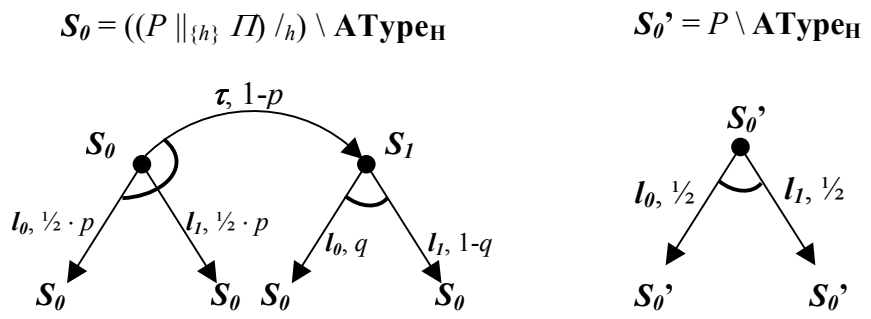


Fig. 3.9. $S_0 \approx_B S_1 \approx_B S_0'$ $S_0 \not\approx_{PB} S_1 \not\approx_{PB} S_0'$ **eccetto per** $q = ½$.

Concludiamo questo paragrafo sottolineando che il flusso di informazione rilevato nel processo P dell'ultimo esempio è puramente

probabilistico (ovvero viene rilevato solamente osservando la distribuzione di probabilità delle azioni di basso livello osservabili) e non viene rilevato neppure dalle più forti proprietà (*SBSNNI* e *SBNDC*) proposte in [20].

3.3.3 Estensione conservativa

In questa sezione mostriamo che la proprietà di sicurezza *PBNDC* definita per l'analisi di sistemi probabilistici è un'estensione naturale e conservativa delle proprietà di sicurezza *BNDC* descritta nella sezione **2.2** per l'analisi di sistemi non-deterministici. Intuitivamente l'introduzione di parametri probabilistici nella descrizione di sistemi costituisce un'informazione aggiuntiva che estende la conoscenza disponibile nel caso non-deterministico. Nei paragrafi precedenti abbiamo mostrato, attraverso alcuni esempi, che se anche un sistema non-deterministico P_{nd} soddisfa la proprietà di sicurezza *BNDC*, la sua versione probabilistica P, ottenuta modellando il comportamento probabilistico del sistema, potrebbe non soddisfare la proprietà *PBNDC*. Ora vogliamo mostrare che se un termine P, definito attraverso il calcolo probabilistico *PPA*, soddisfa la proprietà di sicurezza *PBNDC*, allora il sistema P_{nd}, definito nel linguaggio formale *NDPA*, soddisfa la proprietà *BNDC*. Per dimostrare tale relazione analizziamo innanzitutto come si modifica tale proprietà di sicurezza nel passaggio dal modello probabilistico al modello non-deterministico.

Il lemma che segue stabilisce che se due processi P e Q sono equivalenti per bisimulazione debole probabilistica allora i due termini P_{nd} e Q_{nd}, ottenuti rimovendo i parametri probabilistici dagli operatori algebrici di P e Q, sono equivalenti per bisimulazione debole.

Lemma 3.1. Dati $P, Q \in \mathcal{G} : P \approx_{PB} Q \;\Rightarrow\; P_{nd} \approx_B Q_{nd}, \quad P_{nd}, Q_{nd} \in \mathcal{G}_{nd}$

Dimostrazione: Per dimostrare che $P_{nd} \approx_B Q_{nd}$ mostriamo l'esistenza di una bisimulazione debole $\mathcal{R}$ che include la coppia (P_{nd}, Q_{nd}). Analizziamo i due casi possibili:

- $P \approx_{PB} Q \Rightarrow Prob(P, \tau^* \hat{a}, C) = Prob(Q, \tau^* \hat{a}, C) \quad \forall\ a \in \mathbf{GAct}$. Si assuma che $P_{nd} \xrightarrow{a} P'_{nd}$, in cui P'_{nd} è la versione non-deterministica di un termine $P' \in C$. Per ipotesi $\exists\ Q'_{nd}$ derivato da $Q' \in C$: $Q_{nd} \stackrel{\hat{a}}{\Rightarrow} Q'_{nd}$ e $(P'_{nd}, Q'_{nd}) \in \mathcal{R}$, poiché P' e Q' appartengono alla stessa classe di equivalenza C. Lo stesso avviene se scambiamo i ruoli di P e Q.
- $P \approx_{PB} Q \Rightarrow Prob(P, a^*, C) = Prob(Q, a^*, C)\ \forall\ a^* \in \mathbf{RAct}$. Si assuma che $P_{nd} \xrightarrow{a^*} P'_{nd}$, in cui P'_{nd} è la versione non-deterministica di un termine $P' \in C$. Per ipotesi $\exists\ Q'_{nd}$ derivato da $Q' \in C$: $Q_{nd} \xrightarrow{a^*} Q'_{nd}$ e $(P'_{nd}, Q'_{nd}) \in \mathcal{R}$, poiché P' e Q' appartengono alla stessa classe di equivalenza C. Lo stesso avviene se scambiamo i ruoli di P e Q.

Teorema 3.1 (Estensione Conservativa).

$P \in \mathcal{G}^w \in PBNDC \Rightarrow P_{nd} \in \mathcal{G}_{nd} \in BNDC$.

Dimostrazione: Poiché $P \in PBNDC$ abbiamo che:

$P \setminus \mathbf{AType_H} \approx_{PB} ((P \|^p_{\{h_1,\ldots,h_k\}} \Pi) /^{\mathbf{p}}_{h_1 \ldots h_k}) \setminus \mathbf{AType_H}$

$\forall$ sequenza $h_1 \ldots h_k \in \mathbf{AType_H}^*$, $\forall \Pi \in \mathcal{G}^w_H$, $\forall p \in]0, 1[$, $\forall \mathbf{p} \in Seq^k_{]0,1[}$. Per il lemma 3.1 i termini $P_{nd} \setminus \mathbf{AType_H}$ e $((P_{nd} \|_S \Pi) /_S) \setminus \mathbf{AType_H}$ sono equivalenti per bisimulazione debole $\forall \Pi \in \mathcal{G}_{Hnd}$ e $\forall S \subseteq \mathbf{AType_H}$. Di conseguenza $P_{nd} \in BNDC$.

Esempio 3.7. Il sistema non-deterministico $P_{nd} = \tau.l.\underline{0} + h_1.l.\underline{0} + \tau.\underline{0} + h_2.\underline{0}$ è sicuro in accordo alla proprietà *BNDC*: l è l'unico evento di basso livello osservabile e qualsiasi interazione di P_{nd} con processi di alto livello non può alterare tale comportamento. Tuttavia l'esecuzione dell'azione di basso livello l rivela che non è stata eseguita l'azione di alto livello h_2. Un flusso di informazione di questo tipo è catturato solamente dalla proprietà *SBNDC* [20].

Una possibile estensione probabilistica del termine P_{nd} potrebbe essere

data dal sistema $P = (\tau.l.\underline{0} +^{p} h_1.l.\underline{0}) +^{1-p} (\tau.\underline{0} +^{p} h_2.\underline{0})$. Nel modello probabilistico è sufficiente usare la proprietà *PBNDC*. Infatti se si considera il processo $\Pi = h_2^*.\underline{0}$ e l'insieme di sincronizzazione $S = \{h_1, h_2\}$ è facile notare che $P \setminus \mathbf{AType_H} \not\approx_{PB} ((P \|_S \Pi) / S) \setminus \mathbf{AType_H}$.

3.3.4 Misura del flusso di informazione

Modellando il comportamento probabilistico di sistemi è possibile analizzare nuovi aspetti dei potenziali flussi di informazione che si possono generare dal livello alto al livello basso. Analizzando un sistema non-deterministico è possibile solamente stabilire se il sistema è sicuro o meno, nel contesto probabilistico è invece possibile stimare la probabilità con cui un sistema potrebbe rivelare un flusso di informazione insicuro.

Diviene quindi necessario definire un'equivalenza quantitativa che permetta di stabilire se due sistemi probabilistici si comportano quasi nello stesso modo (a meno di piccole fluttuazioni). Più formalmente un'equivalenza di questo tipo deve essere in grado di misurare la distanza tra sistemi di transizioni probabilistici. Possiamo quindi arricchire la bisimulazione debole probabilistica in modo che tolleri delle leggere fluttuazioni, rendendo la proprietà di sicurezza meno restrittiva. In [56] e [57] vengono proposte diverse pseudometriche che permettono di quantificare le similarità tra i comportamenti di sistemi di transizioni probabilistici che non risultano equivalenti per bisimulazione. In particolare, in [57], gli autori considerano uno spazio metrico su catene di Markov parziali. Queste sono una generalizzazione dei *GRTS* pienamente specificati in cui la somma delle probabilità delle transizioni uscenti da uno stato può anche essere minore di 1.

Definiamo ora una bisimulazione debole probabilistica con precisione ε che permetta lievi fluttuazioni tra processi equivalenti.

Definizione 3.6. Una relazione di $\mathcal{R} \subseteq \mathcal{G} \times \mathcal{G}$ è una *bisimulazione debole probabilistica con precisione* ε, in cui $\varepsilon \in]0, 1[$, se e solo se ogni volta

che $(P, Q) \in \mathcal{R}$, allora per ogni $C \in \mathcal{G}/\mathcal{R}$:

- $| Prob(P, \tau^* \hat{a}, C) \quad - \quad Prob(Q, \tau^* \hat{a}, C) | \quad \leq \varepsilon \quad \forall\, a \in \mathbf{GAct}$
- $| Prob(P, a^*, C) \quad - \quad Prob(Q, a^*, C) \quad | \quad \leq \varepsilon \quad \forall\, a^* \in \mathbf{RAct}$

Due termini P e Q sono equivalenti per bisimulazione debole probabilistica con precisione ε, indicato con $P \approx_{PB\varepsilon} Q$, se esiste una bisimulazione debole probabilistica con precisione ε che contiene la coppia (P, Q).

Esempio 3.8. Si consideri il sistema:

$$P = R +^p \tau.Q$$
$$Q = l.\underline{0} +^q h.l.\underline{0}$$
$$R = h.l'.\underline{0}$$

rappresentato graficamente dal *GRTS* in fig. 3.10. Il processo Q è una componente sicura del sistema (si veda l'analisi fatta nell'esempio 3.3). Al contrario il processo R è chiaramente insicuro, si osservi che l'esecuzione dell'azione h permette ad un utente di basso livello di osservare l'evento l' che altrimenti non si verificherebbe.

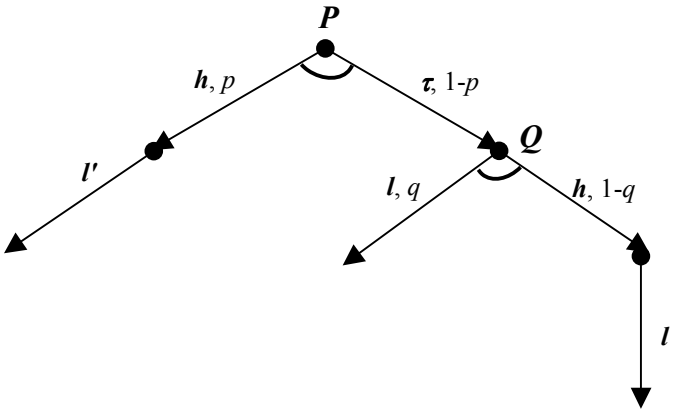


Fig. 3.10. Esempio di un flusso di informazione probabilistico

L'informazione legata al comportamento probabilistico del sistema non è necessaria per catturare questo tipo di flusso di informazione (facilmente rilevato anche nel caso non-deterministico dalla proprietà *BNDC*).

Tuttavia nel contesto probabilistico siamo in grado di stabilire che il sistema P è sicuro con probabilità 1-p e che può rivelare un flusso di informazione con probabilità p. Nel caso in cui p è un valore prossimo a 0, un utente di basso livello può inferire solo con bassissima probabilità il comportamento di alto livello del sistema, che può quindi essere considerato sufficientemente sicuro. Formalmente, dato il processo di alto livello $\Pi = h^*.\underline{0}$, diremo che il termine $P \setminus \mathbf{AType_H}$ è una p-perturbazione di $((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H}$, si verifica infatti che:
$P \setminus \mathbf{AType_H} = \tau.l.\underline{0} \approx_{PB} \tau.(l.\underline{0} +^q \tau.l.\underline{0}) \approx_{PBp} \tau.l'.\underline{0} +^p \tau.(l.\underline{0} +^q \tau.l.\underline{0}) = ((P\|_{\{h\}}\Pi) /_h) \setminus \mathbf{AType_H}$. Di conseguenza il sistema può essere considerato sufficientemente sicuro al tendere di p al valore 0.

Esempio 3.9. Riprendiamo il sistema probabilistico dell'esempio 3.4:

$$P = (l.\underline{0} +^p l.l'.\underline{0}) +^q l.h.l'.\underline{0}$$

Come abbiamo visto P non è sicuro in accordo alla proprietà *PBNDC*. Denotiamo con $C_1 = [\underline{0}]_{\approx_{PB}}$ la classe di equivalenza dello stato inattivo e con $C_2 = [l'.\underline{0}]_{\approx_{PB}}$ la classe di equivalenza del termine $l'.\underline{0}$. In un caso abbiamo che: $Prob(P \setminus \mathbf{AType_H}, \tau^*.l, C_1) = q \cdot p + 1\text{-}q$ e

$$Prob(P \setminus \mathbf{AType_H}, \tau^*.l, C_2) = q \cdot (1\text{-}p).$$

Dato $\Pi = h^*.\underline{0}$ e definito $P_c = ((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H}$ abbiamo invece:
$Prob(P_c, \tau^*.l, C_1) = q \cdot p$ e $Prob(P_c, \tau^*.l, C_2) = 1\text{-}q \cdot p$.
Di conseguenza avremo:
$| Prob(P \setminus \mathbf{AType_H}, \tau^*.l, C_1) - Prob(P_c, \tau^*.l, C_1) | = 1\text{-}q =$
$| Prob(P \setminus \mathbf{AType_H}, \tau^*.l, C_2) - Prob(P_c, \tau^*.l, C_2) |$, da cui possiamo derivare che $P \setminus \mathbf{AType_H} \approx_{PB\varepsilon} ((P \|_{\{h\}} \Pi) /_h) \setminus \mathbf{AType_H}$ se $q \geq 1\text{-}\varepsilon$. Intuitivamente il comportamento insicuro del sistema P è tanto meno frequente quanto più il valore del parametro q si avvicina a 1.

Il nostro approccio probabilistico può quindi essere usato per verificare il livello di sicurezza di sistemi in cui la probabilità gioca un ruolo particolarmente importante.

Concludiamo questa sezione sottolineando che trattare il comportamento probabilistico di sistemi permette inoltre, in assenza di non-determinismo, di valutarne le prestazioni in maniera semplice e naturale. Dall'analisi di sistemi chiusi (il cui *GRTS* derivato è puramente generativo) è infatti possibile derivare una catena di Markov rimovendo semplicemente le azioni dalle etichette delle transizioni.

Esempio 3.10. Si consideri il sistema P descritto nell'esempio 3.6:

$$P = (l_0.P +^{1/2} l_1.P) +^{p} h.(l_0.P +^{q} l_1.P)$$

in cui la distribuzione di probabilità degli eventi di basso livello l_0 e l_1 dipende dal comportamento dell'utente di alto livello. L'analisi delle caratteristiche di sicurezza di P ci ha permesso di stabilire che P non soddisfa la proprietà *PBNDC*. Se, d'altra parte, si è interessati a valutare le prestazioni del sistema, è possibile derivare una catena di Markov dal *GRTS* di P dalla quale, attraverso le classiche tecniche di valutazione delle prestazioni (si veda ad esempio [58]), si può calcolare la probabilità di eseguire l'evento di alto livello h rispetto all'esecuzione di un evento di basso livello (l_0 e l_1). Per cui sulla base dello stesso modello è possibile effettuare un'analisi delle proprietà di sicurezza e valutare le prestazioni di un sistema.

4 Modello formale per la crittografia

Una modellazione astratta delle operazioni di crittografia è spesso più che adeguata per la definizione, l'analisi e l'implementazione di sistemi crittografici. Spesso è conveniente, ad esempio, ignorare i dettagli della funzione di cifratura utilizzata, modellando con una descrizione di più alto livello gli obiettivi che si vogliono realizzare attraverso le operazioni di crittografia. Un'estesa branchia della letteratura tratta dunque le operazioni di crittografia in maniera puramente formale. Ad esempio l'espressione $\{M\}_K$ potrebbe rappresentare un messaggio cifrato in cui M è il testo in chiaro e K è la chiave di cifratura utilizzata. Da questo punto di vista $\{M\}_K$, M e K rappresentano espressioni formali più che stringhe di bit. Varie funzioni possono poi essere applicate a questo tipo di espressioni generandone altre. Una di queste funzioni potrebbe essere quella di decifratura, che, prese in input le espressioni formali $\{M\}_K$ e K, restituisce l'espressione M. In particolare, non dovrebbe esserci alcun modo di ricavare M o K dall'espressione $\{M\}_K$ presa singolarmente (questo è vero in caso di crittografia perfetta, noi in realtà abbandoniamo questa assunzione). Bastano dunque queste poche regole per modellare, in maniera astratta, operazioni di crittografia.

Tra gli autori che hanno dato origine a questa branchia della letteratura ricordiamo, primi tra tutti, Dolev e Yao [3], quindi DeMillo, Lynch e Meritt [4], Millen, Clark e Freedman [5], Kemmerer [6], Burrows, Abadi e Needham [7] e infine Meadows [8].

In quest'opera, per la definizione formale di primitive di crittografia, riprenderemo il modello presentato da Abadi e Rogaway in [2]. Estenderemo poi tale modello definendo un'equivalenza probabilistica tra espressioni formali in cui abbandoniamo l'assunzione di crittografia perfetta, modellando in questo modo anche possibili attacchi all'algoritmo di cifratura.

Nei protocolli reali è infatti impossibile utilizzare algoritmi di cifratura perfetti. Un agente disonesto può, in ogni momento, portare attacchi a qualsiasi algoritmo di cifratura, e, sebbene con probabilità estremamente basse, potrebbe riuscire a decrittare un messaggio senza conoscere la chiave utilizzata per cifrarlo. In letteratura non esistono approcci formali all'analisi delle proprietà di sicurezza di protocolli crittografici che non assumano crittografia perfetta. Noi abbiamo invece deciso di modellare uno scenario in cui siano contemplati anche possibili attacchi agli algoritmi di cifratura utilizzati. Abbandoneremo dunque l'assunzione di crittografia perfetta e definiremo formalmente delle operazioni di crittografia che ci permettano di modellare la possibilità di decrittare un messaggio $\{M\}_K$ senza conoscere la chiave K.

Il formalismo descritto da Abadi e Rogaway permette di modellare algoritmi di cifratura simmetrici. Per maggiore semplicità e chiarezza in fase di trattazione abbiamo deciso di seguire il loro approccio modellando nello specifico primitive di crittografia a chiave simmetrica. L'estensione del modello ad algoritmi di cifratura asimmetrica renderebbe molto più complesse la maggior parte delle definizioni introdotte.

Unità fondamentale del modello sono le *espressioni*, queste rappresentano i messaggi trasmessi nei protocolli di sicurezza. Le espressioni sono costruite partendo da bit e da un insieme di chiavi attraverso operazioni di accoppiamento e cifratura. Una relazione di equivalenza tra espressioni permette poi di stabilire se dei dati possono apparire uguali ad un nemico che non ha conoscenza delle chiavi di cifratura utilizzate. Infine, considerando crittografia imperfetta e modellando

possibili attacchi all'algoritmo di cifratura, attraverso un'equivalenza probabilistica confronteremo le espressioni analizzando le informazioni contenute in esse e la probabilità con cui un nemico può ricavarle. In particolare, analizzeremo la probabilità massima con cui un nemico può rompere un blocco crittato, attraverso attacchi all'algoritmo di cifratura, analizzando nella maniera ottima la conoscenza di cui dispone.

4.1 Espressioni

Indichiamo con **Bool** l'insieme dei bit {0, 1}. Sia poi **Keys** un insieme di simboli fissato e non vuoto disgiunto dall'insieme **Bool**. I simboli K', K'', K''', … e K_1, K_2, K_3, … definiscono elementi dell'insieme **Keys**. Informalmente l'insieme **Keys** rappresenta l'insieme delle chiavi di cifratura usate durante la costruzione di espressioni. Formalmente le chiavi sono dei simboli atomici e non vanno quindi immaginate come stringhe di bit.

Indichiamo quindi con **Exp** l'insieme delle espressioni ottenute dalla seguente grammatica:

$M, N ::=$	espressioni
K	chiave ($K \in$ **Keys**)
i	bit ($i \in$ **Bool**)
(M, N)	accoppiamento
$\{M\}_K$	cifratura

Informalmente (M, N) rappresenta l'accoppiamento delle espressioni M e N, che può essere realizzato attraverso un'operazione di concatenazione aggiungendo dei marcatori, mentre $\{M\}_K$ rappresenta la cifratura dell'espressione M con la chiave K, che può essere realizzata con l'utilizzo di un algoritmo di cifratura simmetrico.

Accoppiamento e cifratura possono essere annidati liberamente come, ad esempio, nell'espressione $(\{((0, 1) , K')\}_{K''} , K''')$.

4.2 Equivalenza

In questa sezione viene data una definizione formale di equivalenza tra espressioni. Questa è basata sulle definizioni di Syverson e Van Oorschot [9], Schneider [10] e Paulson [11]. Alcune delle definizioni aggiunte da Abadi e Rogaway in [2] riguardano particolari modalità su come analizzare e sintetizzare le espressioni.

Inizialmente viene definita una relazione di derivazione tra espressioni che chiameremo *entailment* ($\mapsto$). Questa relazione è basata sulle principali regole utilizzate nella modellazione formale di primitive di crittografia. Attraverso questa relazione esprimeremo, nel seguito di quest'opera, tutte le operazioni che effettueremo per la gestione e la manipolazione delle espressioni. Intuitivamente $M \mapsto N$ indica che l'espressione N può essere ricavata dall'espressione M. Una definizione formale di tale relazione viene data induttivamente rappresentando $\mapsto$ come la minima relazione che soddisfa le proprietà in tab. 4.1.

• $M \mapsto 0$	$\wedge$	$M \mapsto 1$		
• $M \mapsto M$				
• $M \mapsto N_1$	$\wedge$	$M \mapsto N_2$	$\Rightarrow$	$M \mapsto (N_1, N_2)$
• $M \mapsto (N_1, N_2)$			$\Rightarrow$	$M \mapsto N_1 \wedge M \mapsto N_2$
• $M \mapsto N$	$\wedge$	$M \mapsto K$	$\Rightarrow$	$M \mapsto \{N\}_K$
• $M \mapsto \{N\}_K$	$\wedge$	$M \mapsto K$	$\Rightarrow$	$M \mapsto N$

Tab. 4.1. Regole di *entailment*.

La relazione $M \mapsto N$ appena definita modella tutte le espressioni N che possono essere ottenute da un nemico a partire dall'espressione M senza avere alcuna conoscenza a priori delle chiavi utilizzate in M. Di seguito mostriamo alcuni esempi di *entailment*:

$$(\{\{K1\}_{K2}\}_{K3}\ ,\ K3) \quad \mapsto\ K3$$

e

$$(\{\{K1\}_{K2}\}_{K3}\ ,\ K3) \quad \mapsto\ \ \{K1\}_{K2}$$

e infine

$$(\{\{K1\}_{K2}\}_{K3}\ ,\ K3) \quad \not\mapsto\ \ K1$$

Un modello più complesso di quello appena mostrato, che tenga conto di una conoscenza a priori, è facilmente ottenibile: derivare un'espressione N da un'espressione M conoscendo a priori un'espressione M_p equivale a derivare N dall'espressione accoppiamento (M, M_p) con la relazione $\mapsto$ illustrata sopra.

Introduciamo ora il simbolo $\otimes$ ad identificare un testo cifrato che un nemico non può decifrare e definiamo l'insieme **Pat** dei *patterns* come un'estensione dell'insieme delle espressioni attraverso la seguente grammatica:

$P, Q ::=$	patterns
K	chiave ($K \in$ **Keys**)
i	bit ($i \in$ **Bool**)
(P, Q)	accoppiamento
$\{P\}_K$	cifratura
$\otimes$	testo non decifrabile

Intuitivamente un pattern rappresenta un'espressione che può contenere alcune parti non decifrabili.

Definiamo poi una funzione che presi in input un'espressione M e un insieme di chiavi $T \subseteq$ **Keys** riduce M in un pattern (intuitivamente il pattern ottenuto con tale funzione è il pattern che vede un nemico attraverso M se conosce a priori le chiavi in T).

$p(K, T)$	=	K	(con $K \in$ **Keys**)
$p(i, T)$	=	i	(con $i \in$ **Bool**)
$p((M, N), T)$	=	$(p(M, T), p(N, T))$	
$p(\{M\}_K, T)$	=	$\{p(M, T)\}_K$	se $K \in T$
		$\otimes$	altrimenti.

Infine definiamo una funzione che produca un pattern prendendo in input un'espressione senza un insieme ausiliario di chiavi T.

$$pattern(M) = p(M, \{K \in \textbf{Keys} \mid M \mapsto K\})$$

Intuitivamente questo è il pattern che un nemico, senza alcuna conoscenza a priori, può distinguere da un'espressione M utilizzando solo le chiavi che riesce ad ottenere da M attraverso la relazione di entailment. Per esempio:

$$pattern(\ (\{\{K1\}_{K2}\}_{K3}\ ,\ K3)\ ) = (\{\otimes\}_{K3}\ ,\ K3)$$

Definizione 4.1. *Equivalenza di espressioni*
Due espressioni sono *equivalenti* se producono lo stesso pattern:

$$M \equiv N \iff pattern(M) = pattern(N)$$

Cioè ad esempio:
$(\{\{K1\}_{K2}\}_{K3}\ ,\ K3) \equiv (\{\{0\}_{K1}\}_{K3}\ ,\ K3)$
poiché entrambe producono il pattern $(\{\otimes\}_{K3}\ ,\ K3)$.

La definizione di equivalenza appena data è tuttavia eccessivamente restrittiva. Date le due espressioni $(\{0\}_K\ ,\ K)$ e $(\{0\}_{K'},\ K')$ si nota che queste non sono equivalenti, infatti pur contenendo la stessa unità di informazione (in questo caso il bit 0) producono pattern diversi. Se consideriamo la possibilità di ridenominare le chiavi possiamo tuttavia affermare che le due espressioni in questione sono *equivalenti attraverso*

ridenominazione (≅):

Definizione 4.2. *Equivalenza attraverso ridenominazione*.

$M \cong N \Leftrightarrow$ esiste una funzione biunivoca σ sull'insieme **Keys** tale che $M \equiv N\sigma$ in cui $N\sigma$ è l'espressione che si ottiene da N ridenominando le chiavi attraverso la funzione σ.

4.2.1 Alcune considerazioni

Nel prossimo paragrafo introdurremo un'equivalenza probabilistica tra espressioni. Prima di fare ciò si ritiene tuttavia opportuno fare alcune considerazioni sull'equivalenza tra espressioni appena mostrata. L'idea fondamentale di Abadi e Rogaway in [2] costituiva nel modellare un'equivalenza che stabilisse se due espressioni potevano apparire identiche ad un avversario. L'idea che noi invece svilupperemo nella definizione dell'equivalenza probabilistica si basa su un confronto dell'informazione che un'espressione mette a disposizione. Se si considerano, ad esempio, le due espressioni ($\{0\}_K$, K) e (0, K) si ha che l'informazione trasportata dalle due espressioni è costituita dal bit 0 e dalla chiave K. Un nemico infatti è in grado di ricavare il bit 0 dalla prima espressione, nonostante sia cifrato, utilizzando la chiave K contenuta nell'espressione stessa. Le due espressioni producono tuttavia pattern diversi (($\{0\}_K$, K) la prima e (0, K) la seconda) e non risultano quindi equivalenti secondo la definizione di Abadi e Rogaway.

Una leggera modifica nelle definizioni dell'insieme **Pat** e della funzione $p(M, T)$ può tuttavia portare a specificare un'equivalenza in cui le due espressioni prese in considerazione risultino equivalenti.

Definiamo quindi il nuovo insieme **Pat** dei patterns attraverso la seguente grammatica:

$P, Q ::=$	patterns
K	chiave ($K \in$ **Keys**)
i	bit ($i \in$ **Bool**)
(P, Q)	accoppiamento
$\otimes$	testo non decifrabile

Ovvero un pattern così definito non contiene più parti cifrate.

Quindi modifichiamo la funzione $p(M, T)$ in modo che non mantenga traccia delle chiavi di cifratura quando incontra dei blocchi cifrati di cui si conosce la chiave:

$p(K, T)$	=	K	(con $K \in$ **Keys**)
$p(i, T)$	=	i	(con $i \in$ **Bool**)
$p((M, N), T)$	=	$(p(M, T), p(N, T))$	
$p(\{M\}_K, T)$	=	$\boldsymbol{p(M, T)}$	se $K \in T$
		$\otimes$	altrimenti.

Con queste assunzioni entrambe le espressioni $(\{0\}_K, K)$ e $(0, K)$ producono il pattern $(0, K)$ e risultano quindi equivalenti.

4.3 Equivalenza probabilistica di espressioni

Con questa sezione si conclude la presentazione del modello formale per la crittografia. L'idea che si vuole introdurre è quella di un'equivalenza tra espressioni che tenga conto della possibilità di un nemico di decifrare un'espressione $\{M\}_K$ pur non conoscendo formalmente la chiave K necessaria a tale scopo. Cade quindi l'assunzione di crittografia perfetta, generalmente utilizzata nella definizione di modelli crittografici formali.
Introduciamo la nozione di *espressione probabilistica*: un'espressione probabilistica è un'espressione *non cifrata* a cui viene associato un parametro probabilistico $p \in]0,1]$ che rappresenta la probabilità con cui

l'espressione può essere messa in chiaro.

Formalmente un'espressione probabilistica è una coppia (P, p) in cui P è un'espressione non cifrata e $p \in]0,1]$ è un parametro probabilistico; per semplicità scriveremo $P_{.p}$ al posto di (P, p)[5]. Indichiamo quindi con **pExp** l'insieme delle espressioni probabilistiche ottenute con la seguente grammatica:

$P_{.p}, Q_{.p} ::=$	espressioni probabilistiche
$K_{.p}$	chiave ($K \in$ **Keys**)
$i_{.p}$	bit ($i \in$ **Bool**)
$(P_{.p}, Q_{.p})_{.p}$	accoppiamento
$p \in]0,1]$	

La nozione di espressione probabilistica ci occorre per poter definire un'equivalenza tra espressioni che tenga in considerazione la probabilità di decifrare dei dati pur non disponendo della chiave utilizzata per la cifratura. Nelle espressioni probabilistiche non compaiono più dei dati cifrati ma solo dei dati con associata la probabilità di vederli in chiaro.

Il nostro prossimo obiettivo è quindi quello di definire una modalità univoca che permetta di trasformare un'espressione generica in un'espressione probabilistica. Definiti tali meccanismi diventa poi possibile specificare una nuova equivalenza tra espressioni che confronti le espressioni probabilistiche ad esse associate.

L'espressione probabilistica associata ad un'espressione M si ottiene sostituendo tutte le parti cifrate presenti in M con delle parti in chiaro a cui è associata la probabilità di decifrarle. Ad esempio l'espressione probabilistica associata all'espressione $\{0\}_K$ è $0.p_{dec}(\{0\}_K)$, in cui $p_{dec}(\{N\}_K)$, $N \in$ **Exp** e $K \in$ **Keys**, è una funzione che calcola la

[5] Se non possono sorgere ambiguità con le espressioni generiche ometteremo in genere il parametro probabilistico p se uguale ad 1 (cioè scriveremo semplicemente P al posto di $P_{.1}$).

probabilità di ricavare l'espressione N, data l'espressione cifrata $\{N\}_K$, senza conoscere la chiave K. La funzione $p_{dec}(\{N\}_K)$ varia in base all'algoritmo di cifratura utilizzato e viene calcolata attraverso tecniche di crittoanalisi.

Ricavare l'espressione probabilistica associata ad un'espressione generica non è tuttavia sempre così semplice. Si consideri, ad esempio, l'espressione $(\{\{0\}_{K1}\}_{K2}, \{(K1, K2)\}_K)$; qual è la probabilità di mettere in chiaro il bit 0? La risposta più semplice e immediata potrebbe essere $p_{dec}(\{\{0\}_{K1}\}_{K2}) \cdot p_{dec}(\{\{0\}_{K1}\})$, ovvero la probabilità di individuare le due chiavi $K1$ e $K2$ tentando di rompere in due passi successivi il blocco $\{\{0\}_{K1}\}_{K2}$. Tuttavia analizzando meglio l'espressione si nota che con probabilità $p_{dec}(\{(K1, K2)\}_K)$ si possono mettere in chiaro le due chiavi $K1$ e $K2$, e quindi con la stessa probabilità potrebbe essere messo in chiaro anche il bit 0, cifrato appunto con le due chiavi in questione. La probabilità di mettere in chiaro il bit 0 dell'esempio precedente varia dunque a seconda dell'approccio che un nemico può avere cercando di rompere le due parti cifrate dell'espressione. L'analisi diviene ancora più interessante nel caso di espressioni particolarmente complesse. Se si considera ad esempio la seguente espressione:

$(\ (\{\{\{\{0\}_{K1}\}_{K2}\}_{K3}, \{(K1, K2)\}_K), (\{(K2, K3)\}_{K'}, \{(K1, K3)\}_{K''})\ )$

un nemico ha a disposizioni diversi modi per individuare le tre chiavi $K1$, $K2$, $K3$ che gli servono per mettere in chiaro il bit 0. Assumiamo quindi che un nemico, analizzando un'espressione e cercando di mettere in chiaro le sue parti cifrate, segua sempre l'approccio ottimo, ovvero quello che gli permette di massimizzare la probabilità di mettere in chiaro ciascuna parte cifrata. Sotto questa assunzione, ritornando all'esempio relativo all'espressione $(\{\{0\}_{K1}\}_{K2}, \{(K1, K2)\}_K)$, possiamo dire che la probabilità di mettere in chiaro il bit 0 è il valore massimo tra $p_{dec}(\{\{0\}_{K1}\}_{K2}) \cdot p_{dec}(\{\{0\}_{K1}\})$ e $p_{dec}(\{(K1, K2)\}_K)$.

Per ottenere l'espressione probabilistica associata ad un'espressione diventa quindi fondamentale associare ad ogni dato cifrato presente nell'espressione l'insieme di chiavi necessario per

metterlo in chiaro. Un'analisi di questo tipo potrebbe ad esempio essere ottenuta per mezzo della seguente funzione:

$$remEnc(M,\ T) = \begin{cases} i._T \quad \text{se } M = i, \quad i \in \mathbf{Bit} \\ K._T \quad \text{se } M = K, \quad K \in \mathbf{Keys} \\ (remEnc(N,T), remEnc(N',T))._T \quad \text{se } M = (N,N') \\ remEnc(N,T'), \quad T' = T \cup \{K\} \quad \text{se } M = \{N\}_K \end{cases}$$

La funzione *remEnc*(*M*, *T*) prende in input un'espressione *M* e un insieme di chiavi $T \subseteq$ **Keys** (che sarà inizialmente vuoto), elimina dall'espressione *M* tutte le parti cifrate ed associa ai dati fondamentali l'insieme di chiavi necessarie per metterli in chiaro. Tutte le volte che incontra un blocco cifrato la funzione *remEnc* mette in chiaro l'espressione cifrata ed aggiorna l'insieme di chiavi *T* aggiungendo la chiave di cifratura. Data l'espressione $M = (\{0\}_{K1},\ \{\{1\}_{K2}\}_{K3})$ mettendo in chiaro i dati fondamentali dell'espressione (ovvero i bit 0 e 1), e associando ad ognuno di essi le chiavi necessarie per metterli in chiaro, si ha $remEnc(M, \varnothing) = (0._{\{K1\}},\ 1._{\{K2,\ K3\}})._{\varnothing}$.

Fissata l'espressione di cui si vuole conoscere l'espressione probabilistica associata, si dovrà poi definire una funzione che, preso in input un insieme di chiavi, restituisca la probabilità massima di individuare tali chiavi analizzando, nel modo ottimale, l'espressione in questione (gli insiemi {*K1*} e {*K1*, *K2*} associati ai dati dell'esempio precedente vanno cioè trasformati in un valore probabilistico per ottenere l'espressione probabilistica finale). Per specificare tale funzione si dovranno però analizzare tutte le possibili strade che un nemico può percorrere durante la crittoanalisi dell'espressione, ovvero tutte le combinazioni di chiavi che un nemico può ottenere in tutti i possibili modi di analizzare l'espressione.

Data ad esempio l'espressione $(\{0\}_K,\ \{K\}_{K1})$ un nemico potrebbe individuare la chiave *K* decifrando il blocco $\{0\}_K$ con probabilità

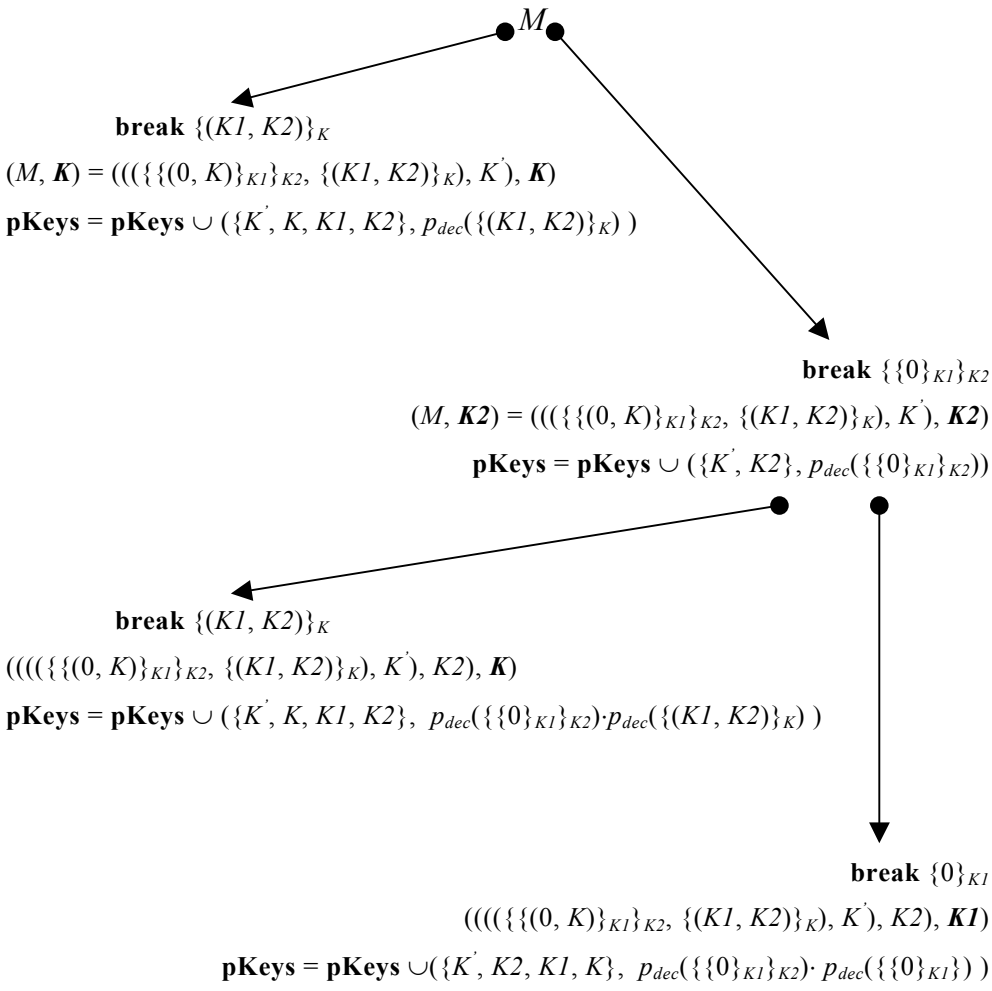
$p_{dec}(\{0\}_K)$, oppure decifrando il blocco $\{K\}_{K1}$ riesce ad individuare la chiave *K1*, e di conseguenza anche la chiave *K*, ovvero riesce ad ottenere l'insieme di chiavi $\{K1, K\}$, con probabilità $p_{dec}(\{K\}_{K1})$.

Fissata un'espressione *M*, si definisce quindi un insieme **pKeys** che contiene delle coppie (*T*, *p*) in cui $T \subseteq$ **Keys** è un insieme di chiavi ricavabili in qualche modo da *M* e $p \in]0,1]$ è la probabilità di ricavare l'insieme di chiavi *T*. Ogni insieme **pKeys**, associato ad un'espressione fissata *M*, conterrà la coppia (*T*, 1) in cui *T* è un insieme di chiavi, eventualmente vuoto, ricavabile da *M* con probabilità 1. Tale insieme è costituito da tutte le chiavi $K \in$ **Keys** : $M \mapsto K$, ovvero da tutte le chiavi *K* presenti in *M* in chiaro o che si ricavano da *M* attraverso la relazione di entailment mostrata nel paragrafo precedente. L'insieme **pKeys** associato all'espressione $((\{0\}_{K'}, K1), \{K2\}_{K1})$ contiene, ad esempio, l'elemento $(\{K1, K2\}, 1)$. **pKeys** conterrà poi insiemi che oltre alle chiavi che si ricavano da *M* attraverso l'entailment contengono anche chiavi che si ricavano da *M* attraverso la crittoanalisi dei blocchi cifrati. Per qualsiasi sequenza di crittoanalisi di blocchi cifrati che un nemico può effettuare l'insieme **pKeys** memorizza l'insieme di chiavi ricavabili da *M* e la probabilità con cui tale insieme viene ricavato.

Vediamo quindi ad esempio come si ricava l'insieme **pKeys** associato all'espressione $M = ((\{\{(0, K)\}_{K1}\}_{K2}, \{(K1, K2)\}_K), K')$:

$M \mapsto K' \quad \Rightarrow \quad$ **pKeys** $= \{ (\{K'\}, 1) \}$

L'insieme **pKeys** viene cioè inizializzato con le chiavi che si ricavano da *M* con probabilità 1. Valutiamo adesso tutte le possibili sequenze di crittoanalisi che aggiungono nuovi elementi all'insieme **pKeys**:



L'insieme **pKeys** viene inizializzato con l'insieme di chiavi che si ricavano da *M* con probabilità 1; nell'esempio con l'elemento $(\{K'\}, 1)$. Quindi si effettua uno studio di tutte le possibili sequenze in cui un nemico può analizzare i blocchi cifrati presenti nell'espressione. Tale studio potrebbe essere effettuato, ad esempio, per mezzo della e procedura *addKeys*(*M*, 1); in cui *addKeys*(*L*, *p*), con $L \in$ **Exp** e $p \in]0, 1]$, è definita informalmente nel seguente modo:

addKeys(*L*, *p*) :: =

Per ogni blocco cifrato che si ricava da *L* di cui non si conosce la chiave:

1. Sia *p*' la probabilità di rompere il blocco.
2. Sia *L*' l'espressione che si ottiene accoppiando *L* con la chiave che cifra il blocco.

3. **pKeys** = **pKeys** $\cup$ ($\{K \in$ **Keys** $\mid L' \mapsto K\}, p \cdot p'\}$.
4. *addKeys*(L', $p \cdot p'$).

All'espressione $((\{\{(0, K)\}_{K1}\}_{K2}, \{(K1, K2)\}_K), K')$ è associato dunque l'insieme **pKeys** = {
$(\{K'\}, 1)$,
$(\{K', K, K1, K2\}, p_{dec}(\{(K1, K2)\}_K))$,
$(\{K', K2\}, p_{dec}(\{\{0\}_{K1}\}_{K2}))$,
$(\{K', K2, K, K1\}, p_{dec}(\{\{0\}_{K1}\}_{K2}) \cdot p_{dec}(\{(K1, K2)\}_K)$,
$(\{K', K2, K1, K\}, p_{dec}(\{\{0\}_{K1}\}_{K2}) \cdot p_{dec}(\{\{0\}_{K1}\}))$ }

Come si può notare l'insieme di chiavi $\{K', K, K1, K2\}$ appare più volte all'interno dell'insieme **pKeys** con diversi valori di probabilità. Ciò è dovuto ai diversi modi che un nemico ha a disposizione per individuare le tre chiavi in questione, ovvero al diverso ordine con cui può decifrare i vari blocchi.

A questo punto è possibile definire una funzione *pGuess*(T), con $T \subseteq$ **Keys**, che, fissata un'espressione M, restituisce la probabilità massima di ricavare le chiavi in T analizzando l'espressione M. Infatti fissata un'espressione M e ricavato l'insieme **pKeys** ad essa associata *pGuess*(T) potrebbe essere definita come:

$$pGuess(T) = \max \{ p \mid (J, p) \in \textbf{pKeys} \ \wedge \ T \subseteq J \}$$

Con l'ausilio della funzione *pGuess*(T) appena introdotta definiamo una nuova funzione che chiamiamo *pExp*(M, T), in cui $M \in$ **Exp** e $T \subseteq$ **Keys**, che, presi in input un'espressione M ed un insieme di chiavi T inizialmente vuoto, trasforma l'espressione M in un'espressione probabilistica. *pExp*(M, T) potrebbe essere definita come segue:

$$pExp(M,\ T) = \begin{cases} i._{pGuess(T)} & \text{se } M = i,\quad i \in \mathbf{Bit} \\ K._{pGuess(T)} & \text{se } M = K,\quad K \in \mathbf{Keys} \\ (pExp(N,T),\ pExp(N',T))._{pGuess(T)} & \text{se } M = (N, N') \\ pExp(N,T'),\quad T' = T \cup \{K\} & \text{se } M = \{N\}_K \end{cases}$$

Come si può notare la funzione *pExp*(*M*, *T*) è una variante della funzione *remEnc*(*M*, *T*) che ai dati cifrati dell'espressione *M* messi in chiaro associa direttamente un valore probabilistico invece di un insieme di chiavi.

Se ad esempio consideriamo ancora $M = (\{\{0\}_{K1}\}_{K2},\ \{(K1,\ K2)\}_K)$ avremo che: $pExp(M, \varnothing) = (0._{pGuess(\{K2,\ K1\})},\ (K1, K2)._{pGuess(\{K\})})._{pGuess(\varnothing)}$.

Infine, data un espressione *M*, chiamiamo p_{max} la probabilità con cui un nemico può ricavare tutte le chiavi presenti nell'espressione. Dato l'insieme **pKeys** associato all'espressione *M*, p_{max} si ricava estraendo la probabilità massima *p* dall'insieme delle coppie (*J*, *p*) in cui *J* contiene tutte le chiavi che si possono dedurre da *M*, ovvero in cui la cardinalità dell'insieme *J* è pari alla cardinalità massima che si può trovare tra gli insiemi *I* tali che $(I, p) \in \mathbf{pKeys}$. In particolare p_{max} è la probabilità massima con cui un nemico può ottenere in chiaro la totalità dell'informazione contenuta nell'espressione *M*.

$$p_{max} = \max\ \{p \mid (J, p) \in \mathbf{pKeys},\ \mid J \mid = \max\ \{n \mid n = \mid I \mid, (I, p) \in \mathbf{pKeys}\}\ \}$$

Fissata un'espressione *M* gli elementi che permettono di ricavare l'espressione probabilistica ad essa associata sono quindi l'insieme **pKeys**, le due funzioni *pGuess* e *pExp* e quindi il valore probabilistico p_{max}.

Descriviamo ora formalmente i meccanismi e le modalità che permettono di trasformare un'espressione generica in un'espressione probabilistica, riunendo le idee mostrate finora nel concetto generale di *pattern probabilistico*.

4.3.1 Pattern Probabilistico

Definizione 4.3. *Pattern probabilistico*
Un pattern probabilistico è una tupla (M, **pKeys**, *pGuess*, *pExp*, p_{max}) in cui:

- $M \in$ **Exp**.
- **pKeys** è un insieme di coppie (T, p) in cui $T \subseteq$ **Keys** e $p \in]0,1]$ è una probabilità[6].
- *pGuess*: $\mathcal{D}_{pGuess} \rightarrow]0,1]$
  $\mathcal{D}_{pGuess} = \{T \in \mathcal{P}(\textbf{Keys}) \mid \exists\, J_{.p} \in \textbf{pKeys} \;:\; T \subseteq J\}$
- *pExp*: **Exp** $\times$ $\mathcal{D}_{pGuess} \rightarrow$ **pExp**.
- $p_{max} \in]0,1]$

Chiamiamo dunque **pPat** l'insieme dei pattern probabilistici contenente tutte le tuple (M, **pKeys**, *pGuess*, *pExp*, p_{max}) $\forall M \in$ **Exp**.

In un pattern probabilistico (M, **pKeys**, *pGuess*, *pExp*, p_{max}) gli elementi **pKeys**, *pGuess*, *pExp* e p_{max} dipendono funzionalmente da M. Di seguito mostriamo come questi elementi vengono ricavati formalmente dall'espressione M.

- **pKeys** viene generato attraverso il seguente algoritmo:

pKeys = { { $K \in$ **Keys** | $M \mapsto K$}$_{.1}$ } }
addKeys(M, 1)

addKeys(L, p), in cui $L \in$ **Exp** e $p \in]0,1]$, è la seguente procedura:

[6] Per semplicità indichiamo gli elementi di **pExp** con la notazione $T_{.p}$ al posto di (T, p).

addKeys(L, p) :: =

$\forall \{N\}_K : (L \mapsto \{N\}_K \ \wedge L \not\mapsto K)$ *do begin*

$p^{'}$ = $p \cdot p_{dec}(\{N\}_K, L)$

$L^{'}$ = (L, K)

T = $\{K \in \mathbf{Keys} \mid L^{'} \mapsto K\}$

pKeys = **pKeys** $\cup$ $\{T._{p'}\}$

addKeys($L^{'}$, $p^{'}$);

end;

in cui $p_{dec}(\{N\}_K, L)$ indica la probabilità di un nemico, che ha una conoscenza definita dall'espressione L, di decifrare l'espressione $\{N\}_K$ pur non possedendo la chiave K (la conoscenza L viene omessa se non disponibile, in questo caso scriveremo semplicemente $p_{dec}(\{N\}_K)$). Il valore di $p_{dec}(\{N\}_K, L)$ viene stimato attraverso tecniche di crittoanalisi in base al tipo di algoritmo di cifratura utilizzato. Il motivo per cui abbiamo aggiunto l'espressione L come input della funzione p_{dec}, rispetto alla trattazione introduttiva in cui p_{dec} prendeva in input solo il blocco cifrato $\{N\}_K$, è legato al fatto che in molte tecniche di crittoanalisi la probabilità di rompere un blocco dipende dalla quantità di informazione (e in particolare di testo cifrato) di cui un nemico dispone [59]. L'espressione L rappresenta quindi la conoscenza del nemico nel momento in cui effettua il tentativo di rompere lo schema di cifratura.

Un nemico analizzando un'espressione e cercando di rompere le parti cifrate di cui non conosce la chiave può trovarsi di fronte a diverse possibili strade che possono portare a situazioni differenti. Si consideri ad esempio l'espressione $N = (\{\{\{0\}_{K2}\}_{K1}\}_K \ , \ \{((K1, K2), K)\}_{K'})$. Se un nemico comincia ad analizzare la componente sinistra dell'espressione N, $\{\{\{0\}_{K2}\}_{K1}\}_K$, può ottenere le tre chiavi K, $K1$ e $K2$ solamente attraverso tre distinti tentativi di rompere lo schema e con una probabilità pari a:
$p_{dec}(\{\{\{0\}_{K2}\}_{K1}\}_K, N) \cdot p_{dec}(\{\{0\}_{K2}\}_{K1}, (N, K)) \cdot p_{dec}(\{0\}_{K2}, ((N, K), K1))$.
Se invece il nemico analizza prima la componente destra dell'espressione N, $\{((K1, K2), K)\}_{K'}$ può ottenere le tre chiavi K, $K1$ e $K2$ con un solo

tentativo di rompere lo schema e con una probabilità pari a:

$$p_{dec}(\{((K1, K2), K)\}_{K'}, N).$$

L'insieme **pKeys** appena definito contiene tutti i possibili insiemi di chiavi, con associata la probabilità di ricavarli, che un nemico può individuare attraverso tutte le possibili strade percorribili analizzando l'espressione *M*. Come vedremo nel seguito l'insieme **pKeys** avrà un ruolo fondamentale nel calcolo dell'espressione probabilistica ottenuta da un'espressione generica.

Di seguito mostriamo un esempio di come viene costruito un insieme **pKeys**. Per semplicità consideriamo costante il valore di $p_{dec}(\{N\}_K, L)$ per ogni $\{N\}_K$ e per ogni L e fissiamo $p_{dec}(\{N\}_K, L) = p_{dec}$.

Sia $M = ((\{(K1, K2)\}_{K'}, \{K\}_{K1}), K)$:

$M \mapsto K \quad \Rightarrow$

pKeys = $\{ \{K\}_{.1} \}$

addKeys(M, 1):: =

- $M \mapsto \{(K1, K2)\}_{K'} \wedge M \nmapsto K' \quad \Rightarrow$

  pKeys = **pKeys** $\cup$ $\{ \{K', K, K1, K2\}_{.p_{dec}} \}$

  $M1 = (M, K')$

  addKeys($M1$, p_{dec}) :: = **void**

- $M \mapsto \{K\}_{K1} \wedge M \nmapsto K1 \quad \Rightarrow$

  pKeys = **pKeys** $\cup$ $\{ \{K1, K\}_{.p_{dec}} \}$

  $M3 = (M, K1)$

  addKeys($M3$, p_{dec}) :: =

  - $M3 \mapsto \{(K1, K2)\}_{K'} \wedge M3 \nmapsto K' \Rightarrow$

    pKeys = **pKeys** $\cup$ $\{ \{K', K, K1, K2\}_{.p_{dec}^2} \}$

    $M4 = (M3, K')$

    addKeys($M4$, p_{dec}^2) :: = **void**

Alla fine otteniamo quindi:

pKeys = $\{\{K\}_{.1}, \{K',K,K1,K2\}_{.p_{dec}}, \{K1,K\}_{.p_{dec}}, \{K',K,K1,K2\}_{.p_{dec}^2}\}$

Come si può notare l'insieme di chiavi $\{K', K, K1, K2\}$ compare due volte con due diversi valori di probabilità. Ciò è dovuto, come abbiamo visto, al diverso approccio che può avere un nemico durante l'analisi dell'espressione. Se infatti un nemico analizzando l'espressione M comincia a decifrare prima la parte di sinistra $\{(K1, K2)\}_{K'}$, con probabilità p_{dec}, trova subito la chiave $K1$ che può poi usare per decifrare l'espressione $\{K\}_{K1}$ senza *sforzi* aggiuntivi (ovvero con probabilità p_{dec} può derivare tutte le chiavi dell'insieme $\{K', K, K1, K2\}$). Se invece analizzando l'espressione M un nemico decifra prima la parte di destra $\{K\}_{K1}$, sempre con probabilità p_{dec}, non trova informazioni utili per decifrare l'espressione di sinistra $\{(K1, K2)\}_{K'}$, che però può ancora una volta essere messa in chiaro con probabilità p_{dec} (in questo caso quindi un nemico individua tutte le chiavi dell'insieme $\{K', K, K1, K2\}$ con probabilità p_{dec}^2).

- *pGuess*: $\mathcal{D}_{pGuess} \rightarrow]0,1]$ è definita formalmente come:

$$\mathcal{D}_{pGuess} = \{T \in \mathcal{P}(\mathbf{Keys}) \mid \exists\, J._p \in \mathbf{pKeys} \;:\; T \subseteq J\}$$
$$pGuess(T) = \max\ \{\ p \mid J._p \in \mathbf{pKeys} \ \wedge\ T \subseteq J\ \}$$

Dato un insieme $T \in \mathcal{P}(\mathbf{Keys}) \mid \exists\, J._p \in \mathbf{pKeys} : T \subseteq J$ la funzione $pGuess(T)$ calcola la probabilità massima di un nemico di individuare tutte le chiavi presenti nell'insieme T analizzando l'espressione M. Calcola cioè la probabilità con cui un nemico può individuare tutte le chiavi dell'insieme T attraverso il percorso migliore che l'espressione M gli permette (tutti i possibili approcci di un nemico durante l'analisi di M sono infatti contenuti nell'insieme **pKeys**). La condizione sul dominio della funzione *pGuess* implica che tutte le chiavi in T siano in qualche modo deducibili dall'espressione M, ovvero che siano presenti in qualche elemento dell'insieme **pKeys**.

Mostriamo alcuni esempi (consideriamo ancora costante il valore di

$p_{dec}(\{N\}_K, L)$ per ogni $\{N\}_K$ e per ogni L e fissiamo $p_{dec}(\{N\}_K, L) = p_{dec}$):

$pGuess(\varnothing) = 1 \qquad \forall\ (M, \mathbf{pKeys}, pGuess, pExp, p_{max}) \in \mathbf{pPat}$

infatti $\forall M \in \mathbf{Exp} \quad \{K \in \mathbf{Keys} \mid M \mapsto K\}_{.1} \in \mathbf{pKeys}$

e $\varnothing \subseteq \{K \in \mathbf{Keys} \mid M \mapsto K\}$

sia $M = ((\{(K1,\ K2)\}_{K'},\ \{K\}_{K1}),\ K)$, come visto in precedenza ad M è associato l'insieme:

$\mathbf{pKeys} = \{\ \{K\}_{.1},\ \{K',K,K1,K2\}.p_{dec},\ \{K1,K\}.p_{dec},\ \{K',K,K1,K2\}.p_{dec}^{\,2}\ \}$

abbiamo quindi:

$pGuess(\{K\}) = 1$

$pGuess(\{K2\}) = p_{dec}$

$pGuess(\{K',\ K,\ K1,\ K2\}) = p_{dec}$

sia invece $M = ((\{(K1,\ K2)\}_K,\ \{(K3,\ K4)\}_{K'}),\ \{\{\{\{K5\}_{K4}\}_{K3}\}_{K2}\}_{K1})$, allora:

$\mathbf{pKeys}=\{\ \{\varnothing\}_{.1},\ \{K,\ K1,\ K2\}.p_{dec},\ \{K,\ K1,\ K2,\ K',\ K3,\ K4,\ K5\}.p_{dec}^{\,2},$

$\{K,\ K1,\ K2,\ K3\}.p_{dec}^{\,2},\ \{K,\ K1,\ K2,\ K3,\ K',\ K4,\ K5\}.p_{dec}^{\,3},$

$\{K,\ K1,\ K2,\ K3,\ K4,\ K5\}.p_{dec}^{\,3},\ \{K,\ K1,\ K2,\ K3,\ K4,\ K5,\ K'\}.p_{dec}^{\,4}$

$\{K',\ K3,\ K4\}.p_{dec},\ \{K',\ K3,\ K4,\ K1\}.p_{dec}^{\,2},$

$\{K',\ K3,\ K4,\ K1,\ K2,\ K5\}.p_{dec}^{\,3},\ \{K1\}.p_{dec},\ \{K1,\ K2\}.p_{dec}^{\,2},$

$\{K1,\ K2,\ K3\}.p_{dec}^{\,3},\ \{K1,\ K2,\ K3,\ K4,\ K5\}.p_{dec}^{\,4},$

$\{K1,\ K2,\ K3,\ K4,\ K5,\ K\}.p_{dec}^{\,5},\ \{K1,\ K2,\ K3,\ K4,\ K5,\ K'\}.p_{dec}^{\,5},$

$\{K1,\ K2,\ K3,\ K4,\ K5,\ K,\ K'\}.p_{dec}^{\,6}\ \}$

abbiamo quindi:

$pGuess(\{K\}) = p_{dec}$

$pGuess(\{K5\}) = p_{dec}^{\,2}$

$pGuess(\{K1,\ K2\}) = p_{dec}$

$pGuess(\{K',\ K4\}) = p_{dec}$

$pGuess(\{K,\ K1,\ K2,\ K',\ K3\}) = p_{dec}^{\,2}$

$pGuess(\{K,\ K1,\ K2,\ K',\ K3,\ K4,\ K5\}) = p_{dec}^{\,2}$

- $pExp$: **Exp** × $\mathcal{D}_{pGuess}$ → **pExp** è definita come:

$pExp(K, T)$	=	$K_{.pGuess(T)}$	($K \in$ **Keys**)
$pExp(i, T)$	=	$i_{.pGuess(T)}$	($i \in$ **Bool**)
$pExp((N, N'), T)$	=	$(pExp(N, T), pExp(N', T))_{.pGuess(T)}$	
$pExp(\{N\}_K, T)$	=	$pExp(N, T')$	($T'=T \cup \{K\}$)

Per trasformare un'espressione in un'espressione probabilistica utilizziamo quindi la funzione $pExp(M, T)$, in cui $M \in$ **Exp** e $T \in \mathcal{D}_{pGuess}$ è un insieme di chiavi inizialmente vuoto. $pExp(M, T)$ trasforma l'espressione M modificando tutte le sottoespressioni cifrate presenti in M in espressioni in chiaro, associando loro un insieme T di chiavi necessarie per mettere in chiaro la sottoespressione. Ogni insieme di chiavi T viene poi convertito in un valore probabilistico attraverso la funzione $pGuess(T)$. Vediamo alcuni esempi:

$M = (\{\{0\}_{K2}\}_{K1}, 1)$ $(M, \textbf{pKeys}, pGuess, pExp, p_{max}) \in \textbf{pPat}$

$pExp(M, \varnothing) = (0_{.pGuess(\{K1, K2\})}, 1_{.pGuess(\varnothing)})_{.pGuess(\varnothing)}$

ma come abbiamo visto $pGuess(\varnothing) = 1$ e ricordando che per le espressioni probabilistiche $P_{.p}$ omettiamo il parametro probabilistico p se uguale ad 1 abbiamo $pExp(M, \varnothing) = (0_{.pGuess(\{K1, K2\})}, 1)$

$M = \{\{K\}_{K2}\}_{K1}$ $(M, \textbf{pKeys}, pGuess, pExp, p_{max}) \in \textbf{pPat}$

$pExp(M, \varnothing) = K_{.pGuess(\{K1, K2\})}$

$M = \{(\{\{0\}_{K2}\}_{K1}, 1)\}_K$ $(M, \textbf{pKeys}, pGuess, pExp, p_{max}) \in \textbf{pPat}$

$pExp(M, \varnothing) = (0_{.pGuess(\{K, K1, K2\})}, 1_{.pGuess(\{K\})})_{.pGuess(\{K\})}$

- $p_{max} \in]0,1]$ viene calcolato nel seguente modo:

$$p_{max} = \max \{p \mid (J, p) \in \textbf{pKeys}, \ |J| = \max \{n \mid n = |I|, (I, p) \in \textbf{pKeys}\}\}$$

Come abbiamo visto, p_{max} rappresenta la probabilità con cui un nemico può ricavare tutte le chiavi presenti nell'espressione M. p_{max} si ricava estraendo la probabilità massima p dall'insieme delle coppie (J, p) in cui J contiene tutte le chiavi che si possono dedurre da M, ovvero in cui la cardinalità di J è pari alla cardinalità massima che si può trovare tra gli insiemi I tali che $(I, p) \in$ **pKeys**. In particolare p_{max} è la probabilità massima con cui un nemico, conoscendo tutte le chiavi che occorrono in M, può ottenere in chiaro la totalità dell'informazione contenuta nell'espressione.

Riproponendo l'espressione $M = ((\{(K1, K2)\}_{K'}, \{K\}_{K1}), K)$, a cui è associato l'insieme:

pKeys $= \{ \{K\}_{.1}, \{K',K,K1,K2\}.p_{dec}, \{K1,K\}.p_{dec}, \{K',K,K1,K2\}.p_{dec}^{2} \}$

abbiamo che:

$p_{max} = \max\{ p \mid (J, p) \in \textbf{pKeys}, \ |J| = 4 \} = \max\{p_{dec}, p_{dec}^{2}\} = p_{dec}$

Definizione 4.4. (*Equivalenza probabilistica di espressioni*)
Date due espressioni M e N e i pattern probabilistici (M, **pKeys**, *pGuess*, *pExp*, p_{max}) e (N, **pKeys'**, *pGuess*', *pExp*', p_{max}') associati, diremo che le due espressioni sono probabilisticamente equivalenti ($M \approx N$) se danno origine alla stessa espressione probabilistica e se i valori p_{max} e p_{max}' coincidono, cioè:

$(M, \textbf{pKeys}, pGuess, pExp, p_{max}), (N, \textbf{pKeys'}, pGuess', pExp', p_{max}') \in \textbf{pPat}$
$M \approx N \ \Leftrightarrow \ pExp(M, \varnothing) = pExp'(N, \varnothing) \wedge p_{max} = p_{max}'$

Se consideriamo le due espressioni:
$M = \{\{0\}_{K2}\}_{K1}$ e $N = \{\{0\}_{K4}\}_{K3}$
e i loro rispettivi pattern probabilistici:
(M, **pKeys**, *pGuess*, *pExp*, p_{max}) e (N, **pKeys'**, *pGuess*', *pExp*', p_{max}')
avremo che:

$$M \approx N \Leftrightarrow pExp(M, \varnothing) = pExp'(N, \varnothing) \wedge p_{max} = p_{max}'$$
$$\Leftrightarrow 0._{pGuess(\{K1, K2\})} = 0._{pGuess'(\{K3, K4\})} \wedge p_{max} = p_{max}'$$
$$\Leftrightarrow pGuess(\{K1, K2\}) = pGuess'(\{K3, K4\})$$

Intuitivamente due espressioni sono quindi equivalenti se contengono la stessa informazione e se tale informazione può essere ottenuta da un nemico con uguale probabilità.

Teorema 4.1. Date due espressioni M, $N \in$ **Exp** stabilire se le due espressioni sono probabilisticamente equivalenti ($M \approx N$) è un problema decidibile.

Verificare che due espressioni siano probabilisticamente equivalenti significa infatti ricavare i pattern probabilistici associati alle espressioni. Questi possono essere ottenuti, in tempo finito, per mezzo dell'algoritmo di costruzione dell'insieme **pKeys**. Poiché ogni espressione è, infatti, una sequenza finita di dati, l'algoritmo di costruzione dell'insieme **pKeys** viene completato in tempo finito. Infine, dal momento che l'insieme **pKyes** così generato è anch'esso finito, le due funzioni *pGuess* e *pExp* sono decidibili sulla base della loro definizione.

Non introduciamo, per l'equivalenza probabilistica, l'idea di ridenominazione delle chiavi utilizzata nella definizione di equivalenza di Abadi e Rogaway [2] che abbiamo mostrato in precedenza. Questa scelta è dovuta al ruolo, fondamentalmente diverso, giocato dalle chiavi nelle due definizioni. La definizione di un'equivalenza probabilistica ha infatti portato all'introduzione della funzione $p_{dec}(\{N\}_K, L)$ il cui valore è strettamente legato alla scelta della chiave K. Ridenominando le chiavi all'interno di un'espressione potrebbe quindi cambiare l'espressione probabilistica che si ottiene con i meccanismi mostrati sopra. Va tuttavia sottolineato che, fissato un algoritmo di cifratura ed un'espressione N, se due chiavi K e K' sono scelte in uno stesso spazio con lo stesso criterio i

valori di $p_{dec}(\{N\}_K)$ e $p_{dec}(\{N\}_{K'})$ dovrebbero essere uguali.

Concludiamo questa sezione mostrando alcuni esempi di base e facendo un confronto diretto tra le definizioni di equivalenza attraverso ridenominazione di Abadi e Rogaway e l'equivalenza probabilistica appena introdotta. Nella tab. 4.2 sono quindi riportati su due colonne alcuni esempi a confronto.

Equivalenza con ridenominazione ($\cong$)	**Equivalenza probabilistica** ($\approx$)
$0 \cong 0$ Banalmente.	$0 \approx 0$ Banalmente.
$0 \ncong 1$ Banalmente.	$0 \not\approx 1$ Banalmente.
$\{0\}_K \cong \{1\}_K$ Entrambe le espressioni producono infatti il pattern $\otimes$, assumendo che un nemico non sia in grado di decifrare le due espressioni in questione.	$\{0\}_K \not\approx \{1\}_K$ Le due espressioni non sono in questo caso equivalenti poiché si è assunto che un nemico è in grado, con una certa probabilità, di decifrare blocchi cifrati di cui non conosce la chiave di cifratura. Le due espressioni in questione contengono unità di informazione diverse e le espressioni probabilistiche ad esse associate sono ovviamente differenti.
$\{0\}_K \cong \{0\}_{K'}$ Come nel caso precedente entrambe le espressioni producono il pattern $\otimes$ e sono quindi equivalenti.	$\{0\}_K \approx \{0\}_{K'} \Leftrightarrow p_{dec}(\{0\}_K) = p_{dec}(\{0\}_{K'})$ In questo caso poiché le due espressioni contengono la stessa unità di informazione sono equivalenti se la probabilità di ricavarle è uguale per entrambe le espressioni. Solo in questo caso infatti le espressioni probabilistiche associate alle espressioni in questione sono uguali.
$K \cong K'$ Dal momento che stiamo analizzando l'equivalenza con ridenominazione delle chiavi le due espressioni sono banalmente equivalenti.	$K \not\approx K'$ Come abbiamo specificato in precedenza nell'equivalenza probabilistica il ruolo giocato dalle chiavi è sostanzialmente diverso da quello che svolgono nell'equivalenza di Abadi e Rogaway [2]. Non abbiamo quindi introdotto il concetto di ridenominazione delle chiavi e le due espressioni sono di conseguenza non equivalenti.
$(\{0\}_K, K) \ncong (0, K)$ In questo caso le due espressioni non sono equivalenti poiché producono pattern diversi: $(\{0\}_K, K)$ e $(0, K)$	$(\{0\}_K, K) \approx (0, K)$ Le due espressioni sono equivalenti poiché contengono le stesse informazioni ricavabili con la stessa probabilità. Entrambe producono infatti l'espressione probabilistica $(0, K)$.
$(\{\{0\}_K\}_{K'}, K) \ncong (\{\{0\}_{K'}\}_K, K)$ Infatti le due espressioni producono rispettivamente i pattern: $(\otimes, K)$ e $(\{\otimes\}_K, K)$	$M=(\{\{0\}_K\}_{K'}, K)$ $N=(\{\{0\}_{K'}\}_K, K)$ $(M, \textbf{pKeys}, pGuess, pExp, p_{max}) \in \textbf{pPat}$ $(N, \textbf{pKeys'}, pGuess', pExp', p_{max}') \in \textbf{pPat}$ $M \approx N \Leftrightarrow$ $pGuess(\{K',K\}) = pGuess'(\{K,K'\})$

Tabella 4.2. Due tipi di equivalenza tra espressioni a confronto.

5 Il modello crittografico-probabilistico

Negli ultimi anni sono state proposte diverse proprietà di sicurezza per protocolli crittografici; tra queste troviamo la *secrecy* (informazioni riservate devono essere reperibili solo dai partecipanti alla comunicazione), *authentication* (capacità di identificare con certezza gli altri partecipanti alla comunicazione), *integrity* (garanzia che il contenuto di messaggi non sia stato alterato), *non repudiation* (garanzia che un messaggio firmato non sia ripudiato dal mittente), *fairness* (durante una comunicazione nessun partecipante può ottenere un vantaggio rispetto agli altri).

Sebbene nella comunità scientifica ci sia un accordo generale sul significato di tali proprietà, raramente si trovano delle loro definizioni formali e nella maggior parte dei casi non sono condivisibili da tutti. Una delle cause che hanno portato a questa situazione potrebbe essere la mancanza di un unico modello formale sul quale impostare il problema e quindi fissare le definizioni formali. In particolare, definito un nuovo modello e proposta una definizione formale, è spesso difficile paragonarla alle altre a causa delle diverse assunzioni matematiche legate al modello.

La nostra idea è che un approccio classico allo studio della sicurezza, come quello utilizzato per l'analisi dei flussi di informazione in sistemi multilivello [35], possa essere usato proficuamente anche per l'analisi di proprietà di sicurezza di protocolli crittografici.

Nel primo paragrafo di questo capitolo definiremo un nuovo linguaggio che permetta di modellare il comportamento probabilistico di sistemi e di specificare proprietà di sicurezza di protocolli crittografici. Nel secondo paragrafo, infine, riprendendo l'analisi della sicurezza di protocolli crittografici sviluppata da Focardi, Gorrieri e Martinelli [17, 18], definiremo alcune delle principali proprietà di sicurezza di sistemi crittografici.

5.1 Un linguaggio crittografico-probabilistico

In questo paragrafo estenderemo il linguaggio probabilistico illustrato nella sezione **3.1** con due nuovi operatori che permettano di modellare operazioni di crittografia. Avremo quindi a disposizione un modello per il quale varranno tutte le considerazioni mostrate per il caso probabilistico (sarà cioè in grado di rilevare flussi di informazione non-deterministici e probabilistici, permetterà di stabilire una misura del livello di sicurezza di sistemi e di valutarne le prestazioni), e che potrà, inoltre, essere utilizzato per l'analisi della sicurezza di sistemi crittografici. Dal momento che gli operatori utilizzati per modellare le operazioni di crittografia sono basati sul modello illustrato nel capitolo **4**, e, in particolare, in assenza dell'assunzione di crittografia perfetta, sarà possibile analizzare anche eventuali attacchi all'algoritmo di cifratura.

Nel primo paragrafo verrà illustrata la sintassi del nuovo linguaggio formale: all'algebra di processi probabilistica *PPA* illustrata nella sezione **3.1**, seguendo l'approccio di Focardi, Gorrieri e Martinelli [17, 18], verranno aggiunte le principali primitive di crittografia, necessarie per la verifica delle proprietà di sicurezza di protocolli crittografici, ed un supporto per gestire il passaggio di valori. Nel secondo paragrafo mostreremo la semantica formale di questo nuovo linguaggio, che, in accordo con il modello probabilistico, è ancora definita attraverso *GRTS*. Infine nel terzo paragrafo proporremo alcuni esempi.

5.1.1 Crypto Probabilistic Process Algebra

La sintassi del linguaggio *Crypto Probabilistic Process Algebra* (*CryptoPPA*) è basata sui seguenti elementi:

- Un insieme **AType** = $\{a, b, c, \ldots\}$ contenete tutti i *tipi* di azione. All'interno di **AType** è anche contenuta l'azione interna τ.
- Due insiemi disgiunti $\mathbf{AType_H}$ e $\mathbf{AType_L}$, rispettivamente di azioni di *alto livello* ed azioni di *basso livello*, che costituiscono una copertura dell'insieme $\mathbf{AType}-\{\tau\}$.
- L'insieme delle espressioni **Exp**, definito nella sezione **4.1** che rappresenta la totalità dei messaggi che possono essere trasmessi.
- Una funzione $Exp : \mathbf{AType}-\{\tau\} \rightarrow \mathcal{P}(\mathbf{Exp})$ che, per ogni tipo di azione in $\mathbf{AType}-\{\tau\}$, restituisce l'insieme delle espressioni che possono essere trasmesse da azioni di quel tipo.
- Gli insiemi $\mathbf{RAct} = \{a^*(M) \mid a \in \mathbf{AType}\text{-}\{\tau\},\ M \in Exp(a)\}$ e $\mathbf{GAct} = \{a(M) \mid a \in \mathbf{AType}\text{-}\{\tau\},\ M \in Exp(a)\} \cup \{\tau\}$ che sono rispettivamente gli insiemi di azioni reattive e generative. Si noti che l'azione τ è considerata un'azione generativa dal momento che rappresenta una transizione interna autonomamente eseguita da un processo senza alcuna interazione con l'ambiente esterno.
- L'insieme delle azioni $\mathbf{Act} = \mathbf{RAct} \cup \mathbf{GAct}$ i cui elementi verranno rappresentati nella forma $\pi, \pi', \ldots$.
- L'insieme delle variabili **Var** rappresentate da $x, x', \ldots$.
- L'insieme **Const** di termini costanti i cui elementi sono rappresentati da $A, B, \ldots$.

L'insieme $\mathcal{L}$ dei termini processo rappresentati da P, Q, ...[7] è generato dalla seguente grammatica:

$$P ::= \underline{0} \mid \pi^*(x).P \mid \pi(e).P \mid \tau.P \mid P +^p P \mid P \|_S^p P \mid P \setminus L \mid P /_a^p \mid A(e_1, \ldots, e_n) \mid [e \approx e'] P \prec P \mid [e^* \mapsto_{rule} x] P \prec P$$

In cui $p \in]0, 1[$ è una probabilità; x è una variabile; $e^* \in \{e, <e, e'>\}$ e infine e, e', e_1, …, e_n sono espressioni o variabili. Indichiamo con $\mathcal{G}$ l'insieme dei termini di $\mathcal{L}$ guardati e chiusi [19]. Infine indichiamo con $\mathcal{G}_{\mathbf{H}}$ l'insieme dei termini di alto livello (processi che effettuano solamente azioni di alto livello), $\mathcal{G}_{\mathbf{H}} = \{P \in \mathcal{G} \mid sort(P) \subseteq \mathbf{AType_H}\}$.

Vediamo ora nel dettaglio i vari operatori del linguaggio (dal momento che la maggior parte di questi sono già stati definiti nella sezione **3.1.2** ne eviteremo la descrizione dettagliata):

$\underline{0}$ indica un termine inattivo che non può effettuare nessuna altra transizione. Potrebbe rappresentare un processo terminato o che si trovi in uno stato di stallo[8].

$\pi^*(x).P$ in cui x è una variabile, rappresenta l'operatore di prefisso con passaggio di valore in input. $\pi^*(x).P$, attraverso una transizione reattiva $\pi^*(M) \in \mathbf{RAct}$ eseguita con probabilità 1, prende in input l'espressione $M \in Exp(\pi)$, comunicata da una transizione generativa $\pi(M)$, e quindi si comporta come il processo P in cui la variabile x viene sostituita con l'espressione M ($P[M/x]$).

$\pi(M).P$ in cui $M \in Exp(\pi)$, rappresenta invece l'operatore di prefisso con passaggio di valore in output. In questo caso il processo $\pi(M).P$,

[7] Manteniamo la stessa notazione usata per la specifica di *PPA* poiché da qui in avanti riferiremo solamente il linguaggio *CryptoPPA*; non possono quindi sorgere ambiguità.

[8] Generalmente abbrevieremo i termini $P.\underline{0}$ omettendo il processo $\underline{0}$ e scrivendo semplicemente P.

attraverso l'esecuzione della transizione generativa $\pi(M) \in \mathbf{GAct}$ eseguita con probabilità 1, comunica l'espressione M e quindi si comporta come il processo P.

$\boldsymbol{\tau.P}$ in cui τ rappresenta l'azione interna, esegue l'azione interna τ con probabilità 1 e quindi si comporta come il processo P.

$\boldsymbol{P +^{p} Q}$ rappresenta l'operatore di composizione alternativa (si veda la definizione analoga nella sezione **3.1.2**).

$\boldsymbol{P} \|_{S}^{p} \boldsymbol{Q}$ rappresenta l'operatore di composizione parallela (si veda la definizione analoga nella sezione **3.1.2**). Essendo *CryptoPPA* un linguaggio che permette il passaggio di valori abbiamo deciso, per evitare che sorgessero ambiguità durante la definizione delle proprietà di sicurezza, di eliminare la possibilità di sincronizzazione tra azioni reattive. Questa è l'unica differenza con l'operatore di composizione parallela definito per il linguaggio *PPA*.

$\boldsymbol{P \setminus L}$ in cui $L \subseteq \mathbf{Atype}\text{-}\{\tau\}$, è l'operatore di restrizione e previene l'esecuzione di tutte le azioni il cui tipo è in L.

$\boldsymbol{P}/_{a}^{p}$ in cui $a \in \mathbf{AType}$ e p è ancora una probabilità, indica l'operatore di mascheramento. Questo operatore trasforma le azioni generative e reattive del tipo a nell'azione generativa interna τ, in accordo con le regole già specificate nella sezione **3.1.2**.

$\boldsymbol{A(x_1, \ldots, x_n)}$ in cui $x_1, \ldots, x_n$ sono variabili, definisce un processo costante. Questo operatore viene utilizzato per specificare sistemi ricorsivi. Le variabili $x_1, \ldots, x_n$ sono sostituite dalle espressioni $M_1,\ldots,M_n$ specificate al momento della chiamata del termine A attraverso l'operatore $A(M_1, \ldots, M_n)$. Generalmente quando si definisce un sistema attraverso un'algebra di processi viene specificato anche un insieme di termini costanti nella forma $A(x_1, \ldots, x_n) \triangleq P$, in cui $P \in \mathcal{G}$.

$\boldsymbol{[M \approx N]\ P \prec Q}$[9] in cui M e N sono espressioni e $\approx$ è la notazione dell'equivalenza probabilistica tra espressioni, è l'operatore di controllo

[9] Generalmente scriveremo $[M \equiv_p N]P$ al posto di $[M \equiv_p N]P \prec \underline{0}$.

di equivalenza tra espressioni. L'operatore effettua un controllo tra le due espressioni M ed N, se queste risultano equivalenti, cioè se $M \approx N$ in accordo con la definizione 4.4 di equivalenza probabilistica, il processo si comporta come il termine P, altrimenti si comporta come il termine Q. $\mathbf{[}\boldsymbol{M}^{*} \mapsto_{rule} \boldsymbol{x}\mathbf{]}\ \boldsymbol{P} \prec \boldsymbol{Q}$[10] in cui M^{*} è un'espressione oppure una coppia di espressioni $<N_1, N_2>$, rappresenta l'operatore di derivazione attraverso regola. Tale operatore viene utilizzato per gestire le espressioni e la crittografia. Attraverso le possibili applicazioni di $\mapsto_{rule}$ diviene infatti possibile la costruzione di espressioni complesse utilizzando le operazioni di accoppiamento e cifratura. In basso mostriamo le regole che permettono di modellare le applicazioni di $\mapsto_{rule}$. Tali regole sono scritte nella forma $\mapsto^{p'}_{rule}$, in cui $p' \in]0,1]$ è un parametro probabilistico. L'operatore $[M^{*} \mapsto_{rule} x]\ P \prec Q$, definito con una regola $\mapsto^{p'}_{rule}$, deriva con probabilità p' un'espressione N a partire da M^{*}. Il processo si comporterà quindi, con probabilità p', come $P[N/x]$ (ovvero come il termine P in cui la variabile x assume il valore dell'espressione derivata N) oppure come il processo Q con probabilità $1\text{-}p'$.

$$\frac{< N_1, N_2 >}{(N_1, N_2)}\ (\mapsto^{1}_{pair}) \qquad \frac{(N, N')}{N}\ (\mapsto^{1}_{fst}) \qquad \frac{(N', N)}{N}\ (\mapsto^{1}_{snd})$$

$$\frac{< N, K >}{\{N\}_K}\ (\mapsto^{1}_{enc}) \qquad \frac{< \{N\}_K, N' >, N' \mapsto K}{N}\ (\mapsto^{1}_{dec})$$

$$\frac{< \{N\}_K, N' > ((\{N\}_K, N'), \mathbf{pKeys}, pGuess, pExp, p_{max}) \in \mathbf{pPat}}{(N, K)}\ (\mapsto^{pGuess(\{K\})}_{break})$$

Sistema di regole per la manipolazione di espressioni e la gestione della crittografia, in cui N_1, N_2, N, $N' \in \mathbf{Exp}$ e $K \in \mathbf{Keys}$.

[10] Generalmente scriveremo $[M^{*} \mapsto_{rule} x]P$ al posto di $[M^{*} \mapsto_{rule} x]P \prec \underline{0}$.

Come mostrato nel capitolo precedente, dato il pattern probabilistico (($\{N\}_K$, N'), **pKeys**, *pGuess*, *pExp*, p_{max}), *pGuess*($\{K\}$) è la probabilità di individuare la chiave K analizzando l'espressione ($\{N\}_K$, N').

Le due regole $\mapsto^{1}_{dec}$ e $\mapsto^{p'}_{break}$ sono utilizzate per decifrare espressioni crittate, la prima viene utilizzata per modellare processi onesti (che decifrano solamente se conoscono la chiave), la seconda viene invece utilizzata per modellare processi ostili (che, pur non conoscendo la chiave, provano a decifrare un'espressione attraverso tentativi di crittoanalisi).

Quando un sistema esegue una possibile applicazione di $\mapsto_{rule}$, nei termini del nostro formalismo è come se eseguisse un'azione interna τ. Il processo onesto $[<\{N\}_K,\ K> \mapsto^{1}_{dec} x]P \prec Q$, ad esempio, decifra l'espressione $\{N\}_K$ con probabilità 1 e quindi, dopo aver eseguito la transizione $\xrightarrow{\tau,1}$ si comporta come il processo $P[N/x]$. Il processo $[<\{N\}_K, K'> \mapsto^{pGuess(\{K\})}_{break} x]\ P \prec Q$, rappresentato in fig. 5.1, dato il pattern probabilistico (($\{N\}_K$, K'), **pKeys**, *pGuess*, *pExp*, p_{max}), può decifrare l'espressione $\{N\}_K$ con probabilità $pGuess(\{K\}) \neq 1$, e quindi si comporta come il processo $P[(N, K)/x]$ con probabilità $pGuess(\{K\})$ o come il processo Q con probabilità 1-$pGuess(\{K\})$.

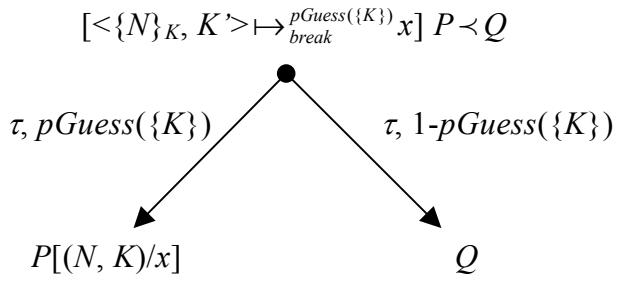


Fig. 5.1. *GRTS* del processo $[<\{N\}_K, K'> \mapsto^{pGuess(\{K\})}_{break} x]\ P \prec Q$.

E' opportuno, infine, sottolineare che per la regola:

$$\frac{< \{N\}_K, N' >,\ ((\{N\}_K, N'), \mathbf{pKeys}, pGuess, pExp, p_{max}) \in \mathbf{pPat}}{(N, K)} \quad (\mapsto_{break}^{pGuess(\{K\})})$$

l'espressione *N'* potrebbe rappresentare la conoscenza di un nemico che tenta di decifrare il blocco $\{N\}_K$.

Per evitare ambiguità introduciamo la relazione di precedenza tra gli operatori: prefisso = controllo di equivalenza delle espressioni = controllo derivazione delle espressioni > restrizione = mascheramento > composizione alternativa > composizione parallela.

Valgono inoltre tutte le altre assunzioni specificate nella sezione **3.1.2**.

5.1.2 La semantica formale

In questa sezione verrà illustrata la semantica formale del linguaggio. La notazione utilizzata per definire le regole di semantica operazionale è la stessa mostrata nella sezione **3.1.3** usata per definire la semantica di *PPA*.

La semantica formale di *CryptoPPA* è data dal sistema di transizioni generative-reattive ($\mathcal{G}$, **AType**, T) i cui stati sono i termini del calcolo e la relazione delle transizioni T è il minimo multiinsieme che soddisfa le regole operazionali specificate nel seguito. Le regole contrassegnate dal simbolo **(L)** fanno riferimento a transizioni del processo di sinistra *P*. Per semplicità abbiamo omesso le regole simmetriche che tengono in considerazione le transizioni del processo di destra *Q*; queste si possono facilmente ottenere scambiando i ruoli dei termini *P* e *Q* e sostituendo *p* con 1-*p* nell'etichetta delle transizioni derivate. Vediamo adesso nel dettaglio, per ogni operatore del linguaggio, le regole di semantica operazionale che permettono la mappatura dei termini del nostro linguaggio in *GRTS*.

Prefisso

$$\frac{M \in Exp(\pi)}{\pi^*(x).P \xrightarrow{\pi^*(M),1} P[M/x]} \qquad \frac{M \in Exp(\pi)}{\pi(M).P \xrightarrow{\pi(M),1} P}$$

$$\frac{}{\tau.P \xrightarrow{\tau,1} P}$$

Composizione alternativa

(L) $\dfrac{P \xrightarrow{a^*,q} P' \quad Q \xrightarrow{a^*}}{P +^p Q \xrightarrow{a^*,p\cdot q} P'}$ **(L)** $\dfrac{P \xrightarrow{a^*,q} P' \quad Q\,! \xrightarrow{a^*}}{P +^p Q \xrightarrow{a^*,q} P'}$

(L) $\dfrac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{\mathbf{GAct}}}{P +^p Q \xrightarrow{a,p\cdot q} P'}$ **(L)** $\dfrac{P \xrightarrow{a,q} P' \quad Q\,! \xrightarrow{\mathbf{GAct}}}{P +^p Q \xrightarrow{a,q} P'}$

Composizione parallela

(L) $\dfrac{P \xrightarrow{a^*,q} P' \quad Q \xrightarrow{a^*}}{P \|^p_S Q \xrightarrow{a^*,p\cdot q} P' \|^p_S Q}$ $a \notin S$ **(L)** $\dfrac{P \xrightarrow{a^*,q} P' \quad Q\,! \xrightarrow{a^*}}{P \|^p_S Q \xrightarrow{a^*,q} P' \|^p_S Q}$ $a \notin S$

(L) $\dfrac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,p\cdot q/\nu_P(G_{S,Q})} P' \|^p_S Q}$ $a \notin S$

(L) $\dfrac{P \xrightarrow{a,q} P' \quad Q\,! \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,q/\nu_P(G_{S,Q})} P' \|^p_S Q}$ $a \notin S$

(L) $\dfrac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a^*,q'} Q' \quad Q \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,p\cdot q\cdot q'/\nu_P(G_{S,Q})} P' \|^p_S Q'}$ $a \in S$

(L) $\dfrac{P \xrightarrow{a,q} P' \quad Q \xrightarrow{a^*,q'} Q' \quad Q\,! \xrightarrow{G_{S,P}}}{P \|^p_S Q \xrightarrow{a,q\cdot q'/\nu_P(G_{S,Q})} P' \|^p_S Q'}$ $a \in S$

Restrizione

$$\frac{P \xrightarrow{a^*,q} P'}{P \setminus L \xrightarrow{a^*,q} P' \setminus L} \quad a \notin L \qquad \frac{P \xrightarrow{a,q} P'}{P \setminus L \xrightarrow{a,q/\nu_P(G_L)} P' \setminus L} \quad a \notin L$$

Mascheramento

$$\frac{P \xrightarrow{a^*,q} P' \quad P \xrightarrow{\text{GAct}}}{P/_a^p \xrightarrow{\tau,p\cdot q} P'/_a^p} \qquad \frac{P \xrightarrow{a^*,q} P' \quad P\,! \xrightarrow{\text{GAct}}}{P/_a^p \xrightarrow{\tau,q} P'/_a^p}$$

$$\frac{P \xrightarrow{b^*,q} P'}{P/_a^p \xrightarrow{b^*,q} P'/_a^p} \; a \neq b \qquad \frac{P \xrightarrow{b,q} P' \quad P \xrightarrow{a^*}}{P/_a^p \xrightarrow{b,(1-p)q} P'/_a^p} \; a \neq b$$

$$\frac{P \xrightarrow{b,q} P' \quad P\,! \xrightarrow{a^*}}{P/_a^p \xrightarrow{b,q} P'/_a^p} \; a \neq b$$

$$\frac{P \xrightarrow{a,q} P' \quad P \xrightarrow{a^*}}{P/_a^p \xrightarrow{\tau,(1-p)q} P'/_a^p} \qquad \frac{P \xrightarrow{a,q} P' \quad P\,! \xrightarrow{a^*}}{P/_a^p \xrightarrow{\tau,q} P'/_a^p}$$

Costante

$$\frac{P[M_1/x_1,\ldots,M_n/x_n] \xrightarrow{\pi,p} P'}{A(M_1,\ldots,M_n) \xrightarrow{\pi,p} P'} \qquad \text{se } A(x_1,\ldots,x_n) \triangleq P$$

Controllo equivalenza espressioni

$$\frac{M \approx N \quad P \xrightarrow{\pi,p} P'}{[M \approx N]P \prec \mathrm{Q} \xrightarrow{\pi,p} P'} \qquad \frac{M \not\approx N \quad Q \xrightarrow{\pi,p} Q'}{[M \approx N]P \prec \mathrm{Q} \xrightarrow{\pi,p} Q'}$$

Derivazione attraverso regola

$$\frac{M^* \mapsto_{rule}^{p} N}{[M^* \mapsto_{rule} x]P \prec Q \xrightarrow{\tau,p} P[N/x]}$$

$$\frac{M^* \mapsto_{rule}^{p} N}{[M^* \mapsto_{rule} x]P \prec Q \xrightarrow{\tau,(1-p)} Q}, \; p < 1$$

$$\frac{\nexists N : M^* \mapsto_{rule}^{p'} N}{[M^* \mapsto_{rule} x]P \prec Q \xrightarrow{\tau,1} Q}$$

Teorema. Se P è un processo di $\mathcal{G}$ allora la semantica operazionale di P è un *GRTS* [14, 15].

5.1.3 Alcuni esempi

Esempio 5.1. Per aumentare la familiarità dei lettori con il linguaggio appena descritto (ed in particolare con gli operatori probabilistici e crittografici) mostriamo come viene specificato un sistema composto dai due processi A e B che comunicano tra loro attraverso l'azione $channel \in \mathbf{AType}$ definiti nella seguente maniera (assumiamo che $Exp(channel) = Exp(c_1) = Exp(c_2) = \mathbf{Exp}$):

$A(\{M_A\}_{K_A}) \;||^{p}_{\{channel\}}\; B(K_A)$ in cui:

$$A(M) \triangleq channel(M).\, A(M) +^{r} \tau.\, A(M)$$

$$B(K) \triangleq$$

$$(channel^{*}(x).[<x,K> \mapsto^{1}_{dec} z]c_1(z).B(K) +^{r'} channel^{*}(x).[<x,K> \mapsto^{1}_{dec} z]c_2(z).B(K))$$
$$+^{q} \tau.B(K)$$

L'operatore probabilistico di composizione parallela $||^{p}_{\{channel\}}$ definisce l'insieme dei componenti del sistema concorrente (A e B), la loro interfaccia di comunicazione ($\{channel\}$) e la loro velocità di avanzamento attraverso il parametro probabilistico (p). L'interfaccia di comunicazione $\{channel\}$ indica che i due processi possono interagire eseguendo in sincronia azioni del tipo *channel*. Ogni altra azione locale è eseguita in maniera asincrona dai due processi. La probabilità p rappresenta il parametro probabilistico con cui uno scheduler, ad ogni stato del sistema, decide quale dei due processi debba essere eseguito (nel nostro caso viene eseguito il processo A con probabilità p e il processo B con probabilità $1\text{-}p$).

Il processo A definito sopra invia al processo B l'espressione M_A cifrata con la chiave K_A attraverso l'azione generativa $channel(x)$. La scelta probabilistica indotta dalla composizione alternativa $+^{r}$ indica che il processo A può alternativamente inviare l'espressione a B tramite l'azione *channel*, con probabilità r, oppure restare inattivo attraverso

un'azione interna τ, con probabilità 1-r, per poi continuare a comportarsi come il processo A. Dal momento che le azioni *channel* e τ sono generative, in accordo con il modello probabilistico generativo [13], il processo A decide autonomamente, in base alla distribuzione di probabilità associata alle transizioni generative che è in grado di eseguire, quale azione eseguire e come comportarsi successivamente (si veda il *GRTS* del processo A in fig. 5.2).

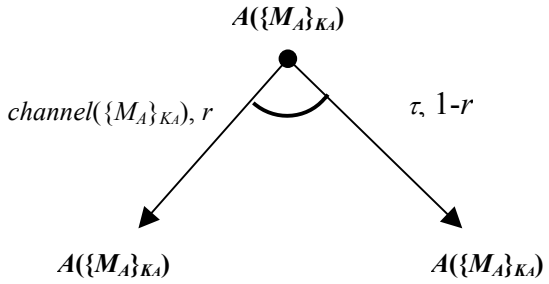


Fig. 5.2. *GRTS* del sistema $A(\{M_A\}_{KA})$.

Il processo B invece riceve l'espressione da decifrare o resta inattivo tramite un'azione interna τ. Le due azioni $channel^*(x)$ permettono di sincronizzarsi con un'azione $channel(x)$ eseguita da un processo esterno attraverso l'operatore di composizione parallela. Tale interazione è guidata dal processo esterno (che esegue un'azione generativa) e quindi puramente non-deterministica, tuttavia quando avviene la sincronizzazione il processo B sceglie in maniera probabilistica quale delle due azioni $channel^*(x)$ eseguire, e quindi in base alla probabilità r' decide attraverso quale delle due azioni c_1, $c_2 \in$ **AType** inviare l'espressione presa in input dopo averla decifrata. Dal momento che le due azioni $channel^*(x)$ sono di tipo reattivo, in accordo con il modello probabilistico reattivo [13], il processo B reagisce internamente all'azione esterna di tipo $channel(x)$ (eseguita dall'ambiente in cui si trova) sulla base della distribuzione di probabilità associata alle azioni reattive del tipo *channel* che è in grado di eseguire. Come abbiamo detto

il processo *B* resta inattivo, eseguendo un'azione interna τ, quando non riceve l'espressione da *A*. La scelta tra le azioni reattive $channel^*(x)$ e l'azione generativa interna τ è quindi puramente non-deterministica e di conseguenza il parametro probabilistico *q* non viene preso in considerazione (si veda il *GRTS* del processo *B* in fig. 5.3).

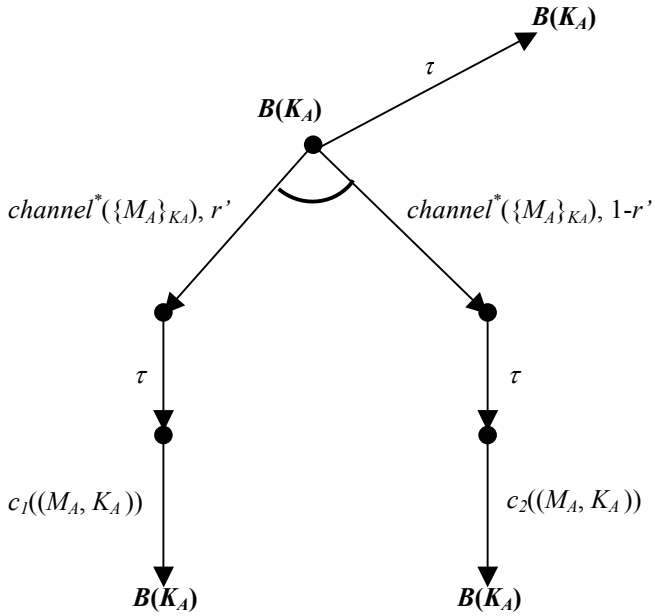


Fig. 5.3. *GRTS* del sistema $B(K_A)$.

Ogni volta che il processo *A* esegue un'azione generativa *channel*(*x*) il processo *B* esegue una sua azione reattiva $channel^*(x)$. Se partiamo dallo stato iniziale dell'esempio appena definito (si veda il *GRTS* del sistema composto in fig. 5.4) il sistema esegue una transizione del processo *A* con probabilità *p*; a questo punto il processo *A* può eseguire l'azione interna τ con probabilità $p\cdot(1\text{-}r)$ o l'azione *channel*(*x*) con probabilità $p\cdot r$ (con probabilità $p\cdot r\cdot r'$ esegue un'azione *channel*(*x*) sincronizzandosi con la prima azione reattiva $channel^*(x)$ del processo *B*, con probabilità $p\cdot r\cdot(1\text{-}r')$ esegue un'azione *channel*(*x*) sincronizzandosi con la seconda azione reattiva $channel^*(x)$ del processo *B*). Viceversa, con probabilità 1-*p*, il sistema può

schedulare il processo B che quindi può eseguire solamente l'azione generativa interna τ.

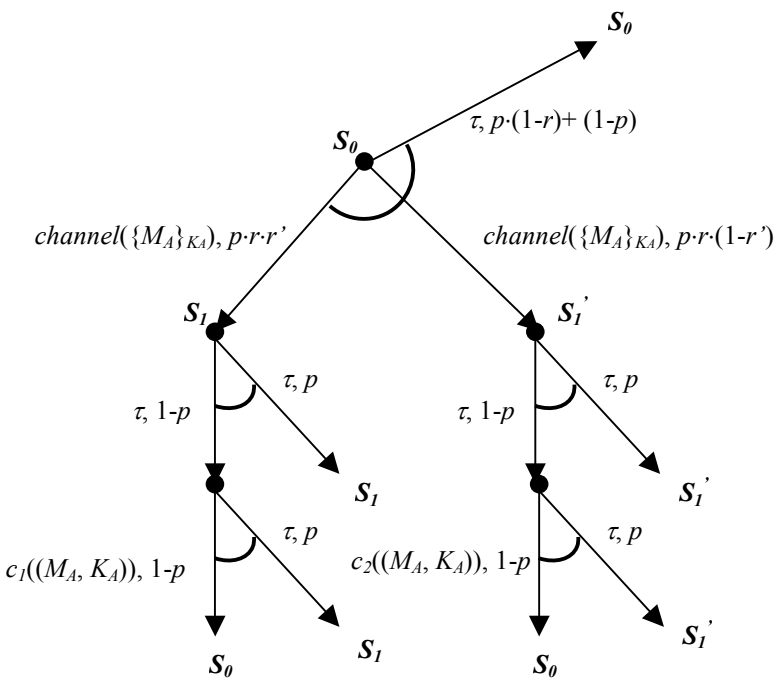


Fig. 5.4. *GRTS* del sistema ottenuto attraverso la composizione parallela dei sistemi *A* e *B*.

Esempio 5.2. Mostriamo ora, attraverso un semplice esempio, come sia possibile individuare un flusso di informazione legato alla debolezza dell'algoritmo di cifratura. Si consideri un sistema A che comunica un'espressione cifrata ad un processo di alto livello Π che non conosce la chiave di cifratura:

$$A(M_A, K_A) \;\|^{p}_{\{h\}}\; \Pi(0), \qquad M_A \not\approx 0 \qquad \text{in cui:}$$

$$A(M, K) \triangleq [<M,K> \mapsto^{1}_{enc} z]\; h(z).Wait(\,(M, K)\,)$$

$$Wait(M) \triangleq \tau.Wait(M) + h^{*}(x).[M \approx x]\; l \prec \underline{0}$$

$$\Pi(M) \triangleq h^{*}(x).\; [<x,M> \mapsto^{p'}_{break} z]\; h(z) \prec h(M).\underline{0}$$

in cui assumiamo $Exp(h)$ = **Exp**. Il processo Π non conosce nessuna

chiave, ma solo il bit 0. Il sistema A comunica, attraverso l'azione di alto livello h, l'espressione cifrata $\{M_A\}_{K_A}$ al processo Π che sincronizza con A attraverso l'azione h^*. A questo punto Π tenta di mettere in chiaro l'espressione M_A rompendo il blocco $\{M_A\}_{K_A}$. Se riesce nel tentativo invia ad A l'espressione (M_A, K_A) altrimenti invia ad A il bit 0. Mentre attende la risposta dal processo Π il sistema A esegue ciclicamente alcune azioni τ. Dopo aver ricevuto la risposta da Π il sistema A controlla che l'espressione ricevuta sia equivalente all'espressione (M_A, K_A). Se questo è il caso viene eseguita l'azione l che altera il comportamento osservabile dagli utenti di basso livello. Come abbiamo mostrato nella sezione **3.3.4**, possiamo valutare la probabilità che si verifichi tale flusso di informazione (questa sarà equivalente alla probabilità del processo Π di rompere il blocco $\{M_A\}_{K_A}$).

Dato il pattern probabilistico $((\{M_A\}_{K_A},0)$, **pKeys**, $pGuess$, $pExp$, $p_{max})$, e fissato $q = pGuess(\{K_A\})$ avremo quindi che:

$A(M_A, K_A) \setminus \mathbf{AType_H} \approx_{PB} \tau.\underline{0}$

$((A(M_A, K_A) \|^p_{\{h\}} \Pi(0)\ ) /_h) \setminus \mathbf{AType_H} \approx_{PB} \tau.(l +^q \underline{0})$

e di conseguenza:

$A(M_A, K_A) \setminus \mathbf{AType_H} \approx_{PBq} ((A(M_A, K_A) \|^p_{\{h\}} \Pi(0)\ ) /_h) \setminus \mathbf{AType_H}$.

Questo semplice esempio mostra come il modello probabilistico sia particolarmente adatto ad essere esteso con operatori crittografici, e a modellare situazioni in cui non si assume una crittografia perfetta.

5.2 Proprietà di sicurezza di sistemi crittografici

Se immaginiamo l'insieme **AType** come l'insieme dei canali[11] attraverso i quali può avvenire la comunicazione tra processi, è opportuno definire un

[11] Da qui in avanti useremo indifferentemente i termini *canale* ed *azione* per indicare tipi di azioni.

sottoinsieme $C \subseteq \mathbf{AType}$ che rappresenti l'insieme dei canali pubblici. L'insieme C contiene, in particolare, tutti i canali insicuri della rete attraverso i quali un nemico può monitorare, intercettare e falsificare messaggi (espressioni), e costituisce l'insieme dei tipi delle azioni che analizzeremo per lo studio di proprietà di sicurezza di sistemi crittografici.

Per poter analizzare le proprietà di sicurezza di un sistema crittografico è inoltre fondamentale caratterizzare tutti i possibili processi ostili che possono interagire con il sistema.

Esempio 5.3. Si consideri un semplice protocollo in cui un processo A invia un'espressione M_A ad un processo B cifrata con la chiave K_{AB} condivisa dai due processi.

$$P \triangleq A(M_A, K_{AB}) \parallel_{\{c\}} B(K_{AB})$$

$$A(M, K) \triangleq [<M,K> \mapsto^{1}_{enc} x]\ c(x)$$

$$B(K) \triangleq c^{*}(y).[<y,K> \mapsto^{1}_{dec} z]\ out(z)$$

in cui $Exp(c) = \{ \{M\}_K \mid M \in \mathbf{Exp}, K \in \mathbf{Keys} \}$. Il protocollo P, dopo la comunicazione tra i due partecipanti, esegue l'azione $out(M_A)$ con probabilità 1, determinando la trasmissione corretta dell'espressione M_A da A a B. In particolare abbiamo che $A(M_A, K_{AB}) \xrightarrow{c(\{M_A\}_{K_{AB}}),1} \underline{0}$, $B(K_{AB})$ è quindi forzato a sincronizzare eseguendo la transizione:

$B(K_{AB}) \xrightarrow{c^{*}(\{M_A\}_{K_{AB}}),1} [<\{M_A\}_{K_{AB}}, K_{AB}> \mapsto^{1}_{dec} z]\ out(z)$. Se mascheriamo la comunicazione interna tra i due partecipanti otteniamo:

$P\ /_c \xrightarrow{\tau,1} (\ \underline{0}\ \parallel_{\{c\}}\ [<\{M_A\}_{K_{AB}}, K_{AB}> \mapsto^{1}_{dec} z]\ \ out(z)\ )\ /_c \xrightarrow{\tau,1}$

$(\underline{0} \parallel_{\{c\}} out(M_A)) \xrightarrow{out(M_A),1} (\underline{0} \parallel_{\{c\}} \underline{0}) \setminus \{c\}$.

Dal momento che la chiave K_{AB} è conosciuta solamente dai processi A e B, il protocollo P dovrebbe poter garantire l'autenticità dell'espressione M_A anche nella presenza di un processo ostile. Assumiamo quindi che $c \in C$ sia un canale pubblico e consideriamo il seguente processo:

$$\Pi(M, K) \triangleq [<M,K> \mapsto^{1}_{enc} x]\ c(x)$$

che può comunicare solamente attraverso canali pubblici dal momento

che $sort(\Pi(M, K)) = \{c\}$. Consideriamo poi, in particolare, il processo $\Pi(M_E, K_{AB})$ che conosce la chiave K_{AB} e può quindi inviare a B un messaggio falsato $\{M_E\}_{K_{AB}}$. Per osservare questo comportamento consideriamo "sotto attacco" il protocollo P e modelliamo la comunicazione tra B e il processo Π:

$$(B \|_{\{c\}} \Pi(M_E, K_{AB})) /_c$$

Il protocollo sotto attacco (modellato componendo il partecipante B con il processo ostile Π) esegue con probabilità 1 la traccia $\tau.\tau.\tau.out(M_E)$. Il partecipante B, che ha ricevuto l'espressione falsata M_E, potrebbe credere che tale espressione gli sia stata inviata da A, dal momento che era cifrata con la chiave K_{AB} che solo loro due dovrebbero condividere. Una situazione di questo tipo si verifica poiché si è permesso di modellare un processo ostile che conosca la chiave K_{AB}, mentre questa dovrebbe essere conosciuta solamente dai processi A e B.

Il problema evidenziato dall'esempio 5.3 viene risolto, come abbiamo già accennato, attraverso una caratterizzazione di tutti i possibili processi ostili. In particolare porremo dei vincoli sulla conoscenza iniziale che può essere accordata ad un nemico.

Dato un processo P, chiamiamo $ID(P)$ l'espressione che si ottiene accoppiando in sequenza tutte le espressioni che appaiono sintatticamente in P. In particolare $ID(P)$ è l'espressione che modella la conoscenza del processo P. Formalmente definiamo $ID(P)$ come la funzione composta $pairExp(I(P,\varnothing))$ in cui $I : \mathcal{G} \times \mathcal{P}(\mathbf{Const}) \rightarrow \mathcal{P}(\mathbf{Exp})$, e $pairExp$: $\mathcal{P}(\mathbf{Exp}) \rightarrow \mathbf{Exp}$, in cui la funzione $pairExp$ è formalizzata nella tabella 5.1, mentre la funzione I è definita formalmente nella tabella 5.2.

$pairExp(\{M\})$	=	M
$pairExp(\{M_1, \ldots, M_n\})$	=	$((\ldots((M_1, M_2), \ldots M_{n-1}), M_n)$

Tab. 5.1. Definizione formale della funzione *pairExp*.

La funzione *pairExp* non fa altro che accoppiare in successione tutte le espressioni presenti nell'insieme che gli viene passato come input.

$$I(\underline{0}, V) = (0, 1)$$

$$I(\pi(x).P, V) = I(P, V)$$

$$I(\pi^*(e).P, V) = getExp(e) \cup I(P,V)$$

$$I(\tau.P, V) = I(P, V)$$

$$I(P +^p Q, V) = I(P, V) \cup I(Q, V)$$

$$I(P \|^p_S Q, V) = I(P, V) \cup I(Q, V)$$

$$I(P \setminus L, V) = I(P, V)$$

$$I(P /^p_a, V) = I(P, V)$$

$$I(A(e_1, \ldots, e_n), V) = \begin{cases} \bigcup_{i=1}^{n} getExp(e_i) & se\ A \in V \\ \bigcup_{i=1}^{n} getExp(e_i) \cup I(P, V \cup \{A\}) & se\ A \notin V \end{cases}$$

in cui $A(x_1, \ldots, x_n) \triangleq P$

$$I([e \approx e'] P \prec Q) = getExp(e) \cup getExp(e') \cup I(P, V) \cup I(Q, V))$$

$$I([e^* \mapsto^{p'}_{rule} x]P \prec Q) = \begin{cases} getExp(e) \cup I(P,V) \cup I(Q,V) & se\ e^* = e \\ getExp(e) \cup getExp(e') \cup I(P,V) \cup I(Q,V)\ se\ e^* = <e, e'> \end{cases}$$

in cui:

$$getExp(e) = \begin{cases} \{e\} & se\ e \in \mathbf{Exp} \\ \{\} & se\ e \in \mathbf{Var} \end{cases}$$

Tab. 5.2. Definizione formale della funzione *I*(*P*, *V*).

Intuitivamente $I(P, V)$ è una funzione che ricorsivamente visita i sotto-termini di P e il corpo dei termini costanti richiamati in P. L'argomento V è usato per controllare che l'analisi di un termine costante venga effettuata una volta sola. Sottolineiamo che il processo inattivo $\underline{0}$ non fornisce alcuna conoscenza utile, di conseguenza $I(\underline{0}, V) = (0, 1)$ indica che dal processo $\underline{0}$ si ricavano solo i bit 0 e 1.

Esempio 5.4. Si consideri il processo $A(M_1)$ in cui:

$$A(x) \triangleq P = c(x).\underline{0} \parallel_{\{c\}} c(M_2).A(M_3)$$

Avremo che:

$I(A(M_3), \{A\}) = \{M_3\}$

$I(c(x).\underline{0}, \{A\}) = I(\underline{0}, \{A\}) = \{(0, 1)\}$

$I(c(M_2).A(M_3), \{A\}) = \{M_2\} \cup I(A(M_3), \{A\})) = \{M_2, M_3\}$

$I(P, \{A\}) = I(c(x).\underline{0}, \{A\}) \cup I(c(M_2).A(M_3), \{A\}) = \{(0,1), M_2, M_3\}$

$I(A(M_1), \varnothing) = \{M_1\} \cup I(P, \{A\}) = \{M_1, (0, 1), M_2, M_3\}$

Dunque $ID(A(M_1)) = pairExp(I(A(M_1), \varnothing)) = (((M_1, (0, 1)), M_2), M_3)$.

Sia quindi $\Delta \in \mathbf{Exp}$ l'espressione che modella la conoscenza iniziale che si vuole accordare ai processi ostili; Δ conterrà ad esempio, oltre alle chiavi private dei processi ostili, le chiavi e le informazioni rese pubbliche. Dato un processo ostile Π vogliamo quindi che la sua conoscenza, rappresentata da $ID(\Pi)$, sia derivabile (attraverso la relazione di entailment mostrata nella sezione **4.2**) dall'espressione Δ.

In questa prima analisi, definiamo quindi l'insieme di tutti i possibili nemici come $\mathcal{G}_{C,\Delta} = \{ \Pi \in \mathcal{G}^w \mid sort(\Pi) \subseteq C \wedge \Delta \mapsto ID(\Pi) \}$, in cui, attraverso considerazioni identiche a quelle effettuate per la definizione 3.4, $\mathcal{G}^w = \{ P \in \mathcal{G} \mid P$ è *ben definito* $\}$ rappresenta l'insieme dei processi di *CryptoPPA* ben definiti.

La definizione dell'insieme di tutti i possibili nemici $\mathcal{G}_{C,\Delta}$ risolve il problema presentato nell'esempio 5.3 semplicemente ponendo la condizione $\Delta \mapsto K_{AB}$, di conseguenza $\Delta \mapsto ID(\Pi(M_E, K_{AB})) = (M_E, K_{AB})$ e il processo $\Pi(M_E, K_{AB}) \notin \mathcal{G}_{C,\Delta}$.

5.2.1 Crypto Probabilistic Non-deducibility on Composition

La nozione della non-interference [37] può essere applicata anche per la verifica di varie proprietà di sicurezza di protocolli crittografici [60, 61], tra le quali troviamo anche la proprietà di autenticazione. La correttezza di un

protocollo può infatti essere dimostrata garantendo che un nemico non sia in grado di interferire con l'esecuzione del protocollo. In particolare, è sufficiente verificare che l'esecuzione del protocollo isolato sia uguale all'esecuzione del protocollo sotto l'attacco di qualsiasi possibile nemico.

Definiamo quindi la proprietà della *NDC* nel contesto crittografico probabilistico. L'idea è la seguente: il gruppo U di utenti disonesti, che non deve interferire con gli altri utenti onesti, è caratterizzato da un insieme di azioni che i suoi componenti possono eseguire. Dato l'insieme C dei canali (azioni) che i processi dell'insieme U possono utilizzare come mezzo di comunicazione, un processo P è *CryptoProbabilisticBNDC* (*CryptoPBNDC*), se ogni utente disonesto $X \in U$ composto al sistema P non è in grado di modificare il comportamento di P osservabile dagli altri utenti onesti.

Dal momento che la composizione parallela *CSP*-like adottata da *CryptoPPA* non permette ai processi di eseguire localmente azioni il cui tipo è all'interno dell'insieme di sincronizzazione (in particolare non possono interagire con l'esterno attraverso azioni di quel tipo), non si riesce a modellare la composizione tra il protocollo ed un processo ostile. Superiamo questo ostacolo descrivendo le comunicazioni che avvengono su canali pubblici attraverso la composizione tra il protocollo ed un processo che modella il comportamento dei canali pubblici coinvolti. Ad esempio ogni azione generativa c eseguita da un partecipante su un canale pubblico verrà rimodellata con un'azione generativa c_s sul canale pubblico. Il processo che modella il canale effettuerà quindi un'azione reattiva c_s^*, ricevendo l'espressione, e quindi attraverso un'azione generativa c_t smista l'espressione al partecipante che la riceverà attraverso un'azione reattiva c_t*****.

Sia quindi $\mathbf{AType_C} = \{ c_s \mid c \in C \} \cup \{ c_t \mid c \in C \}$, inoltre sarà $Exp(c_s) = Exp(c_t) = Exp(c)$. Chiamiamo $\mathbf{AType_P} = \{c \in \mathbf{AType} - \{\tau\} \mid c \notin C\}$ l'insieme dei canali sicuri e ridefiniamo l'insieme $\mathbf{AType} = \mathbf{AType_P} \cup \mathbf{AType_C} \cup \{\tau\}$.

Sia poi $\mathcal{G}^C=\{P\in\mathcal{G}^w \mid sort(P)\subseteq\mathbf{AType_C}\}$ l'insieme dei possibili processi che modellano le comunicazioni su canali pubblici.

Allo scopo di semplificare l'analisi per la definizione delle proprietà di sicurezza di sistemi crittografici assumiamo di trattare solamente protocolli con due partecipanti[12]. In particolare, assumiamo che ogni protocollo sia rappresentato nella forma $P = (A \|_S^p B) \|_{\mathbf{AType_C}}^q \boldsymbol{C}$, in cui $S\subseteq\mathbf{AType_P}$ è un insieme di canali privati su cui A e B possono comunicare e $\boldsymbol{C}\in\mathcal{G}^C$ è il processo che regola, onestamente, la comunicazione tra A e B su canali pubblici. Definiamo l'insieme dei protocolli che prenderemo in analisi come:

$\mathcal{G}^{Pw} = \{P = (A\|_S^p B)\|_{\mathbf{AType_C}}^q \boldsymbol{C} \mid A,B\in\mathcal{G}^w, \boldsymbol{C}\in\mathcal{G}^C, S\subseteq\mathbf{AType_P}, p, q\in]0,1[\ \}$.

Un semplice esempio di un protocollo $P\in\mathcal{G}^{Pw}$ potrebbe essere il seguente:

$$P = (\ A\ \|_S^p\ B\ )\ \|_{\mathbf{AType_C}}^q\ \boldsymbol{C} \text{ in cui:}$$

$S = \varnothing$ (i partecipanti comunicano solamente attraverso canali pubblici)

$p = q = ½$ (ma in questo caso sono ininfluenti)

$$A = c_s(M) +^p c'_s(M)$$

$$B = c_t^*(x).out(x) + c'^*_t(x).out'(x)$$

$$\boldsymbol{C} = c_s^*(x).c_t(x) + c'^*_s(x).c'_t(x)$$

$c_s, c_t, c'_s, c'_t\in\mathbf{AType_C}$, $out, out'\in\mathbf{AType_P}$.

In cui il partecipante A decide probabilisticamente se inviare l'espressione M a B attraverso il canale c o il canale c'. Il processo $\boldsymbol{C}$, come abbiamo visto, gestisce in maniera sicura la comunicazione che avviene sui canali pubblici. B riceve invece l'espressione M e la smista sul canale privato *out* o sul canale privato *out'* in base al canale da cui la ha ricevuta.

[12] Ciò non costituisce una reale limitazione poiché è sempre possibile trattare protocolli con più partecipanti raggruppandoli per coppie.

A questo punto possiamo descrivere l'interazione del protocollo con potenziali processi ostili modificando il comportamento del processo $\boldsymbol{C}$ in considerazione che eventuali nemici siano in grado di monitorare i canali pubblici e quindi intercettare e smistare messaggi. Chiamiamo dunque $\Pi(\boldsymbol{C})$ un generico processo che modella lo scambio di messaggi su canali pubblici con eventuali interazioni ostili. In particolare $\Pi(\boldsymbol{C})$ dovrà simulare tutte le comunicazioni che venivano gestite da $\boldsymbol{C}$.

Definiamo quindi la relazione di simulazione come segue:

Definizione 5.1. ***C**-simulazione*

Una relazione $\mathcal{R} \subseteq \mathcal{G}^C \times \mathcal{G}^C$ è una ***C**-simulazione* $\Leftrightarrow$

$\forall\ (P, Q) \in \mathcal{R}$ e $\forall \pi : t(\pi) \in \mathbf{AType_C}$:

- $P \xrightarrow{\pi} P'$, $\pi \in \mathbf{GAct} \Rightarrow \exists\ Q'$: $Prob(Q, \tau^* \hat{\pi}, Q') > 0 \wedge (P', Q') \in \mathcal{R}$
- $P \xrightarrow{\pi^*} P'$, $\pi^* \in \mathbf{RAct} \Rightarrow \exists\ Q'$: $Prob(Q, \pi^*, Q') > 0 \wedge (P', Q') \in \mathcal{R}$

Diremo quindi che un processo $\Pi(\boldsymbol{C})$ ***C**-simula* il processo $\boldsymbol{C}$ (con notazione $\Pi(\boldsymbol{C}) \sim \boldsymbol{C}$), se esiste una relazione $\mathcal{R}$ di ***C**-simulazione* che contiene la coppia $(\boldsymbol{C}, \Pi(\boldsymbol{C}))$.

La proprietà *CryptoPBNDC* può quindi essere definita formalmente nella seguente maniera:

Definizione 5.2. *CryptoPBNDC*

$P \in \mathcal{G}^{Pw} \in CryptoPBNDC, \quad P = (A \|_S^p B) \|^q_{\mathbf{AType_C}} \boldsymbol{C} \Leftrightarrow$

$$((A \|_S^p B) \|^q_{\mathbf{AType_C}} \boldsymbol{C}) \,/\, \mathbf{AType_C} \approx_{PB} ((A \|_S^p B) \|^q_{\mathbf{AType_C}} \Pi(\boldsymbol{C})) \,/\, \mathbf{AType_C}$$

$\forall\ \Pi(\boldsymbol{C}) \in \mathcal{G}^C :\ \Pi(\boldsymbol{C}) \sim \boldsymbol{C} \ \wedge \ \Delta \mapsto ID(\Pi(\boldsymbol{C}))$

in cui $\approx_{PB}$ è la bisimulazione debole probabilistica in accordo alla definizione 3.3.

La proprietà *CriptoPBNDC* richiede quindi che le interazioni del protocollo con tutti i potenziali processi ostili modellati da $\Pi(\boldsymbol{C})$, non siano in alcun modo in grado di alterare il comportamento del sistema, osservabile dagli utenti onesti, rappresentato dal termine $(A \|_S^p B) \|^q_{\mathbf{AType_C}} \boldsymbol{C}$. In particolare, $(A \|_S^p B) \|^q_{\mathbf{AType_C}} \boldsymbol{C}$ rappresenta il protocollo P in cui non è permessa alcuna attività disonesta (le comunicazioni attraverso i canali pubblici sono regolate dal processo onesto $\boldsymbol{C}$). Se tale sistema è equivalente per bisimulazione debole probabilistica al termine $(A \|_S^p B) \|^q_{\mathbf{AType_C}} \Pi(\boldsymbol{C})$, significa che il processo $\Pi(\boldsymbol{C})$ non è in grado di alterare in alcun modo osservabile il comportamento del sistema.

Una definizione di questo tipo ha, tuttavia, senso solamente data l'assunzione di crittografia perfetta. Dal momento che nel nostro modello abbiamo deciso di abbandonare questa assunzione, è veramente difficile definire protocolli che siano sicuri in accordo alla definizione 5.1. Infatti, se si modella la probabilità (anche estremamente bassa) per un nemico di portare a termine con successo un attacco all'algoritmo di cifratura, difficilmente un protocollo composto con un processo ostile potrà essere equivalente per bisimulazione debole probabilistica al protocollo sicuro (ottenuto modellando in maniera sicura le comunicazioni sui canali pubblici).

Esempio 5.5. Si consideri di nuovo il protocollo dell'esempio 5.3, riscritto nella formula $(A \|_S^p B) \|^q_{\mathbf{AType_C}} C$:

$$P = (A(M_A, K_{AB}) \| B(K_{AB})) \|^q_{\mathbf{AType_C}} \boldsymbol{C}$$

$$A(M, K) = [<M,K> \mapsto^1_{enc} x]\ c_s(x)$$

$$B(K) = c_t^*(y).[<y,K> \mapsto^{p'}_{dec} z]\ out(z)$$

$$\boldsymbol{C} = c_s^*(x).c_t(x)$$

Assumendo che $\Delta \mapsto\!\!\!\!\!\rightarrow K_{AB}$ limitiamo la conoscenza di ogni possibile processo ostile. Il protocollo è sicuro solo con l'assunzione di crittografia perfetta. Si consideri ad esempio il processo ostile:

$$\Pi(\boldsymbol{C})(M_E,\ \{M_A\}_{KAB}) = c_s^*(x).[<\{M_A\}_{KAB},(M_E,\ \{M_A\}_{KAB})> \mapsto^{p'}_{break} t]$$

$$(\ [t \mapsto^1_{snd} v]\ ([<M_E, v> \mapsto^1_{enc} z]\ c_t(z)\ )\ ) \prec c_t(x)$$

in cui assumiamo che $ID(\Pi(\boldsymbol{C})) \mapsto \{M_A\}_{KAB}$ (l'assunzione ha senso se immaginiamo che il processo $\Pi(\boldsymbol{C})$ abbia catturato l'espressione $\{M_A\}_{KAB}$ durante un'esecuzione precedente del protocollo). Il processo $\Pi(\boldsymbol{C})$ cerca quindi di decifrare l'espressione $\{M_A\}_{KAB}$, e se ci riesce utilizza la chiave K_{AB} per cifrare l'espressione M_E da inviare a B. In caso contrario si comporta come il processo che modella comunicazioni sicure $\boldsymbol{C}$, inviando a B l'espressione $\{M_A\}_{KAB}$. In particolare, dato il pattern probabilistico $((\{M_A\}_{KAB}, M_E)$, **pKeys**, *pGuess*, *pExp*, $p_{max})$ la probabilità che il processo $\Pi(\boldsymbol{C})$ decifri il blocco $\{M_A\}_{KAB}$, è $p' = pGuess(\{K_{AB}\})$, per cui:

$(\ (A(M_A, K_{AB}) \| B(K_{AB})) \|^q_{\mathbf{AType_C}} \boldsymbol{C}\ )$ / $\mathbf{AType_C} \approx_{PB} \tau.out(M_A) \not\approx_{PB} \tau.out(M_E) +^{p'}$

$\tau.out(M_A) \approx_{PB} (A(M_A, K_{AB}) \| B(K_{AB})) \|^q_{\mathbf{AType_C}} \Pi(\boldsymbol{C}))$ / $\mathbf{AType_C}$.

Per cui il sistema non può essere considerato sicuro. Tuttavia se utilizziamo l'equivalenza quantitativa di bisimulazione debole con precisione ε (definizione 3.6) abbiamo che:

$((A \|^p_S B) \|^q_{\mathbf{AType_C}} \boldsymbol{C}\ )$ / $\mathbf{AType_C} \approx_{PB_{p'}} ((A \|^p_S B) \|^q_{\mathbf{AType_C}} \Pi(\boldsymbol{C}))$ / $\mathbf{AType_C}$.

Ridefiniamo quindi la proprietà *CryptoPBNDC* utilizzando l'equivalenza quantitativa della definizione 3.6, che permette di stabilire se due sistemi probabilistici si comportano quasi nello stesso modo (a meno di piccole fluttuazioni).

Come abbiamo visto nella sezione **3.3.4**, il modello probabilistico ci permette di analizzare quanto un sistema è insicuro. Quindi, fissata una soglia ε, definiamo la nozione di ε-*CryptoPBNDC*.

Definizione 5.3. *ε-CryptoPBNDC*

$P \in \mathcal{G}^{Pw} \in \varepsilon\text{-}CryptoPBNDC,\ P = (A \|_S^p B) \|^q_{\mathbf{AType_C}} \boldsymbol{C} \Leftrightarrow$

$((A \|_S^p B) \|^q_{\mathbf{AType_C}} \boldsymbol{C}) / \mathbf{AType_C} \approx_{PB\varepsilon} ((A \|_S^p B) \|^q_{\mathbf{AType_C}} \Pi(\boldsymbol{C})) / \mathbf{AType_C}$

$\forall\ \Pi(\boldsymbol{C}) \in \mathcal{G}^C :\ \Pi(\boldsymbol{C}) \sim \boldsymbol{C} \ \wedge\ \Delta \mapsto ID(\Pi(\boldsymbol{C}))$

Sottolineiamo infine come l'idea di non assumere crittografia perfetta si adatti in maniera naturale al modello probabilistico (si veda ancora l'esempio 5.5).

5.2.2 Proprietà di Autenticazione

In questo paragrafo mostriamo come, fissata una soglia di sicurezza ε, la proprietà *ε-CryptoPBNDC* sia in grado di verificare se un protocollo crittografico soddisfa la proprietà di autenticazione.

In [23] viene proposta un'interessante definizione per la proprietà di autenticazione. Si consideri un protocollo $P(M)$ il cui obiettivo è trasmettere l'espressione M da un utente A ad un utente B. Il protocollo $P(M)$ soddisfa la proprietà di autenticazione per l'espressione M se ogni volta che un'espressione M' viene inviata da $P(M)$ a B l'espressione M' è uguale all'espressione M.

L'idea che sta alla base della verifica di questa proprietà è la seguente: a partire dal protocollo $P(M)$ si genera un protocollo $P_{sec}(M)$ che viene definito in modo da garantire la proprietà di autenticazione ($P_{sec}(M)$ potrebbe, ad esempio, essere costruito in modo che l'unica espressione che possa essere comunicata a B sia M). A questo punto è sufficiente verificare che il protocollo $P(M)$ sia equivalente per bisimulazione debole probabilistica al protocollo $P_{sec}(M)$.

Come abbiamo visto una specifica sicura del protocollo $P(M)$ si può ottenere facilmente modellando in maniera sicura, attraverso il processo $\boldsymbol{C}$, le comunicazioni su canali pubblici.

La proprietà di autenticazione può quindi essere riformulata nel nostro formalismo come:

Definizione 5.4. *ε-Authentication*

Dato il protocollo $P(M_1, \ldots, M_n) \in \mathcal{G}^{Pw}$, P garantisce la proprietà di autenticazione con precisione $\varepsilon \Leftrightarrow P(M_1, \ldots, M_n) \in \varepsilon\text{-}CryptoPBNDC\ \forall M_1,\ldots,M_n \in \mathbf{Exp}$.

Esempio 5.6. Si consideri un semplice protocollo in cui un utente A invia a B un'espressione M in chiaro, attraverso il canale pubblico c:

$$P(x) = (A(x) \| B) \|^{q}_{\mathbf{AType_C}} \boldsymbol{C}$$

$$A(x) = c_s(x)$$

$$B = c_t^*(y).F(y)$$

$$\boldsymbol{C} = c_s^*(x).c_t(x)$$

in cui il processo B, dopo avere ricevuto l'espressione nella variabile y, si comporta come $F(y)$ in cui probabilmente utilizzerà l'espressione ricevuta. Il protocollo, chiaramente, non soddisfa la proprietà di autenticazione. Qualsiasi processo ostile potrebbe infatti comunicare a B attraverso il canale pubblico c un'espressione diversa dall'espressione che il processo A vuole inviargli. Formalmente si consideri un processo ostile:

$$\Pi(\boldsymbol{C}) = c_s^*(x).(c_t(x) +^p c_t(M')), \text{ con } M' \neq M$$

che $\boldsymbol{C}$-simula il processo $\boldsymbol{C}$ ma può anche inviare a B un messaggio diverso da quello inviato da A. Formalmente avremo che:

$((A(M)\|B)\|^{q}_{\mathbf{AType_C}} \boldsymbol{C})$ / $\mathbf{AType_C}$ = $\tau.\tau.F(M) \not\approx_{PB\varepsilon} \tau.\tau.F(M) +^p \tau.\tau.F(M') =$ $((A(M)\|B)\|^{q}_{\mathbf{AType_C}} \Pi(\boldsymbol{C}))$ / $\mathbf{AType_C}$, con $\varepsilon < p$ e con $F(M) \not\approx_{PB\varepsilon} F(M')$. Quest'ultima condizione è sicuramente sensata ed è legata al fatto che B, ricevendo un'espressione diversa da quella che riceverebbe in un protocollo sicuro, ha un comportamento futuro (rappresentato da $F(y)$) alterato. Se tale condizione non valesse, il valore dell'espressione ricevuta non sarebbe rilevante rendendo inutile il controllo sull'autenticazione di M.

Sulla base dell'approccio generale di Focardi, Gorrieri e Martinelli [17,

18], è possibile modellare, attraverso gli strumenti del nostro formalismo, svariate altre proprietà di sicurezza di sistemi crittografici. Infatti, una volta stabilito quale è il comportamento sicuro del protocollo sulla base di una certa proprietà, basterà verificare che tale comportamento sia equivalente al comportamento del protocollo che può ricevere eventuali attacchi.

6 Un caso di studio

In questo capitolo mostreremo come la nozione della non-interference nel contesto probabilistico possa essere utilizzata per catturare una perdita di informazione dovuta al comportamento probabilistico di sistemi crittografici. In particolare, come caso di studio, analizzeremo e modelleremo un protocollo di *non-repudiation* che utilizza un algoritmo probabilistico per soddisfare la proprietà di *fairness*. L'analisi di tale protocollo confermerà che l'approccio non-deterministico per l'individuazione dei flussi di informazione non è sufficiente per lo studio di proprietà di sicurezza di sistemi crittografici.

6.1 Introduzione

Presentiamo un semplice ed efficace caso di studio che mostra come l'approccio non-deterministico per l'analisi dei flussi di informazione non sia sufficiente a derivare risultati significativi. In particolare, analizzeremo un protocollo di non-repudiation basato su un algoritmo probabilistico che permette a due partecipanti di scambiare informazioni in maniera equa senza doversi affidare ad un terzo partecipante onesto. Nel contesto delle proprietà di sicurezza di protocolli crittografici l'idea della fairness può essere riformulata nel seguente modo: durante l'esecuzione di un protocollo, o entrambi i partecipanti ottengono la parte di informazione che gli spetta, o la

probabilità che solo uno dei partecipanti ottenga l'informazione voluta è inferiore ad una soglia fissata ε. Di conseguenza, per valutare il livello di sicurezza del protocollo, dovremo stimare il valore di ε, questo è il parametro più importante per garantire la fairness del sistema. Utilizzando il classico approccio non-deterministico non potremmo derivare una stima quantitativa dei potenziali flussi di informazione indesiderati, è quindi necessario effettuare un'analisi che prenda in considerazione il comportamento probabilistico del sistema.

Nel prossimo paragrafo descriveremo il protocollo probabilistico di non-repudiation. Nell'ultimo descriveremo il protocollo utilizzando la sintassi di *CryptoPPA*. Quindi formalizzeremo la proprietà di non-repudiation seguendo un approccio simile a quello mostrato nel capitolo 5, e verificheremo tale proprietà per l'algoritmo probabilistico.

6.2 Un protocollo probabilistico di non-repudiation

In un protocollo con scambio di messaggi, la repudiation consiste nella negazione, da parte di una delle entità coinvolte nella comunicazione, di aver partecipato al protocollo o ad una parte di esso. In questo contesto, la *non-repudiation dell'origine* serve a prevenire che il mittente di un messaggio neghi di averlo mandato, mentre la *non-repudiation del ricevente* previene che il destinatario di un messaggio neghi di averlo ricevuto. La proprietà di non-repudiation diviene fondamentale, ad esempio, nel commercio elettronico se si vuole garantire la validità di una transazione (se la conferma di un pagamento ricevuto dovesse essere effettuata con un acknowledgement, il cliente potrebbe rifiutarsi di inviare l'ack negando di avere ricevuto il pagamento). Generalmente, per assicurare la proprietà di non-repudiation, viene coinvolto un terzo partecipante onesto che svolgerà il ruolo di garante in caso di dispute future. Il protocollo che noi considereremo garantisce la proprietà di non-repudiation, con una certa probabilità, senza però coinvolgere un terzo partecipante. Il protocollo garantisce che per ogni messaggio inviato dal mittente *O*, che offre un

servizio, sia equamente fornito un acknowledgement inviato dal destinatario *R*, a conferma di aver ricevuto il servizio. In particolare, il protocollo probabilistico è *ε-fair*, ovvero, ad ogni passo di esecuzione del protocollo, o entrambi i partecipanti hanno ricevuto l'informazione aspettata o la probabilità che solo il partecipante disonesto riceva l'informazione desiderata è minore di ε.

Prima di passare alla descrizione dell'algoritmo introduciamo alcune nozioni ed assunzioni. Si supponga che una fase di autenticazione preceda il protocollo di non-repudiation, durante la quale i partecipanti stabiliscono una chiave condivisa da utilizzare per lo scambio dei messaggi. Se *K* è la chiave stabilita indicheremo con $\{M\}_K$ il messaggio[13] *M* cifrato con la chiave *K*. Assumiamo quindi che ogni messaggio tra *R* ed *O* sia cifrato con la chiave condivisa *K*. Sia poi *n* il numero dei passi del protocollo. Il valore di *n* viene deciso dall'origine che, inoltre, calcola *n* funzioni $f_1, \ldots, f_n$ che costituiscono le parti di una funzione composta. In particolare, $f_n(M) \circ f_{n-1}(M) \circ \ldots \circ f_1(M) = M$. Infine, in ogni messaggio scambiato, verrà utilizzato un marcatore temporale *t*.

Una descrizione del protocollo è fornita nella tab. 6.1, in cui la notazione $R \rightarrow O : Msg$ indica che il messaggio *Msg* viene inviato da *R* e ricevuto da *O*. Il ricevente *R* dà inizio al protocollo inviando una richiesta al fornitore del servizio *O*. L'origine *O*, a sua volta, decide autonomamente il numero *n* dei passi del protocollo e quindi invia ad *R* il primo messaggio contenente la funzione $f_n(M)$. Ad ogni messaggio $f_i(M)$ ricevuto, *R* invia un messaggio di acknowledgement $ack_i = (i, R, O, t)$, confermando di aver ricevuto il messaggio $f_i(M)$. Dal momento che il ricevente non conosce il numero dei passi *n* non può determinare quando il protocollo terminerà, e, di conseguenza, neppure quando riceverà il messaggio finale che gli permetterà di calcolare il valore di *M*. In particolare, poiché le funzioni f_i vengono inviate in ordine inverso, assumendo che la loro composizione non

[13] Formalmente un messaggio è rappresentato da un'espressione, in accordo al modello mostrato nel capitolo 4.

sia commutativa, il ricevente non può eseguire computazioni in anticipo tentando di indovinare il messaggio M prima di aver ricevuto l'ultimo messaggio $f_1(M)$. Dopo aver ricevuto l'ack relativo all'ultimo messaggio inviato (contenente $f_1(M)$), il fornitore del servizio O termina in maniera corretta il protocollo. Si noti che non è necessario l'invio di un messaggio da parte di O per indicare la terminazione del protocollo, infatti quando il ricevente R smette di ricevere messaggi può dedurre che il protocollo è terminato e quindi calcolare il valore di M.

1.	$R \rightarrow O : \{request, R, O, t\}_K$
2.	$O \rightarrow R : \{f_n(M), O, R, t\}_K$
3.	$R \rightarrow O : \{ack_1\}_K$
…	…
$2n$	$O \rightarrow R : \{f_1(M), O, R, t\}_K$
$2n+1$	$R \rightarrow O : \{ack_n\}_K$

Tab. 6.1. Protocollo probabilistico di non-repudiation

E' opportuno sottolineare che ogni messaggio contiene un marcatore temporale (t), questo può essere considerato un valore casuale utilizzato per garantire la freschezza del messaggio e per proteggere i partecipanti contro attacchi che utilizzano vecchi messaggi replicati. Intuitivamente, la non-repudiation dell'origine è garantita dai messaggi $\{f_n(M), O, R, t\}_K$ in cui $i \in \{1, \ldots n\}$, mentre la non-repudiation del ricevente è assicurata dall'ultimo acknowledgement $\{ack_n\}_K$. Se il protocollo termina dopo l'invio dell'*n-esimo* ack, entrambi i partecipanti hanno ottenuto l'informazione desiderata ed il protocollo risulta fair. Se il protocollo termina prima dell'invio del messaggio che contiene $f_1(M)$, nessuno dei partecipanti ottiene l'informazione desiderata e la proprietà fair viene preservata. Dal momento che il numero n dei passi del protocollo è conosciuto solamente dall'origine,una possibile strategia per un ricevente disonesto potrebbe essere:

(*i*) indovinare quale sia l'ultimo messaggio che contiene $f_1(M)$ calcolando la composizione delle funzioni f_i ricevute

(*ii*) violare la proprietà di fairness del protocollo bloccando la trasmissione dell'ultimo ack.

Di conseguenza diventa fondamentale, per la sicurezza del protocollo, l'invio immediato degli ack ad ogni $f_i(M)$ ricevuto. Un ricevente disonesto non deve, cioè, avere il tempo di computare la composizione delle funzioni ricevute fino a quel momento allo scopo di capire se ha già ricevuto o meno il messaggio M. Quindi l'origine deve bloccare il protocollo se, dopo l'invio di un messaggio $f_i(M)$, non riceve l'ack del ricevente entro una soglia di tempo fissata. A questo scopo la scelta delle funzioni f_i deve essere tale che il tempo necessario per la computazione della composizione sia superiore al tempo necessario per la trasmissione di un ack. In ogni modo, un ricevente disonesto potrebbe tentare di indovinare a caso quale sia l'ultimo messaggio contenente $f_1(M)$ in base allo spazio e alla distribuzione di probabilità utilizzati dall'origine O per definire il valore di n.

Concludiamo questa sezione mostrando una proposta concreta, basata sull'algoritmo probabilistico di non-repudiation mostrato sopra, che semplifica notevolmente la scelta delle funzioni f_i. Data una funzione di cifratura che utilizza una chiave segreta K_c, poniamo $f_n(M) = \{M\}_{Kc}$, consideriamo quindi che le funzioni $f_{n-1}(M)$, …, $f_2(M)$ contengano messaggi casuali, infine $f_1(M) = K_c$ è la chiave, in chiaro, che permette di ricavare M. In questo modo il ricevente ottiene subito il messaggio M cifrato con una chiave sconosciuta, e prima che gli sia inviato l'ultimo messaggio (contenente la chiave) non riceve alcuna altra informazione utile. Per difendersi da un ricevente disonesto, l'algoritmo di cifratura deve essere scelto in modo che il tempo impiegato per verificare una chiave (tentando di decifrare il messaggio con un generico $f_i(M)$) sia superiore al tempo necessario per trasmettere un ack.

6.3 Analisi della sicurezza del protocollo

In questa sezione modelleremo attraverso l'algebra di processi *CryptoPPA* una proposta concreta del protocollo descritto nel paragrafo precedente con l'obiettivo di verificare formalmente la proprietà di non-repudiation. Astraendo dai ritardi di trasmissione legati al canale, assumiamo che un messaggio che viene ritardato (o non inviato) da un partecipante non verrà ricevuto dall'altro partecipante.

Attraverso la sintassi del linguaggio *CryptoPPA* introduciamo i modelli dell'origine e del ricevente che si comportano in maniera onesta. La specifica algebrica dell'origine è fornita nella tab. 6.2. Il processo O è pronto, in ogni momento, a dare inizio all'invio del messaggio accettando eventuali richieste (sia *Req* l'espressione che modella una richiesta), inviando il primo messaggio contenente l'informazione M cifrata con la chiave segreta K_c, e ricevendo il primo messaggio di acknowledgement ack_1. Quindi, assumendo che il numero n dei passi del protocollo sia scelto nello spazio $\{1, \ldots, \eta\}$ con una distribuzione di probabilità uniforme, nel termine O' viene selezionato il valore di n attraverso una scelta probabilistica tra η azioni generative di tipo c (in cui c è il canale pubblico attraverso il quale avviene la comunicazione tra l'origine ed il ricevente), in cui la probabilità associata ad ogni azione è $1/\eta$. Se l'azione c selezionata attraverso la scelta probabilistica di O' è quella del termine O_n, l'origine invia al ricevente il primo di n messaggi, ed attende il relativo ack. Sia M_0 l'espressione che corrisponde al messaggio $(\{M\}_{Kc}, O, R, t)$; M_i, con $0<i<n$, l'espressione che modella il messaggio (N_i, O, R, t), in cui N_i è un'espressione generica che non contiene informazioni utili; M_n l'espressione corrispondente al messaggio (K_c, O, R, t); infine chiamiamo A_i, con $0 \leq i \leq n$, l'espressione che modella l'informazione dell'i-esimo acknowledgement (i, R, O, t). In particolare modelliamo formalmente la ricezione di un ack attraverso la traccia $c^*(x).[<x, K> \mapsto^1_{dec} z]\ [z \approx A_i]$, mentre l'azione di input $stop^*$ modella la mancata ricezione dell'ack entro la soglia di tempo prestabilita (causando

una terminazione prematura del protocollo). Se il termine O_n riceve l'ack passa nello stato rappresentato da O_{n-1} ed esegue un nuovo passo del protocollo. Dal momento che abbiamo deciso di astrarre dalla modellazione del tempo, la scelta tra l'invio di un ack e l'azione *stop* è puramente non-deterministica e viene lasciata all'ambiente (come vedremo, tale scelta verrà risolta dal ricevente che può decidere di inviare o meno l'ack). La terminazione *fair* del protocollo viene raggiunta quando l'origine riceve l'ultimo ack ed esegue l'azione generativa *stop* (si vedano i termini O_1 e O_{end}). Il protocollo termina invece in maniera non-*fair* solamente se viene eseguita l'azione reattiva $stop^*$ quando l'origine si trova nel termine O_{end}. Solo in questo caso, infatti, l'origine ha inviato l'ultimo messaggio contenente la chiave K_c che permette di ricavare M, senza avere ricevuto l'ultimo ack che avrebbe completato lo scambio equo di informazione. Rendiamo esplicita quest'ultima situazione attraverso l'esecuzione dell'azione speciale **unfair**.

$$
\begin{array}{lcl}
O(M, K, K_c) & = & c^*(x).[<x,K> \mapsto^1_{dec} z]\ (\ [z \approx Req]\ O_{acc}(M, K, K_c)\ ) \\
O_{acc}(M, K, K_c) & = & c(\{M_0\}_K).c^*(x).[<x,K> \mapsto^1_{dec} z]\ (\ [z \approx A_0]\ O'(K, K_c)\ ) \\
O'(K, K_c) & = & O_1(K, K_c, A_1) +^{1/\eta} \\
 & & (O_2(K, K_c, A_1) +^{1/(\eta-1)} \ldots \\
 & & \ldots(O_{\eta-1}(K, K_c, A_1) +^{1/2} O_\eta(K, K_c, A_1)\ldots) \\
O_i(K, K_c, A_i) & = & c(\{M_i\}_K).O_i'(K, K_c, A_i) \\
O_i'(K, K_c, A_i) & = & (c^*(x).[<x,K> \mapsto^1_{dec} z]\ ([z \approx A_i]\ O_{i-1}(K, K_c, A_{i+1}))) + stop^* \\
O_1(K, K_c, A_i) & = & c(\{M_n\}_K).O_{end}(K, A_i) \\
O_{end}(K, A_i) & = & (c^*(x).[<x,K> \mapsto^1_{dec} z]\ (\ [z \approx A_i]\ stop\ )\ ) + stop^*.\mathbf{unfair}
\end{array}
$$

Tab. 6.2. Protocollo probabilistico di non-repudiation - Modello dell'origine.

Nella tab. 6.3 mostriamo la specifica algebrica di un processo ricevente che si comporta in maniera onesta (*HR*). Il processo *HR* dà inizio al protocollo

inviando una richiesta (attraverso l'azione $c(\{Req\}_K)$) e quindi attende il primo messaggio. Quindi per ogni messaggio ricevuto invia il relativo ack e aggiunge il messaggio ricevuto alla propria conoscenza. Il protocollo termina con l'esecuzione dell'azione reattiva $stop^*$ indicando la ricezione dell'ultimo messaggio contenente la chiave K_c necessaria per mettere M in chiaro.

$HR(K)$	=	$c(\{Req\}_K).c^*(x).[<x, K> \mapsto^1_{dec} z]\ (\ c(\{A_0\}_K).HR_1(K, z)\ )$
$HR_i(K, N)$	=	$c^*(x).[<x,K> \mapsto^1_{dec} z]\ ([<N, z> \mapsto^1_{pair} y]\ (c(\{A_i\}_K).HR_{i+1}(K, y)$
		$+\ stop^*\ )\ )$

Tab. 6.3. Protocollo probabilistico di non-repudiation - Modello del ricevente onesto.

Per completare la descrizione algebrica del protocollo specifichiamo l'interfaccia di comunicazione tra partecipanti componendo in parallelo i processi $O(M, K, K_c)$ e $HR(K)$ nel sistema $O(M, K, K_c)\ \|^p_S\ HR(K)$, in cui i due processi si sincronizzano sulle azioni il cui tipo è in $S = \{c, stop\}$ che contiene tutte le azioni utilizzate eccetto l'azione speciale unfair. Si noti che:

- il parametro p non è rilevante (nel seguito lo ometteremo) dal momento che in nessuno stato del sistema composto è possibile effettuare una scelta probabilistica tra le azioni di O e di HR;
- il sistema composto $O(M, K, K_c)\ \|^p_S\ HR(K)$ è *performance closed*, ovvero non include scelte non-deterministiche.

Sottolineiamo inoltre che, dal momento che nel caso della non-repudiation eventuali attacchi sono portati dai componenti interni al sistema, non utilizziamo il processo ***C*** per la modellazione delle comunicazioni sicure su canali pubblici. Infatti, come vedremo nel prossimo paragrafo, definiremo la proprietà di non-repudiation attraverso un analisi di tutti i possibili riceventi disonesti che possono partecipare al protocollo e non attraverso l'analisi di

tutti i possibili processi esterni che potrebbero interagire con il protocollo. Ricordiamo che ***C*** era stato introdotto per permetterci di definire l'interazione del protocollo con processi esterni.

6.3.1 Proprietà di non-repudiation

In linea con il modello iniziale definito nella sezione **5.2** per la verifica di proprietà di sicurezza di sistemi crittografici definiamo l'insieme *C* dei canali pubblici. E' sensato assumere che lo scambio di informazioni tra l'origine ed il ricevente avvenga attraverso dei canali pubblici, dal momento che il ricevente (che potrebbe essere considerato un potenziale processo ostile) può operare sulla rete in cui viene eseguito il protocollo. In particolare faranno parte di *C* le azioni il cui tipo è contenuto nell'insieme di sincronizzazione $S = \{c, stop\}$. L'azione osservabile $\mathbf{unfair} \notin C$ (che viene eseguita quando il protocollo non rispetta la proprietà di fairness) è invece stata inclusa nel modello per poter verificare la proprietà di non-repudiation. Intuitivamente, l'idea principale alla base della verifica della proprietà di non-repudiation, è che un utente onesto che segue l'esecuzione del protocollo non deve osservare l'evento legato all'azione **unfair**.

Anche in questo caso formalizzeremo la proprietà di non-repudiation attraverso il classico approccio della *NDC*. Rispetto alle proprietà *PBNDC* e *CryptoPBNDC*, in cui viene osservato il comportamento del sistema in parallelo con l'ambiente esterno, nel caso della proprietà di non-repudiation un comportamento disonesto è dovuto ad un componente interno al sistema che cerca di violare la condizione di fairness. Non è quindi significativo osservare possibili attacchi di un terzo partecipante esterno, non coinvolto nel protocollo. In particolare attacchi alla proprietà di autenticazione sono prevenuti attraverso l'invio, durante il protocollo, di espressioni cifrate con una chiave condivisa (abbiamo infatti supposto che il protocollo sia preceduto da una fase sicura di autenticazione), attacchi attraverso messaggi replicati sono invece prevenuti tramite l'utilizzo dei marcatori temporali. Per questi motivi non è stato necessario definire il processo ***C*** per regolare le comunicazioni sui canali pubblici, ovvero non ci interessa, in questo caso,

modellare l'interazione del protocollo con processi esterni. Dal momento che l'origine non trae alcun vantaggio dall'interrompere l'esecuzione del protocollo prima della sua terminazione naturale, l'ambiente ostile è rappresentato dal processo ricevente che potrebbe tentare di ottenere l'informazione desiderata senza inviare l'ultimo messaggio di ack. Sulla base di queste considerazioni verifichiamo la proprietà di non-repudiation controllando l'equivalenza semantica tra il protocollo in cui entrambi i partecipanti si comportano onestamente ed ogni possibile modello del protocollo in cui è coinvolto un potenziale ricevente disonesto.

Formalmente la proprietà viene verificata nel seguente modo:

Non-repudiation:

il protocollo soddisfa la proprietà di non-repudiation $\Leftrightarrow$

$$((O \|_S HR) /S) \setminus C \approx_{PB} ((O \|^{p}_{\{c_1,\ldots,c_k\}} \Pi) /^{\mathbf{p}}_{c_1 \ldots c_k}) \setminus C$$

$\forall$ sequenza $c_1 \ldots c_k \in C^*$, $\forall \Pi \in \mathcal{G}_{C,\Delta}$, $\forall p \in]0, 1[$, $\forall \mathbf{p} \in Seq^{k}_{]0,1[}$.

Informalmente osserviamo che il protocollo non soddisfa tale proprietà di sicurezza. In particolare, se entrambi i partecipanti si comportano in maniera onesta, non viene mai osservato un comportamento scorretto; tuttavia è possibile individuare un ricevente disonesto che riceve l'informazione voluta senza inviare l'ultimo acknowledgement.

Formalmente, il processo *HR* non interrompe mai prematuramente il protocollo attraverso l'azione *stop*, di conseguenza l'azione reattiva $stop^*$ del termine O_{end} non può venire abilitata, né l'azione **unfair** può essere eseguita. Di conseguenza, se entrambi i partecipanti sono onesti, il protocollo è fair con probabilità 1. In particolare è semplice verificare che:

$Prob(((O \|_S HR) /S) \setminus C, \tau^* \mathbf{unfair}, C_i) = 0 \quad \forall C_i \in \mathcal{G}/\approx_{PB}$ e che:

$\sum_{C_i \in \mathcal{G}/\approx_{PB}} Prob(((O \|_S HR) / S) \setminus C, \tau^*, C_i) = 1$.

Nella tab. 6.4 è descritto il modello di un ricevente disonesto Π che può abilitare l'esecuzione dell'azione **unfair** se composto con l'origine *O*. Se assumiamo che il ricevente conosca lo spazio e la distribuzione di probabilità utilizzati dall'origine per stabilire il numero *n* dei passi del protocollo, può massimizzare la probabilità di indovinare quale sia l'ultimo messaggio inviato attraverso una strategia intelligente.

Ad ogni passo del protocollo, la probabilità di ricevere l'ultimo messaggio è $1/\eta$, in cui η è la cardinalità dello spazio di campionamento, ovvero il massimo numero di passi che il protocollo può effettuare. Dopo aver ricevuto il primo messaggio, contenente l'espressione *M* cifrata con la chiave K_c, il ricevente interrompe il protocollo attraverso l'azione *stop* con probabilità $1/\eta$, oppure invia il relativo ack con probabilità $1\text{-}1/\eta$. Se il ricevente invia l'ack e il protocollo non è terminato, dopo la ricezione del secondo messaggio può interrompe il protocollo con probabilità $1/(\eta\text{ -}1)$, poiché la probabilità che quello sia l'ultimo messaggio è ridistribuita tra gli η-1 eventi dello spazio di campionamento ridotto (si vedano i termini Π_i). Infine, se dopo η-1 passi il protocollo non è ancora terminato, il ricevente è sicuro che il prossimo messaggio sarà l'ultimo e quindi interrompe il protocollo eseguendo l'azione *stop* con probabilità 1 (si veda il termine Π_η).

$$
\begin{array}{lcl}
\Pi(K) & = & c(\{Req\}_K).c^*(x).[<x, K> \mapsto^1_{dec} z]\ (\ c(\{A_0\}_K).\Pi_1(K, z)\ ) \\
\Pi_i(K, N) & = & (c^*(x).[<x, K> \mapsto^1_{dec} z]\ (\ [<N, z> \mapsto^1_{pair} y]\ (\ stop \quad +^{1/(\eta-i+1)} \\
 & & c(\{A_i\}_K).\Pi_{i+1}(K, y)\ )\ ) \quad \text{con } 1 \leq i < \eta \\
\Pi_\eta(K, N) & = & c^*(x).[<x, K> \mapsto^1_{dec} z]\ stop
\end{array}
$$

Tab. 6.4. Protocollo probabilistico di non-repudiation - Modello di un ricevente disonesto.

Calcoliamo la probabilità che il ricevente riesca ad eludere la proprietà fair:

- la probabilità che l'origine selezioni il valore 1 come numero di passi del protocollo è $\frac{1}{\eta}$, e la probabilità che il ricevente interrompa il protocollo dopo il primo passo è $\frac{1}{\eta}$, di conseguenza la probabilità di indovinare l'evento $n = 1$ è $\frac{1}{\eta} \cdot \frac{1}{\eta} = \frac{1}{\eta^2}$;

- la probabilità che l'origine selezioni il valore 2 è $\frac{1}{\eta}$, e la probabilità che il ricevente interrompa il protocollo dopo due passi è $\frac{\eta-1}{\eta} \cdot \frac{1}{\eta-1} = \frac{1}{\eta}$, in cui $\frac{\eta-1}{\eta}$ è la probabilità di inviare il primo ack e $\frac{1}{\eta-1}$ è la probabilità di interrompere il protocollo al secondo passo. La probabilità di indovinare l'evento $n = 2$ è ancora $\frac{1}{\eta} \cdot \frac{1}{\eta} = \frac{1}{\eta^2}$;

- la probabilità che l'origine selezioni il valore $i > 2$ è $\frac{1}{\eta}$, e la probabilità che il ricevente interrompa il protocollo dopo i passi è $\frac{\eta-1}{\eta} \cdot \frac{\eta-2}{\eta-1} \cdot \ldots \cdot \frac{1}{\eta-i+1} = \frac{1}{\eta}$, per cui la probabilità di indovinare l'evento $n = i$ è $\frac{1}{\eta} \cdot \frac{1}{\eta} = \frac{1}{\eta^2}$.

Dal momento che lo spazio di campionamento è composto da η eventi, la probabilità complessiva di indovinare quale sia l'ultimo messaggio ricevuto è $\eta \cdot \frac{1}{\eta^2} = \frac{1}{\eta}$, che è esattamente il valore che ci aspettavamo.

Formalmente, analizzando il sistema composto $((O \|_S \Pi) / S) \setminus C$, otteniamo che $Prob(((O \|_S \Pi) / S) \setminus C, \tau^* \textbf{unfair}, C_{\underline{0}}) = \frac{1}{\eta}$ (in cui $C_{\underline{0}}$ è la classe di equivalenza dello stato inattivo). Di conseguenza il protocollo non soddisfa la proprietà di non-repudiation. Tuttavia se sostituiamo la relazione

di equivalenza $\approx_{PB}$, con la bisimulazione con precisione ε ($\approx_{PB\varepsilon}$, vedi definizione 3.6), possiamo aggiungere che il protocollo soddisfa la proprietà di non-repudiation con tolleranza $\frac{1}{\eta}$. Infatti si può verificare formalmente che: $((O \parallel_S HR) /S) \setminus C \approx_{PB_{\frac{1}{\eta}}} ((O \parallel_S I\!I) /S) \setminus C$. Il livello di sicurezza del protocollo è quindi parametrizzato in base alla dimensione η dello spazio di campionamento. Tale parametro può essere stabilito dall'origine in funzione del livello di sicurezza che intende garantire.

Una strategia alternativa che può permettere ad un ricevente disonesto di ottenere l'informazione M senza inviare alcun ack potrebbe consistere nel tentare di decrittare il primo messaggio $\{M\}_{Kc}$ attraverso tentativi di crittonalisi. Come abbiamo visto, il nostro formalismo permette di modellare una situazione di questo tipo. Abbiamo tuttavia deciso di non considerare questa eventualità nel caso di studio poiché la probabilità di decrittare il messaggio $\{M\}_{Kc}$ (assumendo che venga utilizzato un algoritmo di cifratura efficace e non banale) è molto inferiore alla probabilità di indovinare il numero n dei passi dell'algoritmo. Lo spazio di campionamento in cui viene selezionato il valore n non potrà mai raggiungere, ad esempio, la dimensione dello spazio di campionamento in cui vengono scelte le chiavi dall'algoritmo *DES* (valori troppo alti di n non possono essere ammissibili, il protocollo impiegherebbe troppo tempo per essere eseguito). In particolare, fissato un parametro di sicurezza ε in base allo spazio di campionamento del numero dei passi $\{1, \ldots, \eta\}$, la probabilità di decrittare il blocco $\{M\}_{Kc}$ non influenza significamene tale soglia.

Concludiamo questo capitolo sottolineando che in un contesto non-deterministico potremmo solamente stabilire che il protocollo non garantisce la proprietà di non-repudiation del ricevente, dal momento che è possibile osservare un comportamento insicuro. Il modello probabilistico ci permette invece di stimare la probabilità con cui si può verificare tale comportamento

indesiderato, inoltre permette al partecipante onesto di fissare il fattore di rischio ε che definisce il livello di sicurezza del protocollo.

7 Conclusioni

Con l'obiettivo di definire uno strumento generale per la verifica di proprietà di sicurezza di sistemi multilivello e protocolli crittografici, abbiamo ideato un nuovo formalismo che permette di modellare sia il comportamento probabilistico di un sistema, sia le operazioni crittografiche di un protocollo. Abbiamo mostrato che attraverso tale formalismo è possibile definire proprietà di sicurezza che siano in grado di catturare flussi di informazione puramente probabilistici, e proprietà di sicurezza di protocolli crittografici. In particolare, come mostrato nel caso di studio, il nuovo modello proposto permette l'analisi di proprietà di sicurezza di protocolli crittografici che utilizzano algoritmi stocastici e possono quindi mostrare un comportamento probabilistico.

L'introduzione degli operatori di crittografia in un contesto probabilistico ha permesso, inoltre, di colmare un grave vuoto nella letteratura che tratta le definizioni di proprietà si sicurezza di sistemi crittografici attraverso approcci puramente formali. In maniera estremamente naturale, abbiamo infatti esteso il modello probabilistico per la verifica di proprietà di sicurezza anche al caso in cui non si assuma crittografia perfetta. In particolare il nostro apparato formale permette di modellare possibili attacchi all'algoritmo di cifratura, rendendo la specifica teorica di protocolli crittografici più simile al caso reale.

Un limite al nostro studio è dato tuttavia dall'utilizzo di proprietà di sicurezza basate sulla nozione della *NDC*. Queste prevedono, infatti, una quantificazione universale su tutti i possibili processi ostili che possono interagire con il sistema in analisi. Verificare questo tipo di proprietà potrebbe quindi risultare estremamente complesso. Le proprietà *SBSNNI* e *SBNDC* definite da Focardi e Gorrieri in [20] superano il problema legato alla verifica delle proprietà di *NDC*, tuttavia non sono applicabili all'analisi della sicurezza di protocolli crittografici.

Una soluzione alternativa alla quantificazione universale di tutti i processi ostili, già introdotta da Focardi, Gorrieri e Martinelli in [17, 18] nel contesto dell'equivalenza per tracce, potrebbe essere la definizione generale di un unico processo ostile che abbia le capacità e la conoscenza di ogni possibile intruso. Tale processo può essere immaginato come una sorta di *nemico più potente*. Questo permetterebbe di effettuare un unico controllo di equivalenza tra il sistema che si comporta in maniera sicura ed il sistema composto con il processo ostile più potente (se il comportamento del protocollo non viene alterato in composizione con il nemico più potente si può assumere che il protocollo sia sicuro).

Un possibile progetto futuro, che estenda il lavoro svolto finora, potrebbe consistere quindi nella ricerca di una definizione di nemico più potente nel contesto della bisimulazione. Una nozione di questo tipo renderebbe infatti notevolmente più semplice la verifica di proprietà di sicurezza di sistemi crittografici.

Riferimenti bibliografici

[1] Alessandro Aldini. “Probabilistic Information Flow in a Process Algebra”. In *Proceedings of the 12th International Conference on Concurrency Theory*. *Springer* LNCS 2154, pagine 152-168, Agosto 2001.

[2] Martin Abadi, Phillip Rogaway. “Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)”. *Journal of Cryptology*, 15(2): 103-127, 2002.

[3] Danny Dolev, Andrew C. Yao. “On the security of public key protocols”. *IEEE Transactions on Information Theory*, IT-29(12):198-208, Marzo 1983.

[4] Richard A. DeMillo, Nancy A. Lynch, Michael Merrit. “Cryptographic protocols”. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, 1982.

[5] Jonathan K. Millen, Sidney C. Clark, Sheryl B. Freedman. “The Interrogator: Protocol Security Analysis”. *IEEE Transactions on Software Engineering*, SE-13(2):274-288. Febbraio 1987.

[6] Richard A. Kemmerer. “Analyzing encryption protocol using formal verification techniques”. *IEEE Journal on Selected Areas in Communications*, 7(4):448-457, Maggio 1989.

[7] Michael Burrows, Martin Abadi, Roger Needham. “A logic of authentication”. In *Proceedings of the Royal Society of London*, 426:233-271, 1989.

[8] Catherine Meadows. “A system for the specification and analysis of key management protocols”. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pagine 182-195, 1991.

[9] Paul F. Syverson, Paul C. van Oorschot. “On unifying some cryptographic protocol logics”. In *IEEE Computer society Symposium on Research in Security and Privacy*, pagine 14-28, 1994.

[10] Steve Schneider. "Security properties and CSP". In *IEEE Computer society Symposium on Security and Privacy*, pagine 174-187, 1996.

[11] Lawrence C. Paulson. "The inductive approach to verifying cryptographic protocols". *Journal of computer security*, 6(1-2):85-128, 1998.

[12] C. A. R. Hoare. "Communicating sequential process". Prentice-Hall, 1985.

[13] R. J. van Glabbeek, S. A. Smolka, B. Steffen. "Reactive, Generative and Stratified Models of Probabilistic Processes". In *Information and computation*, 121:59-80, 1995.

[14] A. Aldini, M Bravetti. "An Asynchronous Calculus for Generative-Reactive Probabilistic Systems". In *Proceedings of the 8th internat. Workshop on Process Algebra and Performance Modeling*, Carleton Scientific, pagine 591-605, 2000.

[15] M. Bravetti, A. Aldini. "Discrete Time Generative-Reactive Probabilistic Processes with Different Advancing Speeds". Da pubblicare in *Theoretical Computer Science*.

[16] M. Bravetti, A. Aldini. "Expressing Processes with Different Action Durations through Probabilities". In *Proceedings of Joint International Workshop PAPM-PROBMIV*, *Springer LNCS* 2165, pagine 168-183, 2001.

[17] R. Focardi, F. Martinelli. "A Uniform Approach for the Definition of Security Properties". In *Proceedings of FM'99*, *LNCS* 1708, Tolosa (Francia), Settembre 1999.

[18] R. Focardi, R. Gorrieri, F. Martinelli. "Non Interference for the Analysis of Cryptographic Protocols". In *Proceedings of ICALP'00, Springer LNCS* 1853, pagine 744-755, Luglio 2000.

[19] R. Milner. "Communication and Concurrency". Prentice-Hall, 1989.

[20] R. Focardi, R. Gorrieri. "Classification of Security Properties (Part I: Information Flow)". *FOSAD 2000*, *LNCS 2171*, pagine 331-396, 2001.

[21] R. Focardi, R. Gorrieri. "The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties". *IEEE Transactions on Software Engineering*, 27:550-571, 1997.

[22] M. Abadi. "Secrecy by Typing in Security Protocols". *Journal od ACM*, 46(5):749-786, 1999.

[23] M. Abadi, A. D. Gordon. "A calculus for cryptographic protocols: The spi calculus". *Information and computation*, 148(1):1-70, 1999.

[24] C. Bodei, P.Degano, F. Nielson, H. Riis Nielson. "Static Analysis of Processes for No Read-Up and No Write-Down". In *Proceedings of 2nd FoSSaCS'99*, Amsterdam, Marzo 1999.

[25] C. Bodei, P. Degano, R. Focardi, C. Priami. "Authentication via Localized Names". In *Proceedings of CSFW'99*, pagine 98-110. IEEE press, 1999.

[26] C. Bodei, P. Degano, R. Focardi, C. Priami. "Primitives for Authentication in Process Algebras". *Theoretical Computer Science*, 2001.

[27] N. Durgin, J. Mitchell, D. Pavlovic. "Protocol composition and correctness". In *Proceedings of Workshop on Issues in the Theory of Security* (*WITS'00*), Università di Genova, Luglio 2000.

[28] R. Focardi. "Analysis and Automatic Detection of Information Flows in Systems and Networks". Tesi di dottorato, Università di Bologna, 1999.

[29] G. Lowe. "Casper: A Compiler for the Analysis of Security Protocols". *Journal of Computer Security*, 6:53-84, 1998.

[30] G. Lowe, B. Roscoe. "Using CSP to detect Errors in the TMN Protocol". *IEEE Transactions on Software Engineering*, 23(10):659-669, 1997.

[31] J. C. Mitchell, M. Mitchell, U. Stern. "Automated Analysis of Cryptographic Protocols Using Murϕ". In *Proceedings of the 1997 IEEE Symposium on Research in Security and Privacy*, pagine 141-153. IEEE Computer Society Press, 1997.

[32] L. C. Paulson. "Proving Properties of Security Protocols by Induction". In *10th Computer Security Foundations Workshop*, pagine 70-83. IEEE Computer Society Press, 1997.

[33] S. Schneider. "Verifying authentication protocols in CSP". *IEEE Transactions on Software Engineering*, 24(9), Settembre 1998.

[34] G. Smith, D. M. Volpano. "Secure Information Flow in a Multi-Threaded Imperative Language". In *Proceedings of POPL*, pagine 355-364, 1998.

[35] D. E. Bell, L. J. La Padula. "Secure Computer Systems: Unified Exposition and Multics Interpretation". *ESD-TR-75-306*, *MITRE MTR-2997*, Marzo 1976.

[36] C. R. Tsai, V. D. Gligor, C. S. Chandersekaran. "On the Identification of Covert Storage Channels in Secure Systems". *IEEE Transactions on Software Engineering*, pagine 569-580, Giugno 1990.

[37] J. A. Goguen, J. Meseguer. "Security Policy and Security Models". In *Proceedings of the 1982 Symposium on Security and Privacy*, pagine 11-20. IEEE Computer Society Press, Aprile 1982.

[38] J. T. Wittbold, D. M. Johnson. "Information Flow in Nondeterministic Systems". In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pagine 144-161. IEEE Computer Society Press, 1990.

[39] D. M. Johnson, F. J. Thayer. "Security and the Composition of Machines". In *Proceedings of the Computer Security Foundations Workshop*, pagine 72-89, Giugno 1988.

[40] D. McCullough. "Noninterference and the Composability of Security Properties". In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pagine 177-186. IEEE Computer Society Press, Aprile 1988.

[41] J. K. Millen. "Hookup Security for Synchronous Machines". In *Proceedings of the 3rd Computer Security Foundations Workshop*. IEEE Computer Society Press, 1990.

[42] D. Sutherland. "A Model of Information". In *proceedings of the 9th National Computer Security Conference*, pagine 175-183. National Bureau of Standards and National Computer Security Centre, Settembre 1986.

[43] P. Bieber, F. Cuppens. "A Logical View of Secure Dependencies". *Journal of Computer Security*, 1(1):99-129, 1992.

[44] R. Keller. "Formal Verification of Parallel Programs". *Communications of the ACM*, 19(7):561-572, 1976.

[45] C. Baier, H. Hermanns. "Weak Bisimulation for Fully Probabilistic Processes". In *Proceedings of the 9th International Conference on Computer Aided Verification*. Springer LNCS 1254, pagine 119-130, 1997.

[46] P. R. Halmos. "Measure Theory". Springer-Verlag, 1950.

[47] S. D. Brookes, C. A. R. Hoare, A. W. Roscoe. "A Theory of Communicating Sequential Processes". *Journal of the Association for Computing Machinery*, 31(3):560-599, Luglio 1984.

[48] R. De Nicola, M. Hennessy. "Testing equivalences for processes". *Theoretical Computer Science*, 34:83-133, 1984.

[49] R. Focardi, R. Gorrieri. "A Classification of Security Properties for Process Algebras". *Journal of Computer Security*, 3(1):5-33, 1994/1995.

[50] F. Martinelli. "Partial Model Checking and Theorem Proving for Ensuring Security". In *Proceedings of the 11th Computer Security Foundations Workshop*, (CSFW' 98). IEEE press, 1998.

[51] A. W. Roscoe, G. M. Reed, R. Forester. "The success and failures of behavioural models". In *Millennial Perspectives In Computer Science*. Davies, Roscoe e Woodcock (eds), Palgarave, 2000.

[52] A. Aldini, M. Bravetti, R, Gorrieri. "A Process Algebraic Approach for the Analysis of Probabilistic Non-interference". Technical Report UBLCS-2002-2, Marzo 2002.

[53] A. Sabelfeld, D. Sands. "A Per Model of Secure Information Flow in Sequential Programs". In *Proceedings of the 8th European Symposium on Programming*. Springer LNCS 1576, pagine 40-58, 1999.

[54] A. Sabelfeld, D. Sands. "Porbabilistic Noninterference for Multi-threaded Programs". In *Proceedings of the 13th Computer Security Foundations Workshop*. IEEE CS Press, 2000.

[55] J. W. Gray III. "Toward a Mathematical Foundation for Information Flow Security". *Journal of Computer Security*, 1:255-294, 1992.

[56] F. van Breugel, J. Worrell. "Towards Quantitative Verification of Probabilistic Systems (extended abstract)". In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*. Springer LNCS 2076, pagine 421-432, 2001.

[57] J. Desharnais, V. Gupta, R. Jagadeesan, P. Panangaden. "Metrics for Labelled Markov Processes". In *Proceedings of the 10th International Conference on Concurrency Theory*. Springer LNCS 1664, pagine 258-273, 1999.

[58] R. A. Howard. "Dynamic Probabilistic Systems". John Wiley & Sons, 1971.

[59] M. Bellare, A. Desai, E. Jokipii, P. Rogaway. "A Concrete Security Treatment of Symmetric Encryption". In *Proceedings of the 38th Symposium on Foundations of Computer Science*. IEEE, 1997.

[60] A. Durante, R. Focardi, R. Gorrieri. "A Compiler for Analysing Cryptographic Protocols using Non-Interference". *ACM Transactions on Software Engineering and Methodology* (*TOSEM*), 9(4):488-528, Ottobre 2000.

[61] R. Focardi, A. Ghelli, R. Gorrieri. "Using Non-Interference for the Analysis of Security Protocols". In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.