Stephanie Delaune, Steve Kremer
and Mark D. Ryan

Coercion-Resistance and Receipt-Freeness in Electronic Voting

Research Report LSV-06-08

April 2006

Laboratoire
Spécification
et
Vérification

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Coercion-Resistance and Receipt-Freeness in Electronic Voting *

Stéphanie Delaune
LSV, France Télécom R&D
ENS Cachan, CNRS, France
delaune@lsv.ens-cachan.fr

Steve Kremer
LSV, INRIA
ENS Cachan, CNRS, France
kremer@lsv.ens-cachan.fr

Mark Ryan
School of Computer Science
University of Birmingham, UK
M.D.Ryan@cs.bham.ac.uk

Abstract

In this paper we formally study important properties of electronic voting protocols. In particular we are interested in coercion-resistance and receipt-freeness. Intuitively, an election protocol is coercion-resistant if a voter A cannot prove to a potential coercer C that she voted in a particular way. We assume that A cooperates with C in an interactive way. Receipt-freeness is a weaker property, for which we assume that A and C cannot interact during the protocol: to break receipt-freeness A later provides evidence (the receipt) of how she voted. While receipt-freeness can be expressed using observational equivalence from the applied pi calculus, we need to introduce a new relation to capture coercion-resistance. Our formalization of coercion-resistance and receipt-freeness are quite different. Nevertheless, we show in accordance with intuition that coercion-resistance implies receipt-freeness, which implies privacy, the basic anonymity property of voting protocols, as defined in previous work. Finally we illustrate the definitions on a simplified version of the Lee et al. voting protocol.

1. Introduction

Electronic voting promises the possibility of a convenient, efficient and secure facility for recording and tallying votes. Many protocols specifying the interaction between voters and election administrators have been proposed [6, 9, 11, 3, 10, 7, 12, 14]. While voting of this kind appears to encourage higher voter turnout, it also carries the potential of making abuse easier to perform and easier to perform at a large scale. Such protocols involve a high level of concurrency and even small protocols are known to be extremely error-prone. Formal analysis is crucial to assess their security.

* This work has been partly supported by the RNTL project PROUVÉ 03V360 and the ACI-SI Rossignol.

Among the properties considered desirable of electronic voting protocols are the following:

Privacy: the system cannot reveal how a particular voter voted.

Coercion-resistance: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

The privacy property guarantees that the link between a voter and her vote remains secret. Coercion-resistance guarantees that the coercer cannot become convinced of how a voter votes, even if the voter cooperates with him. Of course the voter can tell a coercer how she voted, but coercion-resistance asserts that she is unable to prove it, so the coercer has no reason to believe her. Intuitively, coercion-resistance is a stronger property than privacy, since if it is possible for a coercer to detect the value of a voter's vote without the voter's cooperation, then it is certainly also possible with the voter's cooperation. Receipt-freeness is a related property that has also been studied in the literature, which is between coercion-resistance and privacy in strength:

Receipt-freeness: a voter does not gain any information (a *receipt*) which can be used to prove to a coercer that she voted in a certain way.

Note that in literature the distinction between receipt-freeness and coercion-resistance is not very clear. The definitions are usually given in natural language and are insufficiently precise to allow comparison. The notion of receipt-freeness first appeared in [4]. Since then, several schemes [4, 15] were proposed in order to meet the condition of receipt-freeness, but later shown not to satisfy it. One of the reasons for such flaws is that no formal definition of receipt-freeness has been given. The situation for coercion-resistance is similar. Systems have been proposed aiming to satisfy it; for example, Okamoto [16] presents a system resistant to interactive coercers, thus aiming to satisfy what we call coercion-resistance, but this property is stated only in natural language. Recently, a rigorous definition in a computational model has been proposed for coercion-resistance [12]. We present in this paper what we believe to be the first "formal methods" definition of receipt-freeness and coercion-resistance. We use the applied pi calculus framework and we make a clear distinction between the two notions. One of the advantages of the applied pi calculus is that it allows us to reason about equational theories that allow us to model the less classical cryptographic primitives often used in voting protocols. It seems difficult to compare our definition and the one given in [12] due to the inherently different models. We believe that the two notions are in fact incomparable.

As is often done in protocol analysis, we assume the Dolev-Yao abstraction: cryptographic primitives are assumed to work perfectly, and the attacker controls the public channels. The attacker can see, intercept and insert messages on a public channel, but can only encrypt, decrypt or sign messages for which he has the relevant key. In the case of both receipt-freeness and coercion-resistance, we assume that the coercer has all the capabilities of the attacker on the public channels.

We consider that the difference between receipt-freeness and coercion-resistance lies in the powers of the coercer to interact with the voter during the voting stage. In receipt-freeness, we assume a coercer who simply examines evidence gained from observing the election process. Such evidence includes information provided by the cooperating voter, *e.g.*, the voter's private key and random coins used for probabilistic encryption. In coercion-resistance, the coercer has additional capabilities. He can interact with the cooperating voter, for example by (adaptively) preparing messages which the voter will send during the process.

Coercion-resistance cannot possibly hold if the coercer can physically vote on behalf of the voter. Some mechanism is necessary for isolating the voter from the coercer at the moment she cases her vote. This can be realised by a voting booth, which we model here as a restricted, i.e., private and anonymous channel between the voter and the election administrators.

Receipt-freeness is formalised as an observational equivalence. Intuitively, a protocol is receipt-free if a coercer cannot detect a difference between Alice voting in the way he instructed, and her voting in some other way, provided Bob votes in the complementary way each time. Alice cooperates with the coercer by sharing secrets, but the coercer cannot interact with Alice to give her some prepared messages.

In the case of coercion-resistance, the coercer is assumed to communicate with Alice during the protocol, and can prepare messages which she should send during the election process. This gives the coercer much more power. It turns out that observational equivalence is not flexible enough, and we generalise it to a notion which we call *adaptive simulation*. We expect that adaptive simulation will prove to be of interest for other kinds of properties and protocols than the ones we study here.

Although the three properties are formalised in markedly different ways, we prove that, in accordance with intuition, coercion-resistance implies receipt-freeness, which in turn implies privacy.

It is rather classical to formalize anonymity properties as some kind of observational equivalence in a process algebra or calculus, going back to the work of Schneider and Sidiropoulos [17]. However, to the best of our knowledge this is the first formalisation of the notion of *not being able to prove* in this kind of framework.

Our formalization of coercion-resistance gives interesting insights, it enables the coercer to perform *fault attacks*. This kind of attack is also mentioned in [12] as a "randomization attack". The idea of a fault attack is to let the coercer test the behavior of a coerced voter by requiring this voter to send a *garbage* message at some point of the protocol. If the voter is unable to decide whether the message is garbage or not, the attacker may distinguish a voter following the coercer's instructions from a voter who is trying to cheat the coercer, as the protocol would block on the incorrect message. Verifying whether a message is correct or not is often difficult when for instance ciphertexts are sent. In practice, a coercer can use this technique to probabilistically check whether a coerced voter is behaving as expected. Avoiding these attacks requires to carefully choose some of the implementation issues.

We illustrate our definitions and the idea of a fault attack on a simplified version of a voting protocol proposed by Lee *et al.* [14].

Outline of the paper. In Section 2 we briefly recall the applied pi calculus. Next, in Section 3, we formalise receipt-freeness and coercion-resistance. We prove that coercion-resistance implies receipt-freeness, which implies privacy. We also introduce the fault attacks which are motivated by the definition of coercion-resistance. In Section 4, we illustrate our ideas in terms of the protocol by Lee *et al.* [14]. Finally, we conclude in Section 5.

2. The applied pi calculus

The applied pi calculus [2] is a language for describing concurrent processes and their interactions. It is based on the pi calculus, but is intended to be less pure and therefore more convenient to use. The applied pi calculus has been used to study a variety of security protocols, such as those for private authentication [8], for key establishment [1], as well as an electronic voting protocol [13].

To describe processes in the applied pi calculus, one starts with a set of *names* (which are used to name communication channels or other constants), a set of *variables*, and a *signature* Σ which consists of the function symbols which will be used to define terms. In the case of security protocols, typical function symbols will include enc for encryption, which takes plaintext and a key and returns the corresponding cipher text, and dec for decryption, taking ciphertext and a key and returning the plaintext. Terms are defined as names, variables, and function symbols applied to other terms. Terms and function symbols are sorted, and of course function symbol application must respect sorts and arities. By the means of an equational theory E we describe the equations which hold on terms constructed from the signature. We denote $=_E$ the equivalence relation induced by E. A typical example of an equational theory useful for cryptographic protocols is

$$dec(enc(x, k), k) = x.$$

For example, given this theory and terms $T_1 = dec(enc(enc(n, k_1), k_2), k_2)$ and $T_2 = enc(n, k_1)$, we have $T_1 =_E T_2$ (while obviously the syntactic equality $T_1 = T_2$ does not hold). We write $vars(T)$ for the set of variables occuring in T. When $vars(T) = \emptyset$ we say that the term T is *ground*.

In the applied pi calculus, one has (plain) processes and extended processes. Plain processes are built up in a similar way to processes in the pi calculus, except that messages can contain terms (rather than just names). In the grammar described below, M and N are terms, n is a name, x a variable and u is a metavariable, standing either for a name or a variable.

$P, Q, R :=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
if $M = N$ then P else Q	conditional
$\mathsf{in}(u, x).P$	message input
$\mathsf{out}(u, N).P$	message output

Extended processes add *active substitutions* and restriction on variables:

$A, B, C :=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

$\{^M/_x\}$ is the substitution that replaces the variable x with the term M. Active substitutions generalise "let". The process $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to "let $x = M$ in P". As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of free and bound variables and free and bound names of A, respectively. We say that an extended process is closed if all its variables are either bound or defined by an active substitution.

Active substitutions are useful because they allow us to map an extended process A to its *frame* $\phi(A)$ by replacing every plain process in A with 0. A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ can be viewed as an approximation of A that accounts for the static knowledge A exposes to its environment, but not A's dynamic behaviour. The domain of a frame φ, $\mathrm{dom}(\varphi)$, is the set of variables for which φ defines a substitution (those variables x for which φ contains a substitution $\{M/x\}$ not under a restriction on x). We denote by $\sigma_{|X}$ the substitution σ restricted to the set of variables X, *i.e.*, $\sigma_{|X} = \sigma(x)$ if $x \in X$ and $\sigma_{|X} = x$ otherwise.

An evaluation context $C[\_]$ is an extended process with a hole instead of an extended process. Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, by application of evaluation contexts, and satisfying some further basic structural rules such as $A \mid 0 \equiv A$, associativity and commutativity of $|$, binding-operator-like behaviour of ν, and when $M =_E N$ the equivalences

$$\nu x.\{^M/_x\} \equiv 0 \qquad \{^M/_x\} \equiv \{^N/_x\}$$
$$\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}.$$

Example 1 *Consider the following process P:*

$$\nu s, k.(\mathsf{out}(c_1, enc(s, k)) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, dec(y, k))).$$

The first component publishes the message $enc(s, k)$ by sending it on c_1. The second receives a message on c_1, uses the secret key k to decrypt it, and forwards the resulting plaintext on c_2. P is structurally equivalent to the following extended process A:

$$A = \nu s, k, x_1, x_2.(\mathsf{out}(c_1, x_1) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, x_2) \mid \{enc(s, k)/x_1, dec(y, k)/x_2\})$$

We have $\phi(A) = \nu s, k, x_1, x_2.\{enc(s, k)/x_1, k/x_2\} \equiv 0$ (x_1 and x_2 are under a restriction).

The following lemmas will be useful in the remaining of the paper. Their proofs are given in appendix.

Lemma 1 *Let C be an evaluation context. There exist a sequence $\tilde{u}$ of names and variables and an extended process B such that for all extended process A, we have $C[A] \equiv \nu\tilde{u}.(A \mid B)$.*

Lemma 2 *Let C_1, C_2 be two evaluation contexts. For any extended process A, we have $C_1[C_2[A]] \equiv C_2[C_1[A]]$.*

We can now define what it means for two frames to be statically equivalent [2].

Definition 1 (Static equivalence ($\approx_s$)) *Two closed frames φ_1 and φ_2 are* statically equivalent*, written $\varphi_1 \approx_s \varphi_2$, if* *and only if for some names $\tilde{n}_1$, $\tilde{n}_2$ and substitutions σ_1, σ_2, such that $\varphi_1 \equiv \nu\tilde{n}_1\sigma_1$ and $\varphi_2 \equiv \nu\tilde{n}_2\sigma_2$, we have*

(i) $\mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$,

(ii) for all terms M, N with variables included in $\mathrm{dom}(\varphi_i)$ and using no names occurring in $\tilde{n}_1$ or $\tilde{n}_2$, $M\sigma_1 =_E N\sigma_1$ is equivalent to $M\sigma_2 =_E N\sigma_2$.

Two extended processes A and B are statically equivalent if and only if $\phi(A) \approx_s \phi(B)$.

Example 2 *Let $\varphi_0 = \nu k.\sigma_0$ and $\varphi_1 = \nu k.\sigma_1$ where $\sigma_0 = \{enc(s_0, k)/x_1, k/x_2\}$, $\sigma_1 = \{enc(s_1, k)/x_1, k/x_2\}$ and s_0, s_1 and k are names. Let E be the theory defined by the axiom $dec(enc(x, k), k) = x$. We have $dec(x_1, x_2)\sigma_0 =_E s_0$ but not $dec(x_1, x_2)\sigma_1 =_E s_0$. Therefore, $\varphi_0 \not\approx_s \varphi_1$ although $\nu k.\{enc(s_0, k)/x_1\} \approx_s \nu k.\{enc(s_1, k)/x_1\}$.*

The operational semantics of processes in the applied pi calculus is defined by structural rules defining two relations: *structural equivalence* (described above) and *internal reduction*, noted $\rightarrow$. Internal reduction $\rightarrow$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that $\mathsf{out}(a, x).P \mid \mathsf{in}(a, x).Q \rightarrow P \mid Q$ and for any ground terms M and N, whenever $M \neq_E N$, we have

$$\text{if } M = M \text{ then } P \text{ else } Q \quad \rightarrow \quad P$$
$$\text{if } M = N \text{ then } P \text{ else } Q \quad \rightarrow \quad Q.$$

The operational semantics is extended by a *labeled* operational semantics enabling us to reason about processes that interact with their environment. Labeled operational semantics defines the relation $\xrightarrow{\alpha}$ where α is either an input, or the output of a channel name or a variable of base type. We adopt the following rules in addition to the internal reduction rules.

$$\mathsf{in}(a, x).P \xrightarrow{in(a,M)} P\{^M/_x\}$$

$$\mathsf{out}(a, u).P \xrightarrow{out(a,u)} P$$

$$\frac{A \xrightarrow{out(a,u)} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.out(a,u)} A'}$$

$$\frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

$$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$$

Note that the labeled transition is not closed under application of evaluation contexts. Moreover the output of a term M needs to be made "by reference" using a restricted variable and an active substitution.

Example 3 *Consider the process P defined in Example 1. A sequence of transitions is described in Figure 2. Let A be the extended process obtained after this sequence of reduction steps, the frame $\phi(A)$ associated to this extended process is $\nu s, k.\{enc(s, k)/x_1, s/x_2\}$.*

Definition 2 (Labeled bisimilarity ($\approx_\ell$)) *Labeled bisimilarity is the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A \mathcal{R} B$ implies*

1. *$A \approx_s B$,*
2. *if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B',*
3. *if $A \xrightarrow{\alpha} A'$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B'.*

In [2], it is shown that labeled bisimilarity coincides with observational equivalence. We prefer to work with labeled bisimilarity, rather than observational equivalence, because proofs for labeled bisimilarity are generally easier. Labeled bisimilarity can be used to formalize many security properties, in particular anonymity properties, such as those studied in this paper.

3. Formalisation

In this section, we show how both receipt-freeness and coercion-resistance can be formalised. While receipt-freeness is expressed in terms of labeled bisimilarity, coercion-resistance requires the definition of a new simulation relation which we call *adaptive simulation*.

3.1. Voting protocols in applied pi-calculus

Before defining the properties, we need to define what is an electronic voting protocol in applied pi calculus. Different voting protocols often have substantial differences. However, we believe that a large class of voting protocols can be represented by processes corresponding to the following structure.

Definition 3 (Voting process) *A voting process is a closed plain process*

$$VP \equiv \nu\tilde{n}.(V\sigma_1 \mid \cdots \mid V\sigma_n \mid A_1 \mid \cdots \mid A_m).$$

The $V\sigma_i$ are the voter processes and the A_js the different election authorities and the $\tilde{n}$ are channel names. We also suppose that $v \in \mathrm{dom}(\sigma_i)$ is a variable which refers to the value of the vote and at some moment the outcome of the vote is made public. More formally, this means that there exists A such that $VP(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* A$ and $\phi(A) \equiv \varphi \mid \{^{v\sigma_1}/_{x_1}, \ldots, ^{v\sigma_n}/_{x_n}\}$ for some φ.

We also define an evaluation context S which is as VP, but has a hole instead of two of the $V\sigma_i$.

$$\begin{array}{lcl} P & \equiv & \nu s,k,x_1.(\mathsf{out}(c_1,x_1) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,dec(y,k)) \mid \{enc(s,k)/x_1\}) \\ & \xrightarrow{\nu x_1.out(c_1,x_1)} & \nu s,k.(\mathsf{in}(c_1,y).\mathsf{out}(c_2,dec(y,k)) \mid \{enc(s,k)/x_1\}) \\ & \xrightarrow{in(c_1,x_1)} & \nu s,k.(\mathsf{out}(c_2,dec(x_1,k)) \mid \{enc(s,k)/x_1\}) \\ & \equiv & \nu s,k,x_2.(\mathsf{out}(c_2,x_2) \mid \{enc(s,k)/x_1\} \mid \{dec(x_1,k)/x_2\}) \\ & \xrightarrow{\nu x_2.out(c_1,x_2)} & \nu s,k.\{enc(s,k)/x_1\} \mid \{dec(x_1,k)/x_2\} \end{array}$$

Figure 1: Transitions

Note that it is likely that a voting process will have the property that for any permutation π on $\{1,\ldots,n\}$ there exists A such that $VP(\rightarrow^* \xrightarrow{\alpha} {}^*)^* A$ and $\phi(A) \equiv \varphi \mid \{{}^{v\sigma_1}/_{x_{\pi(1)}},\ldots,{}^{v\sigma_n}/_{x_{\pi(n)}}\}$ for some φ; in words, it is able to mix up the votes before publishing them. However, we don't require this property in general. Protocols that do not respect this property most likely do not respect privacy, but we do not wish to exclude them.

3.2. Formalising privacy and receipt-freeness

Similarly to privacy, receipt-freeness may be formalised as an observational equivalence. In [13], we modeled an election scheme as an applied pi calculus process VP, and formalised privacy as the observational equivalence between VP and another version of VP in which two voters V_A and V_B have swapped their votes. In the remainder of the paper we denote by V_A and V_B two particular processes such that $V_A = V\sigma_{A|\mathrm{dom}(\sigma_A)\setminus\{v\}}$ and $V_B = V\sigma_{B|\mathrm{dom}(\sigma_B)\setminus\{v\}}$ where $\mathrm{dom}(\sigma_A) = \mathrm{dom}(\sigma_B) = fv(V)$. Hence v is the only free variable occuring in V_A, respectively V_B. Formally, privacy is defined as follows.

Definition 4 (Privacy) *A voting protocol respects* privacy *if*

$$S[V_A\{{}^a/_v\} \mid V_B\{{}^b/_v\}] \quad \approx_\ell \quad S[V_A\{{}^b/_v\} \mid V_B\{{}^a/_v\}].$$

The intuition is that if an intruder cannot detect if arbitrary honest voters V_A and V_B swap their votes, then in general he cannot know anything about how V_A (or V_B) voted. Note that this definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed: for example, if the vote is unanimous, or if all other voters reveal how they voted and thus allow the votes of V_A and V_B to be deduced.

We also formalize receipt-freeness using observational equivalence. However, we need to model the fact that V_A is willing to provide secret information, *i.e.*, the receipt, to the coercer. We assume that the coercer is in fact the intruder who, as usual in the Dolev-Yao model, controls the public channels. To model V_A's communication with the coercer, we consider that V_A executes a voting process which has been modified: any input and any freshly generated names are forwarded to the coercer.

Definition 5 (Process P^{ch}) *Let P be a plain process and ch a channel name. We define P^{ch} inductively as follows:*

- $0^{ch} \,\hat{=}\, 0$,
- $(P \mid Q)^{ch} \,\hat{=}\, P^{ch} \mid Q^{ch}$,
- $(\nu n.P)^{ch} \,\hat{=}\, \nu n.\mathsf{out}(ch,n).P^{ch}$,
- $(\mathsf{in}(u,x).P)^{ch} \,\hat{=}\, \mathsf{in}(u,x).\mathsf{out}(ch,x).P^{ch}$,
- $(\mathsf{out}(u,M).P)^{ch} \,\hat{=}\, \mathsf{out}(u,M).P^{ch}$,
- $(!P)^{ch} \,\hat{=}\, !P^{ch}$,
- *(if $M = N$ then P else Q)^{ch} $\hat{=}$ if $M = N$ then P^{ch} else Q^{ch}.*

In the remainder, we implicitly assume that $ch \notin fn(P) \cup bn(P)$ before applying the transformation.

Given an extended process A and a channel name ch, we need to define the extended process $A^{\setminus out(ch,\cdot)}$. Intuitively, such a process is as the process A, but hiding the ouputs on the channel ch.

Definition 6 (Process $A^{\setminus out(ch,\cdot)}$) *Let A be an extended process. We define*

$$A^{\setminus out(ch,\cdot)} \,\hat{=}\, \nu ch.(A \mid !\mathsf{in}(ch,x)).$$

We are now ready to define receipt-freeness. Intuitively, a protocol is receipt-free if, for all voters V_A, the process in which V_A votes according to the intruder's wishes is indistinguishable from the one in which she votes something else. As in the case of privacy, we express this as an observational equivalence to a process in which V_A swaps her vote with V_B, in order to avoid the case in which the intruder can distinguish the situations merely by counting the votes at the end. Suppose the coercer's desired vote is c. Then we define receipt-freeness as follows.

Definition 7 (Receipt-freeness) *A voting protocol is* receipt-free *if there exists a closed plain process V', such that $V'^{\setminus out(chc,\cdot)} \approx_\ell V_A\{{}^a/_v\}$ and*

$$S[V_A\{{}^c/_v\}^{chc} \mid V_B\{{}^a/_v\}] \quad \approx_\ell \quad S[V' \mid V_B\{{}^c/_v\}].$$

V' is a process in which voter V_A votes a but communicates with the coercer C in order to feign cooperation with him. Thus, the equivalence says that the coercer cannot tell the difference between a situation in which V_A genuinely cooperates with him in order to cast the vote c and one in which she pretends to cooperate but actually casts the vote a, provided there is some counterbalancing voter that votes the other way around.

According to intuition, whenever a protocol is receipt-free, it also respects privacy.

Proposition 1 *Let VP be a voting protocol.*

$$VP \text{ is receipt-free} \implies VP \text{ respects privacy.}$$

To prove this proposition we introduce two lemmas.

Lemma 3 *Let A be an extended process, C an evaluation context and ch a channel name. If $ch \notin fn(C[0])$ we have*

$$C[A]^{\backslash out(ch,\cdot)} \equiv C[A^{\backslash out(ch,\cdot)}].$$

The proof is given in appendix.

Lemma 4 *Let P be a closed plain process and ch a channel name such that $ch \notin fn(P) \cup bn(P)$. We have*

$$(P^{ch})^{\backslash out(ch,\cdot)} \approx_\ell P.$$

Proof.(sketch, see Appendix for details) We show by induction on P that for any substitution ρ closing P, we have $(P\rho)^{ch\backslash out(ch,\cdot)} \approx_\ell P\rho$. The base case where $P = 0$ is trivial. Then, we consider the different possibilities for building P. $\square$

Now, we are able to prove Propositon 1.

Proof.[of Proposition 1] By hypothesis, there exists a closed plain process V', such that $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$ and

$$S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \quad \approx_\ell \quad S[V' \mid V_B\{^c/_v\}].$$

By applying the evalution context $\nu chc.(\_|!\mathsf{in}(chc, x))$ on both sides we obtain that:

$$\begin{array}{c} S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]^{\backslash out(chc,\cdot)} \\ \approx_\ell \\ S[V' \mid V_B\{^c/_v\}]^{\backslash out(chc,\cdot)}. \end{array}$$

Thanks to Lemma 3, we deduce that:

$$\begin{array}{c} S[V_A\{^c/_v\}^{chc\backslash out(chc,\cdot)} \mid V_B\{^a/_v\}] \\ \approx_\ell \\ S[V'^{\backslash out(chc,\cdot)} \mid V_B\{^c/_v\}]. \end{array}$$

Finally, the equivalence $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$ and Lemma 4 allow us to conclude. $\square$

3.3. Formalising coercion-resistance

Coercion-resistance is a stronger property as we give the coercer the ability to communicate *interactively* with the voter and not only receive information. In this model, the coercer can for instance prepare the messages he wants the voter to send. As for receipt-freeness, we are going to modify the voter process. However, we give the coercer the possibility to provide the messages the voter should send.

Definition 8 (Process P^{c_1,c_2}) *Let P be a plain process and c_1, c_2 be channel names. We define P^{c_1,c_2} inductively as follows:*

- $0^{c_1,c_2} \mathrel{\widehat{=}} 0$,
- $(P \mid Q)^{c_1,c_2} \mathrel{\widehat{=}} P^{c_1,c_2} \mid Q^{c_1,c_2}$,
- $(\nu n.P)^{c_1,c_2} \mathrel{\widehat{=}} \nu n.\mathsf{out}(c_1, n).P^{c_1,c_2}$,
- $(\mathsf{in}(u, x).P)^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{in}(u, x).\mathsf{out}(c_1, x).P^{c_1,c_2}$,
- $(\mathsf{out}(u, M).P)^{c_1,c_2} \mathrel{\widehat{=}} \mathsf{in}(c_2, x).\mathsf{out}(u, x).P^{c_1,c_2}$ *where x is a fresh variable ,*
- $(!P)^{c_1,c_2} \mathrel{\widehat{=}} !P^{c_1,c_2}$,
- $(\textit{if } M = N \textit{ then } P \textit{ else } Q)^{c_1,c_2} \mathrel{\widehat{=}}$ $\textit{if } M = N \textit{ then } P^{c_1,c_2} \textit{ else } Q^{c_1,c_2}$.

As a first approximation, we could try to define coercion-resistance in the following way:

$$S[V_A\{^c/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}] \quad \approx_\ell \quad S[V' \mid V_B\{^c/_v\}].$$

This definition has an obvious problem as the coercer could oblige $V_A\{^c/_v\}^{c_1,c_2}$ to vote $c' \neq c$. In that case, the process $V_B\{^c/_v\}$ would not counterbalance the outcome to avoid a trivial way of distinguishing.

To properly define coercion-resistance, we define a new simulation relation which allows the second voting process on the right-hand side to dynamically adapt its vote to correspond to the coercer's choice. Before defining this relation itself we have to introduce some more notation.

Definition 9 (X_A) *Given an extended process A, we define X_A to be the set of unbound variables that are not defined by an active substitution, i.e.,*

$$X_A = fv(A) \setminus \mathrm{dom}(\phi(A)).$$

Example 4 *Consider the following extended process B:*

$$\begin{array}{c} \mathsf{out}(c_1, x_1).\mathsf{out}(c_2, x_2) \mid \mathsf{in}(c_1, y_1).\mathsf{in}(c_2, y_2) \mid \\ \{enc(z_1, z_2)/x_1\} \mid \{enc(z_1, z_3)/x_2\}. \end{array}$$

We have $X_B = \{z_1, z_2, z_3\}$.

Definition 10 ($X_A(A\sigma \overset{(\alpha)}{\to} A')$) *Let A be an extended process and σ be a substitution closing A, i.e., $\mathrm{dom}(\sigma) = X_A$. We define $X_A(A\sigma \to A')$ (resp. $X_A(A\sigma \overset{\alpha}{\to} A')$) to be the variables in X_A that are bound by σ during the internal reduction $A\sigma \to A'$ (resp. labeled transition $A\sigma \overset{\alpha}{\to} A'$). More formally, see Figure 2.*

We naturally extend this definition to sequences δ of reduction steps.

$X_{out(u,M).P_1|in(v,x).P_2}((\mathsf{out}(u,M).P_1 \mid \mathsf{in}(v,x).P_2)\sigma \to (P_1 \mid P_2\{^M/_x\})\sigma) = vars(M) \cup vars(u) \cup vars(v)$
$X_{\text{if } (M=N) \text{ then } P_1 \text{ else } P_2}((\text{if } (M=N) \text{ then } P_1 \text{ else } P_2)\sigma \to P_i\sigma) = vars(M) \cup vars(N)$
$X_A(A\sigma \to A') = X_B(B\sigma \to B')$ if $A \equiv B$ and $A' \equiv B'$
$X_{C[A]}(C[A]\sigma \to C[A']\sigma) = X_{A\rho}(A\rho\sigma \to A')$ if $\phi(C[A]) \equiv \nu\tilde{n}.\rho$

$X_{in(u,x).P}((\mathsf{in}(u,x).P)\sigma \xrightarrow{in(u\sigma,M)} P\sigma\{^M/_x\}) = vars(u)$
$X_{out(u,M).P}(\mathsf{out}(u,M).P\sigma \xrightarrow{out(u,M)\sigma} P\sigma) = vars(M) \cup vars(u)$.
$X_{\nu v.A_1}(\nu v.A_1\sigma \xrightarrow{\nu v.out(u,v)} A_2) = X_{A_1}(A_1\sigma \xrightarrow{out(u,v)} A_2)$ if $u\sigma \neq v\sigma$.
$X_{A_1}((\nu u.A_1)\sigma \xrightarrow{\alpha} \nu u.A_2) = X_{A_1}(A_1\sigma \xrightarrow{\alpha} A_2)$ if u does not occur in α.
$X_{A_1|B}((A_1 \mid B)\sigma \xrightarrow{\alpha} A_2 \mid B\sigma) = X_{A_1\rho}(A_1\rho\sigma \xrightarrow{\alpha} A_2)$ if $bv(\alpha) \cap fv(B) = \emptyset$ and $bn(\alpha) \cap fn(B) = \phi(A \mid B) \equiv \nu\tilde{n}.\rho$.
$X_A(A\sigma \xrightarrow{\alpha} A') = X_B(B\sigma \xrightarrow{\alpha} B')$ if $A \equiv B$ and $A' \equiv B'$.

Figure 2: $X_A(A\sigma \overset{(\alpha)}{\to} A')$

Definition 11 ($X_{A_0}(\delta)$) *Let $\delta = A_0\sigma \overset{(\alpha)}{\to} \dots \overset{(\alpha)}{\to} A_n\sigma$ be a (possibly empty) sequence of reduction steps. We inductively define X_δ as:*

$$X_{A_0}(\delta) = \begin{cases} \emptyset & \text{if } n = 0 \\ X_{A_0}(A_0\sigma \overset{(\alpha)}{\to} A_1\sigma) \cup \\ \quad X_{A_1}(A_1\sigma \overset{(\alpha)}{\to} \dots \overset{(\alpha)}{\to} A_n\sigma) & \text{otherwise} \end{cases}$$

Example 5 *Let B be the process defined in Example 4. We have $\phi(B) = \{enc(z_1,z_2)/x_1, enc(z_1,z_3)/x_2\}$. Let $\sigma = \{z_1 \mapsto a, z_2 \mapsto b, z_3 \mapsto c\}$. Let $B_1 = \mathsf{out}(c_2,x_2) \mid \mathsf{in}(c_2,y_2) \mid \{enc(a,b)/x_1\} \mid \{enc(a,z_3)/x_2\}$ and $B_2 = \{enc(a,b)/x_1\} \mid \{enc(a,c)/x_2\}$. We have $X_B(B\sigma \to B_1\sigma) = \{z_1,z_2\}$ and $X_{B_1}(B_1\sigma \to B_2) = \{z_3\}$.*

Definition 12 (Adaptive simulation ($\preceq_a$)) *Adaptive simulation is the largest relation $\mathcal{R}$ on extended processes A and B, such that $A \mathrel{\mathcal{R}} B$ implies that for all σ_A such that $\mathrm{dom}(\sigma_A) = X_A$*

1. *$A\sigma_A \approx_s B\sigma_B$ for some substitution σ_B with $\mathrm{dom}(\sigma_B) = X_B$;*
2. *if $A\sigma_A \to A'\sigma_A$ then there exists σ_B with $\mathrm{dom}(\sigma_B) = X_B$ such that*
   - *(a) there exists B' such that $B\sigma_B \to^* B'\sigma_B$ and $A'\theta_A \mathrel{\mathcal{R}} B'\theta_B$ where $\theta_A = \sigma_A|_{X_A(A\sigma_A \to A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_B(B\sigma_B \to^* B'\sigma_B)}$;*
   - *(b) if $B\sigma_B \to B'\sigma_B$ then there exists A' such that $A\sigma_A \to^* A'\sigma_A$ and $A'\theta_A \mathrel{\mathcal{R}} B'\theta_B$ where $\theta_A = \sigma_A|_{X_A(A\sigma_A \to^* A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_B(B\sigma_B \to B'\sigma_B)}$.*
3. *if $A\sigma_A \xrightarrow{\alpha} A'\sigma_A$ then there exists σ_B with $\mathrm{dom}(\sigma_B) = X_B$ such that*
   - *(a) there exists B' such that $B\sigma_B \to^* \xrightarrow{\alpha} \to^* B'\sigma_B$ and $A'\theta_A \mathrel{\mathcal{R}} B'\theta_B$ where $\theta_A = \sigma_A|_{X_A(A\sigma_A \xrightarrow{\alpha} A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_B(B\sigma_B \to^* \xrightarrow{\alpha} \to^* B'\sigma_B)}$;*
   - *(b) if $B\sigma_B \xrightarrow{\alpha} B'\sigma_B$ then there exists A' such that $A\sigma_A \to^* \xrightarrow{\alpha} \to^* A'\sigma_A$ and $A'\theta_A \mathrel{\mathcal{R}} B'\theta_B$ where $\theta_A = \sigma_A|_{X_A(A\sigma_A \to^* \xrightarrow{\alpha} \to^* A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_B(B\sigma_B \xrightarrow{\alpha} B'\sigma_B)}$.*

Intuitively, adaptive simulation models the fact that no matter how the process A is closed and no matter how the environment reacts, B can be closed, such that the processes are indistinguishable. The existential quantification on σ_B follows the reductions of A, in order to let B adapt to the inputs of the environment. While the *(a)* parts ensure that B indeed simulates A, the *(b)* parts ensure that B cannot "do anything more" than A. Finally, by the means of θ_A and θ_B we progressively close A and B, so that once their reductions depend on some variables they cannot be changed any more, i.e. they have to stick to their choices and cannot undo them anymore. Note that if A and B are closed processes $A \approx_\ell B$ if and only if $A \preceq_a B$. Moreover adaptive simulation is transitive and enjoys several other useful properties stated below.

Lemma 5 *Let A and B be two extended processes. If for all ρ_A with $\mathrm{dom}(\rho_A) = X_A$ there exists ρ_B with $\mathrm{dom}(\rho_B) = X_B$ such that $A\rho_A \approx_\ell B\rho_B$, then $A \preceq_a B$.*

Proof. It follows directly from the definition of $\preceq_a$. □

Note that the converse is not true. Consider the two processes A and B described below:
$A = \nu c.(\mathsf{in}(c_1,x).\mathsf{out}(c,a).\mathsf{out}(c_2,x) \mid \mathsf{in}(c,z).\mathsf{out}(c_2,a))$;
$B = \nu c.(\mathsf{in}(c_1,x).\mathsf{out}(c,a).\mathsf{out}(c_2,a) \mid \mathsf{in}(c,z).\mathsf{out}(c_2,y))$.
We have that $A \preceq_a B$. However there is no ρ_B such that for all ρ_A we have $A\rho_A \approx_\ell B\rho_B$.

Corollary 1 *Let A and B be two extended processes such that $X_A = X_B$. If for all substitution ρ with $\mathrm{dom}(\rho) = X_A = X_B$ we have $A\rho \approx_\ell B\rho$, then $A \preceq_a B$.*

We say that an evaluation context is *strict*, if the hole does not appear under a restriction of a variable.

Lemma 6 *Let A, B be two extended processes and C a strict evaluation context, such that $fv(C[0]) = \emptyset$. We have*

$$A \preceq_a B \implies C[A] \preceq_a C[B].$$

The proof of Lemma 6 is given in appendix. Note that in Lemma 6 we cannot consider any open context where $fv(C[0]) \neq \emptyset$. Consider for example $A = \{^a/_{x_1}\}$ and $B = \{^y/_{x_1}\}$. Then we have $A \preceq_a B$, while $A \mid \{^y/_{x_2}\} \not\preceq_a B \mid \{^y/_{x_2}\}$. One might want to relax this condition to $fv(C[0]) \cap fv(B) = \emptyset$. However, this stronger version is not needed in the remainder and would complicate the proofs. For a similar reason we also restrict ourselves to strict evaluation contexts, as $\nu y.A \not\preceq_a \nu y.B$.

We are now ready to define coercion-resistance.

Definition 13 (Coercion-resistance) *A voting protocol is* coercion-resistant *if there exists a closed extended process V' and a strict evaluation context C such that*

- $S[V_A\{^c/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}] \preceq_a S[V' \mid V_B\{^x/_v\}]$,
- $\nu c_1.\nu c_2.C[V_A\{^c/_v\}^{c_1,c_2}] \approx_\ell V_A\{^c/_v\}^{chc}$,
- $\nu c_1.\nu c_2.C[V']^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,

where x is a fresh free variable.

The intuition of this definition is that whenever the coercer requests a given vote on the left-hand side, then V_B can adapt his vote on the right-hand side and counter-balance the outcome. However, we need to avoid the case where $V' = V_A\{^c/_v\}^{c_1,c_2}$ letting V_B vote a. Therefore we require that when we apply a context C, intuitively the coercer, requesting $V_A\{^c/_v\}^{c_1,c_2}$ to vote c, V' in the same context votes a. There may be circumstances where V' may need not to cast a vote that is not a. We discuss those in Section 3.4.

Proposition 2 *Let VP be a voting protocol.*

$$VP \text{ is coercion-resistant} \implies VP \text{ is receipt-free.}$$

Proof. We need to show that there exists V'' such that

1. $V''^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
2. $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V'' \mid V_B\{^c/_v\}]$.

Let $V'' = \nu c_1.\nu c_2.C[V']$. We directly have 1. It remains to show 2.

(a) Thanks to Lemma 6, we know that:

$$\begin{array}{l} \nu c_1.\nu c_2.C[S[V_A\{^c/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}]] \\ \quad \preceq_a \ \nu c_1.\nu c_2.C[S[V' \mid V_B\{^x/_v\}]]. \end{array}$$

(b) We have

$$\begin{array}{ll} & \nu c_1.\nu c_2.C[S[V_A\{^c/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}]] \\ \equiv & S[\nu c_1.\nu c_2.C[V_A\{^c/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \text{ by Lemma 2} \\ \approx_\ell & S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \text{ by hypothesis} \end{array}$$

Therefore by (a),

$$S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \preceq_a \nu c_1.\nu c_2.C[S[V' \mid V_B\{^x/_v\}]].$$

(c) For any substitution ρ such that $\mathrm{dom}(\rho) = \{x\}$ we have

$$\nu c_1.\nu c_2.C[S[V' \mid V_B\{^{x\rho}/_v\}]] \approx_\ell S[V'' \mid V_B\{^{x\rho}/_v\}].$$

Using Corollary 1 we obtain that

$$\nu c_1.\nu c_2.C[S[V' \mid V_B\{^x/_v\}]] \preceq_a S[V'' \mid V_B\{^x/_v\}].$$

(d) By (b), (c), and transitivity of $\preceq_a$, we deduce that

$$S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \preceq_a S[V'' \mid V_B\{^x/_v\}].$$

(e) By the definition of a voting process,

$$S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}](\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* S'$$

where $\phi(S') \equiv \varphi' | \{^a/_{x_1}, {}^c/_{x_2}\}$ for some φ'. Therefore by (d) there exists a substitution σ with $x \in \mathrm{dom}(\sigma)$ such that

$$S[V'' \mid V_B\{^x/_v\}]\sigma(\rightarrow^* \xrightarrow{\alpha} \rightarrow^*)^* S'' \quad (1)$$

where $\phi(S'') \equiv \varphi'' | \{^a/_{x_1}, {}^c/_{x_2}\}$ for some φ''. Since $V''^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$, V does not output c on a channel it shares with $S[\_]$. From equation (1) and the fact that c does not occur in $S[\_]$, it must be that $\sigma(x) = c$ in all evolutions of $S[V'' \mid V_B\{^x/_v\}]\sigma$ preserving the adaptive simulation (d). Therefore,

$$S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V'' \mid V_B\{^c/_v\}]. \quad \square$$

3.4. Fault attacks

Looking closely at the definition of coercion-resistance, one may notice that the intruder could have an unexpected strategy to distinguish the left-hand process from the right-hand process. In protocols in which the voter is expected to submit certain messages to the election administrators, the arrangement between the voter and the coercer may involve the coercer preparing those messages for the voter. In this case, the coercer may submit messages to $V_A\{^c/_v\}^{c_1,c_2}$ with the pure intention to block the process on the left-hand side of the relation. If V' is unable to detect that the message is incoherent, the process will not block on the right-hand side, thus yielding an observable difference. Our definition therefore allows V' to block too and not to vote if V' thinks that the messages sent by the coercer would block the protocol.

In a real-life scenario the intruder may use this method to test the loyalty of V_A in a probabilistic way. For example, in one out of every one hundred coerced voters, the coercer might submit such a garbage message to the voter. If the voter is not genuinely cooperating with him by submitting the garbage message, and instead casts his own vote successfully, the attacker can perceive a difference.

We argue that this attack, if successful, is an attack against coercion-resistance, since it means that the voter knows that the coercer has the ability to detect whether the voter is cooperating with him or not. Launching the attack costs the coercer that particular vote, but it is a means of applying pressure on the voter to cooperate.

Our definition of coercion-resistance is correct with respect to this kind of attack. That is, a protocol which is vulnerable to the attack is not coercion-resistant according to our definition. A protocol is coercion-resistant if V' can be chosen to mimic $V_A\{^c/_v\}^{c_1,c_2}$. Thus, if V' can detect that the message from the coercer is incoherent, she can act in order to block the protocol, preserving the relation.

4. Example

In this section we apply the formalization, described in the previous section to a simplified version of the Lee *et al.* protocol [14]. One of the main advantages of this protocol is that it is *vote and go*: voters need participate in the election only once, in contrast with [9], where all voters have to finish a first phase before any of them can participate in the second phase.

We simplified the protocol in order to concentrate on the aspects that are important with respect to receipt-freeness and coercion-resistance. In particular we do not consider distributed authorities. Nevertheless, the protocol is neither trivial to model nor to analyse and illustrates well subtle issues in coercion-resistance.

4.1. Description of the protocol

The protocol relies on two less usual cryptographic primitives: re-encryption and designated verifier proofs (DVP) of re-encryption. We start by explaining these primitives.

A re-encryption of a ciphertext (obtained using a randomized encryption scheme) changes the random coins, without changing or revealing the plaintext. In the ElGamal scheme for instance, if (x, y) is the ciphertext, this is simply done by computing (xg^r, yh^r), where r is a random number, and g and h are the subgroup generator and the public key respectively. Note that neither the creator of the original ciphertext nor the person re-encrypting knows the random coins used in the re-encrypted ciphertext, for they are a function of the coins chosen by both parties. In particular, a voter cannot reveal the coins to a potential coercer who could use this information to verify the value of the vote, by ciphering his expected vote with these coins.

A DVP of the re-encryption proves that the two ciphertexts contain indeed the same plaintext. However, a designated verifier proof only convinces one intended person, *e.g.*, the voter, that the re-encrypted ciphertext contains the original plaintext. In particular this proof cannot be used to convince the coercer. Technically, this is achieved by giving the designated verifier the ability to simulate the transcripts of the proof.

Our simplified protocol can be described in three steps.

1. The voter encrypts his vote with the collector's public key, signs the encrypted vote and sends it to an administrator on a private channel. The administrator checks whether the voter is a legitimate voter and has not voted yet. Then the administrator *re-encrypts* the given ciphertext, signs it and sends it back to the voter. The administrator also provides a DVP that the two ciphertexts contain indeed the same plaintext. In practice, this first stage of the protocol can be done using a voting booth where eligibility of the voter is tested at the entrance of the booth. The booth contains a tamper-proof device which performs re-encryptions, signatures and DVP proofs.
2. The voter sends (via an anonymous channel) the re-encrypted vote, which has been signed by the administrator to the public board.
3. The collector checks the administrator's signature on each of the votes and, if valid, decrypts the votes and publishes the final results.

4.2. Applied pi calculus model

Cryptographic primitives as an equational theory. The equational theory is represented in Figure 3. The functions and equations that handle public keys, probabilistic encryption and digital signature are as usual. To model re-encryption we add a function $\mathsf{rencrypt}$, that permits us to obtain a different encryption of the same message with another random coin which is function of the original one and the one used during the re-encryption. We also add a pair of functions dvp and $\mathsf{checkdvp}$: dvp permits us to build a *designated verifier proof* of the fact that a message is a re-encryption of another one and $\mathsf{checkdvp}$ allows the designated verifier to check that the proof is valid. Note that $\mathsf{checkdvp}$ also succeeds for a *fake* dvp created using the designated verifier's private key.

Main (Process 1). The main process sets up private channels and specifies how the processes are combined in parallel. Most of the private channels are for key distribution. The private channel cha is a private channel between the administrator and voters. This is motivated by the fact that the administrator corresponds to a tamper-proof hardware device in this protocol. For ease of presentation, we only model the protocol for two voters and launch two copies of the administrator and collector process, one for each voter.

Keying material (Process 2). Our model includes a dedicated process for generating and distributing keying material modeling a PKI. Additionally, this process registers le-

```
equation decrypt(pencrypt(m,pk(sk),r),sk) = m.
equation rencrypt(pencrypt(m,pk(sk),r1),r2) = pencrypt(m,pk(sk),f(r1,r2)).
equation checksign(m,sign(m,sk),pk(sk)) = ok.
equation checkdvp(dvp(x,rencrypt(x,r),r,pkv), x,rencrypt(x,r),pkv) = ok.
equation checkdvp(dvp(x,y,z,skv),x,y,pk(skv)) = ok.
```

Figure 3: Equational theory

```
process
    (* private channels *)
    ν privCh.ν chA.ν pkaCh.ν pkcCh.
    ν skaCh.ν skcCh.ν skvaCh.ν skvbCh
    (* administrators *)
    (processK | processA | processA |
    processC | processC |
    (* voters *)
    (let skvCh=skvaCh in
     let v=a in processV)
    (let skvCh=skvbCh in
     let v=b in processV) )
```

Process 1: Main process

```
let processK=
    (* private key *)
    ν ska. ν skc. ν skv1. ν skv2.
    (* public key *)
    let pka=pk(ska) in
    let pkc=pk(skc) in
    let pkv1=pk(skv1) in
    let pkv2=pk(skv2) in
    out(ch,pka). out(ch,pkc).
    out(ch,pkv1). out(ch,pkv2).
    (* register legitimate voters *)
    (out(privCh,pkv1)|out(privCh,pkv2) |
    (* keys disclosure on priv chan *)
    !out(pkaCh,pka) | !out(pkcCh,pkc) |
    !out(skaCh,ska) | !out(skcCh,skc) |
    out(skvaCh,skv1) | out(skvbCh,skv2))
```

Process 2: Administrator for keying material

```
let processV=
    (* his private key *)
    in(skvCh,skv).
    (* public keys of administrators *)
    in(pkaCh,pubka).
    in(pkcCh,pubkc).
    ν r.
    let e=pencrypt(v,pubkc,r) in
    out(chA,(pk(skv),e,sign(e,skv))).
    in(chA,m2).
    let (re,sa,dvpV)=m2 in
    if checkdvp(dvpV,e,re,pk(skv))=ok
    then if checksign(re,sa,pubka)=ok
    then out(ch,(re,sa))
```

Process 3: Voter process

```
let processA=
    (* administrator's private key *)
    in(skaCh,skadm).
    (* register a legimitate voter *)
    in(privCh,pubkv).
    in(chA,m1).
    let (pubv,enc,sig)=m1 in
    if pubv=pubkv then
    if checksign(enc,sig,pubv)=ok
    then ν r1.
    let reAd=rencrypt(enc,r1) in
    let signAd=sign(reAd,skadm) in
    let dvpAd=dvp(enc,reAd,r1,pubv) in
    out(chA,(reAd,signAd,dvpAd))
```

Process 4: Administrator process

gitimate voters and also distributes the public keys of the election authorities to legitimate voters: this is modeled using restricted channels so that the attacker cannot provide false public keys.

Voter (Process 3). First, each voter obtains his secret key from the PKI as well as the public keys of the election authorities. Then, a fresh random number is generated to encrypt his vote with the public key of the collector. Then, he signs the result and sends it on a private channel to the administrator. If the voter has been correctly registered, he obtains from the administrator, a re-encryption of his vote signed by the administrator together with a designated verifier proof of the fact that this re-encryption has been done correctly. If this proof is correct, then the voter sends his re-encrypted vote signed by the administrator to the collector.

Administrator (Process 4). The administrator first receives through a private channel his own public key as well as the public key of a legitimate voter. The received public key has to match the voter who is trying to get a re-encryption of his vote signed by the administrator. The administrator has also to prove to the voter that he has done the re-encryption properly. For this, he builds a designated verifier proof which will be only convincing for the voter.

Collector (Process 5). To model the collector, we use the phase command, introduced in the ProVerif tool [5], as a

```
let processC=
    (* collector's private key *)
    in(skcCh, privc).
    (* administrator's public key *)
    in(pkaCh, pkadmin).
    in(ch, m3).
    phase 1.
    let (ev, sev)=m3 in
    if checksign(ev, sev, pkadmin)=ok
    then let voteV=decrypt(ev, privc) in
    out(ch, voteV)
```

Process 5: Collector process

global synchronisation command, *i.e.*, all processes finish the instructions of a given phase before any of them can start the following phase. This separation is crucial for receipt-freeness (and hence coercion-resistance) to hold. First, the collector receives all the signed ballots. He checks the signature and decrypts the result with his private key to obtain the value of the vote in order to publish the results.

4.3. Receipt-freeness, coercion-resistance

We do not give full formal proofs here but only the ideas on how to construct the V' processes. The aim is merely to illustrate the definitions. We denote $V_A = V\{^{skvaCh}/_{skvCh}\}$ and $V_B = V\{^{skvbCh}/_{skvCh}\}$.

Receipt-freeness. To show receipt-freeness one needs to construct a process V' which successfully can fake all secrets to a coercer. The idea is for V' to vote a, but when outputting secrets to the coercer V' prepares all outputs as if he was voting c. The crucial part is that, using his private key, he provides a fake dvp stating that the actual re-encryption of the encryption of vote a is a re-encryption of the encryption of vote c. Given our equational theory, the two resulting frames are statically equivalent as for both the real and the fake dvp, checkdvp would give ok.

Process 6 shows a possible V'. To prove receipt-freeness, we need to show

- $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$, and
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \quad \approx_\ell \quad S[V' \mid V_B\{^c/_v\}]$.

The first one may be seen informally by considering V' with the "out(c1,...)" commands removed, and comparing it visually with V_A. To see the second labelled bisimulation, one can informally consider all the executions of each side. S consists of the Main process, and therefore includes process K, the two processA's, and the two processC's, but it has a hole for the two voter processes. As shown above, the hole is filled by $V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}$ on the left and by $V' \mid V_B\{^c/_v\}$ on the right. Executions of $V_A\{^c/_v\}^{chc}$ are matched with those of V'; similarly, $V_B\{^a/_v\}$ on the left

```
let processV'=
    (* his private key *)
    in(skvCh, skv). out(c1, skv).
    (* public keys of administrators *)
    in(pkaCh, pubka). out(c1, pubka).
    in(pkcCh, pubkc). out(c1, pubkc).

    ν r. out(c1, r).
    let e=pencrypt(a, pubkc, r) in
    out(chA, (pk(skv), e, sign(e, skv))).
    in(chA, m2).

    let (re, sa, dvpV)=m2 in
    if checkdvp(dvpV, e, re, pk(skv))=ok
    then (
        let fk=dvp(encrypt(c, pubkc, r),
                   re, r, skv) in
        out(c1, (re, sa, fk)).
        if checksign(re, sa, pubka)=ok
        then out(ch, (re, sa)).
        )
    else out (c1, (re, sa, dvpV))
```

Process 6: Process V' for receipt-freeness

is matched with $V_B\{^c/_v\}$ on the right. The attacker could try to make one side block, but this will make the other one block too. Both left and right sides result in votes for a and c; thus, when fully executed, if one side results in a process whose frame contains the active substitutions $\{^a/_{x_1}, ^c/_{x_2}\}$ then the other one can be executed that way too.

Coercion-resistance. The construction of V' is similar to the one for receipt-freeness. However, for coercion-resistance the coercer also provides the inputs for the messages to send out. If the coercer prepares messages corresponding to a given vote, we fake the outputs as previously and know that the non-coerced voter will counter-balance the outcome, by adaptively choosing the same vote. The process V' and the strict evaluation context C required for the definition of coercion resistance are shown in Processes 7 and 8. Similar reasoning to that used above (for receipt freeness) can be used here, to establish whether the three conditions

- $\nu c_1.\nu c_2.C[V_A\{^c/_v\}^{c_1,c_2}] \approx_\ell V_A\{^c/_v\}^{chc}$,
- $\nu c_1.\nu c_2.C[V']^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}] \preceq_a S[V' \mid V_B\{^x/_v\}]$,

hold for coercion resistance. The first two are straightforward, but reasoning about the third one reveals some subtleties about details of how the protocol is implemented.

The attacker can observe a difference between both sides if he schedules the processes so that $V_A\{^c/_v\}^{c_1,c_2}$ is delayed on the left while $V_B\{^a/_v\}$ is allowed to proceed. If this happens, we cannot find the substitution for x required for the adaptive simulation so that $V_B\{^x/_v\}$ can proceed. The attacker will observe a difference because $V_B\{^a/_v\}$ will send his vote on the left, while $V_B\{^x/_v\}$ will not. To prevent this attack, we can make the voters report their votes along a private channel instead of a public one (last line of Process 3). This means that the protocol could not be used over the internet if one wants to guarantee coercion resistance.

Another way to attack coercion resistance is to use the fault attacks described in section 3.4. Here, the coercer provides a badly formatted input. The voter should detect this and just follow the instructions to avoid the fault attack. Invalid signatures are easy to detect. However, the case where the first encrypted vote sent to the administrator is an invalid encryption is more difficult to handle as V' cannot detect it. Here we can consider several cases depending on the details of the implementation, namely whether decryption is possible on every bitstring. If so (as in our equational theory), then the other voter could counterbalance by choosing x to be the decryption of the given garbage message. At the tallying stage this would indeed result in an invalid vote on both sides. If one considers encryption with integrity checking, which one could model in applied pi calculus by adding an explicit checking equation, then this protocol would not be coercion-resistant. This is because the non-coerced is only allowed to choose the value of its vote, but in other respects it follows the protocol and in particular encrypts the chosen vote correctly. Therefore it cannot mimick the coerced voter who sends an invalid vote. Thus the collector blocks when trying to decrypt the vote for the coerced voter, but not for V_B, resulting in an observable difference.

5. Conclusion

In this paper we studied two particular anonymity properties of election protocols: receipt-freeness and coercion-resistance. The peculiar thing about these properties is that the voter wants to break his anonymity by proving to a coercer that he voted in a particular way. Both receipt-freeness and coercion-resistance assure that this is not possible. We have modelled the properties in the applied pi calculus. Receipt-freeness can be modelled using standard labelled bisimulation, while coercion-resistance needs a specific, new simulation relation. Although the properties are modelled using these different relations, we can prove that, according to the intuition, coercion-resistance implies receipt-freeness which itself implies privacy. Finally we illustrate the definitions on a simplified version of the Lee *et al.* voting protocol.

As future work we would like to automate the verification of observational equivalence. Although the ProVerif

```
let processV'=
    (* his private key *)
    in(skvCh,skv). out(c1,skv).
    (* public keys of administrators *)
    in(pkaCh,pubka). out(c1,pubka).
    in(pkcCh,pubkc). out(c1,pubkc).

    ν r. out(c1,r).
    let e=pencrypt(a,pubkc,r) in
    in(c2,x1).
    let (pi,ei,si)=x1 in
    if pi=pk(skv) then (
    if si=sign(ei,skv) then
        out(chA,(pk(skv),e,
              sign(e,skv))).
        else out(chA,x1).
    )
     else out(chA,x1).

    in(chA,m2).
    let (re,sa,dvpV)=m2 in
    if checkdvp(dvpV,e,re,pk(skv))=ok
    then (
     let fk=dvp(ei,re,r,skv) in
     out(c1,(re,sa,fk)).
     if checksign(re,sa,pubka)=ok
     then in(c2,x2). out(ch,x2).
         )
    else  out(c1,(re,sa,dvpV))
```

Process 7: Process V' for coercion-resistance

```
let contextC[]=(
    (* private key of V *)
    in(c1,x1). out(chc,x1).
    (* public key of A *)
    in(c1,x2). out(chc,x2).
    (* public key of C *)
    in(c1,x3). out(chc,x3).
    (* nonce of V *)
    in(c1,x4). out(chc,x4).

    let e=pencrypt(c,x3,x4) in
    out(c2,(pk(x1),e,sign(e,x1))).
    (* dvp *)
    in(c1,x5). out(chc,x5).
    let (re,sa,dvp)=x5 in
    if checkdvp(dvp,e,re,pk(x1))=ok
    then if checksign(re,sa,pubka)=ok
    then out(c2,(re,sa))
    ) | _
```

Process 8: Context C for coercion resistance tool can in many special cases prove observational equivalence, it is not able to do so for privacy or the more elaborated properties of this paper. We foresee to investigate automatic verification of observational equivalence at least for a finite number of sessions (not authorizing replication) and for restricted classes of equational theories.

References

[1] M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. In *Proc. 13th European Symposium On Programming (ESOP'04)*, volume 2986 of *LNCS*, pages 340–354, Barcelona, Spain, 2004. Springer.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symposium on Principles of Programming Languages*, pages 104–115, London, UK, 2001. ACM.

[3] J. Benaloh. *Verifiable Secret Ballot Elections*. PhD thesis, Yale University, 1987.

[4] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th Annual Symposium on Theory of Computing (STOC'94)*, pages 544–553, Montréal, Québec, 1994. ACM Press.

[5] B. Blanchet. *ProVerif: Automatic Cryptographic Protocol Verifier User Manual*, 2005.

[6] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In *Proc. of Advances in Cryptology (Eurocrypt'88)*, volume 330 of *LNCS*, pages 177–182. Springer, 1988.

[7] D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *Proc. 10th European Symposium On Research In Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 118–139, Milan, Italy, 2005. Springer.

[8] C. Fournet and M. Abadi. Hiding names: Private authentication in the applied pi calculus. In *Proc. of Int. Symposium on Software Security (ISSS'02)*, volume 2609 of *LNCS*, pages 317–338, Tokyo, Japan, 2003. Springer.

[9] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Proc. of Advances in Cryptology (AUSCRYPT'92)*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[10] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. of Advances in Cryptography (Eurocrypt'00)*, volume 1807 of *LNCS*, pages 539–556, Bruges, Belgium, 2000. Springer.

[11] W.-S. Juang and C.-L. Lei. A secure and practical electronic voting scheme for real world environments. *IEICE Transaction on Fundamentals of Electronics, Communications and Computer Science*, 1:64–71, 1997.

[12] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proc. of Workshop on Privacy in the Electronic Society (WPES'05)*, Alexandria, USA, 2005. ACM Press.

[13] S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proc. 14th European Symposium On Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200, Edinburgh, U.K., 2005. Springer.

[14] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proc. of Information Security and Cryptology (ICISC'03)*, volume 2971 of *LNCS*, pages 245–258, Seoul, Korea, 2004. Springer.

[15] T. Okamoto. An electronic voting scheme. In *IFIP World Conference on IT Tools*, pages 21–30, Canberra, Australia, 1996.

[16] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *LNCS*, pages 25–35, Paris, France, 1997. Springer.

[17] S. Schneider and A. Sidiropoulos. CSP and anonymity. In *Proc. 4th European Symposium On Research In Computer Security (ESORICS'96)*, volume 1146 of *LNCS*, pages 198–218, Roma, Italy, 1996. Springer.

A. Additional proofs

Lemma 1 *Let C be an evaluation context. There exist a sequence $\tilde{u}$ of names and variables and an extended process B such that for all extended process A, we have $C[A] \equiv \nu\tilde{u}.(A \mid B)$.*

Proof. The proof is done by induction on the structure of C. The base case where $C[\_] = \_$ is trivial as $C \equiv \nu\tilde{u}.(\_ \mid B)$ where $\tilde{u}$ is empty and $B = 0$. The possibilities for building $C[\_]$ are the following:

- $C = C_1 \mid B'$ where C_1 is an evaluation context and B' is an extended process. We know that there exist a sequence $\tilde{u}_1$ of names and variables and an extended process B_1 such that for any extended process A we have that $C_1[A] = \nu\tilde{u}_1.(A \mid B_1)$. We can also assume w.l.o.g. that $\tilde{u}_1 \notin fv(B') \cup fn(B')$. Let $B = B' \mid B_1$ and $\tilde{u} = \tilde{u}_1$. Let A be an extended process. We have that

$$\begin{aligned} C[A] &\equiv C_1[A] \mid B' \\ &\equiv \nu\tilde{u}_1.(A \mid B_1) \mid B' \\ &\equiv \nu\tilde{u}.(A \mid (B_1 \mid B')) \quad \text{since } \tilde{u} \notin fv(B') \cup fn(B') \\ &\equiv \nu\tilde{u}.(A \mid B) \end{aligned}$$

- $C = \nu v.C_1$ where C_1 is is an evaluation context and v a name or a variable. We know that there exist a sequence $\tilde{u}_1$ of names and variables and an extended process B_1 as above. Let $B = B_1$ and $\tilde{u} = \tilde{u}_1 \cup v$. Let A be an extended process. We have that

$$\begin{aligned} C[A] &\equiv \nu v.C_1[A] \\ &\equiv \nu v.(\nu\tilde{u}_1.(A \mid B_1)) \quad \text{since } \tilde{u}_1 \notin fv(A) \cup fn(A) \\ &\equiv \nu\tilde{u}.(A \mid B) \end{aligned}$$
□

Lemma 2 *Let C_1, C_2 be two evaluation contexts. For any extended process A, we have $C_1[C_2[A]] \equiv C_2[C_1[A]]$.*

Proof. By Lemma 1, we know that there exist two sequences $\tilde{u}_1, \tilde{u}_2$ of names and variables, B_1, B_2 two closed extended processes such that for any extended process A, we have

1. $C_1[A] \equiv \nu\tilde{u}_1.(A \mid B_1)$;
2. $C_2[A] \equiv \nu\tilde{u}_2.(A \mid B_2)$.

We assume also w.l.o.g. that $\tilde{u}_2 \notin fv(B_1) \cup fn(B_1)$ and $\tilde{u}_1 \notin fv(B_2) \cup fn(B_2)$. Let A be an extended process.

$$\begin{aligned} & C_1[C_2[A]] \\ \equiv\ & C_1[\nu\tilde{u}_2(A \mid B_2)] \\ \equiv\ & \nu\tilde{u}_1.(\nu\tilde{u}_2.(A \mid B_2) \mid B_1) \\ \equiv\ & \nu\tilde{u}_2.\nu\tilde{u}_1.((A \mid B_1) \mid B_2) \text{ since } \tilde{u}_2 \notin fv(B_1) \cup fn(B_1) \\ \equiv\ & \nu\tilde{u}_2.(\nu\tilde{u}_1.(A \mid B_1) \mid B_2) \text{ since } \tilde{u}_1 \notin fv(B_2) \cup fn(B_2) \\ \equiv\ & \nu\tilde{u}_2.(C_1[A] \mid B_2) \\ \equiv\ & C_2[C_1[A]] \end{aligned}$$
□

Lemma 3 *Let A be an extended process, C an evaluation context and ch a channel name. If $ch \notin fn(C[0])$ we have*

$$C[A]^{\backslash out(ch,\cdot)} \equiv C[A^{\backslash out(ch,\cdot)}].$$

Proof. By Lemma 1, we know that there exist a sequence $\tilde{u}$ of names and variables and an extended process B such that $C[\_] \equiv \nu\tilde{u}.(\_ \mid B)$. We can assume w.l.o.g. that $ch \notin \tilde{u}$. As $ch \notin fn(C[0])$ we have that $ch \notin fn(B)$. Hence, we have

$$\begin{aligned} & C[A]^{\backslash out(ch,\cdot)} \\ \widehat{=}\ & \nu ch.(C[A] \mid !\mathsf{in}(ch,x)) \\ \equiv\ & \nu ch.(\nu\tilde{u}.(A \mid B) \mid !\mathsf{in}(ch,x)) \\ \equiv\ & \nu\tilde{u}.\nu ch.(A \mid B \mid !\mathsf{in}(ch,x)) && \text{since } ch \notin \tilde{u} \\ \equiv\ & \nu\tilde{u}.(\nu ch.(A \mid !\mathsf{in}(ch,x)) \mid B) && \text{since } ch \notin fn(B) \\ \widehat{=}\ & \nu\tilde{u}.(A^{\backslash out(ch,\cdot)} \mid B) \\ \equiv\ & C[A^{\backslash out(ch,\cdot)}] \end{aligned}$$
□

Lemma 4 *Let P be a closed plain process and ch a channel name such that $ch \notin fn(P) \cup bn(P)$. We have*

$$(P^{ch})^{\backslash out(ch,\cdot)} \approx_\ell P.$$

Proof. Let P be a plain process (not necessarily closed). We show by induction on P that for any substitution ρ closing P, we have $(P\rho)^{ch\backslash out(ch,\cdot)} \approx_\ell P\rho$. The base case where $P = 0$ is trivial. Hence for any substitution ρ, we have $0\rho = 0$. Hence $(0\rho)^{ch\backslash out(ch,.)} \ \widehat{=}\ \nu ch.((0\rho)^{ch} \mid !\mathsf{in}(ch,x)) \approx_\ell 0\rho$.

Let ρ be a substitution closing P. The possibilities for building P are the following:

- $P = P_1 \mid P_2$. In such a case, we have:

$$\begin{aligned} & P\rho^{ch\backslash out(ch,.)} \\ \widehat{=}\ & (P_1\rho^{ch} \mid P_2\rho^{ch})^{\backslash out(ch,.)} \\ \widehat{=}\ & \nu ch.(P_1\rho^{ch} \mid P_2\rho^{ch} \mid !\mathsf{in}(ch,x)) \\ \approx_\ell\ & \nu ch.(P_1\rho^{ch} \mid !\mathsf{in}(ch,x)) \mid \nu ch.((P_2\rho)^{ch} \mid !\mathsf{in}(ch,x)) \\ & \text{because } \mathsf{in}(ch,.) \text{ occurs neither in } P_1\rho^{ch} \text{ nor in } P_2\rho^{ch}. \\ \approx_\ell\ & (P_1\rho)^{ch\backslash out(ch,.)} \mid (P_2\rho)^{ch\backslash out(ch,.)} \\ \approx_\ell\ & P_1\rho \mid P_2\rho && \text{by induction hypothesis} \\ =\ & P\rho \end{aligned}$$

- $P = \nu n.P_1$. We have:

$$\begin{aligned} & P\rho^{ch\backslash out(ch,.)} \\ =\ & (\nu u.P_1\rho)^{ch\backslash out(ch,.)} \\ \widehat{=}\ & \nu ch.(\nu u.\mathsf{out}(ch,u).P_1\rho^{ch} \mid !\mathsf{in}(ch,x)) \\ \approx_\ell\ & \nu ch.(\nu u.P_1\rho^{ch} \mid !\mathsf{in}(ch,x)) \\ \equiv\ & \nu u.\nu ch.(P_1\rho^{ch} \mid !\mathsf{in}(ch,x)) && \text{since } u \neq ch \\ \widehat{=}\ & \nu u.P_1\rho^{ch\backslash out(ch,.)} \\ \approx_\ell\ & \nu u.P_1\rho && \text{by induction hypothesis} \\ =\ & P\rho \end{aligned}$$

- $P = \mathsf{in}(u,y).P_1$. Note that $u\rho \neq ch$. We have:

$P\rho^{ch\backslash out(ch,.)}$
$= (\mathsf{in}(u,y).P_1)^{ch\backslash out(ch,.)}$
$\hat{=} \nu ch.(\mathsf{in}(u\rho,y).\mathsf{out}(ch,y).P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\approx_\ell \mathsf{in}(u\rho,y).\nu ch.(\mathsf{out}(ch,y).P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\approx_\ell \mathsf{in}(u\rho,y).\nu ch.(P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\hat{=} \mathsf{in}(u\rho,y).P_1\rho^{ch\backslash out(ch,.)}$ by induction hypothesis
$\approx_\ell \mathsf{in}(u\rho,y).P_1\rho$
$= P\rho$

- $P = \mathsf{out}(u,M).P_1$. Note that $u\rho \neq ch$. We have:

$P\rho^{ch\backslash out(ch,.)}$
$= (\mathsf{out}(u\rho,y).P_1\rho)^{ch\backslash out(ch,.)}$
$\hat{=} \nu ch.(\mathsf{out}(u\rho,y).P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\approx_\ell \mathsf{out}(u\rho,y).\nu ch.(P_1^{ch} \mid !\mathsf{in}(ch,x))$
$\hat{=} \mathsf{out}(u\rho,y).P_1\rho^{ch\backslash out(ch,.)}$
$\approx_\ell \mathsf{out}(u\rho,y).P_1\rho$ by induction hypothesis
$= P\rho$

- $P = !P_1$. In such a case, we have:

$P\rho^{ch\backslash out(ch,.)}$
$\hat{=} (!P_1\rho)^{ch\backslash out(ch,.)}$
$\hat{=} \nu ch.(!P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\approx_\ell \nu ch.!(P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\approx_\ell !(\nu ch.(P_1\rho^{ch} \mid !\mathsf{in}(ch,x)))$
because $\mathsf{in}(ch,.)$ does not occur in $P_1\rho^{ch}$.
$\hat{=} !P_1\rho^{ch\backslash out(ch,.)}$
$\approx_\ell !P_1\rho$ by induction hypothesis
$= P\rho$

- $P =$ if $M = N$ then P_1 else P_2.Hence, we have:

$P\rho^{ch\backslash out(ch,.)}$
$= (\text{if } M\rho = N\rho \text{ then } P_1\rho \text{ else } P_2\rho)^{ch\backslash out(ch,.)}$
$\hat{=} \nu ch.(\text{if } M\rho = N\rho \text{ then } P_1\rho^{ch} \text{ else } P_2\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\approx_\ell \nu ch.(\text{if } M\rho = N\rho \text{ then } (P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\quad \text{else } (P_2\rho^{ch} \mid !\mathsf{in}(ch,x)))$
$\approx_\ell \nu ch.(\text{if } M\rho = N\rho \text{ then } (P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\quad \text{else } (P_2\rho^{ch} \mid !\mathsf{in}(ch,x)))$
$\approx_\ell \text{if } M\rho = N\rho \text{ then } \nu ch.(P_1\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\quad \text{else } \nu ch.(P_2\rho^{ch} \mid !\mathsf{in}(ch,x))$
$\hat{=} \text{if } M\rho = N\rho \text{ then } P_1\rho^{ch\backslash out(ch,.)} \text{ else } P_2\rho^{ch\backslash out(ch,.)}$
$\approx_\ell \text{if } M\rho = N\rho \text{ then } P_1\rho \text{ else } P_2\rho$
$= P\rho$ □

Lemma 6 *Let A, B be two extended processes and C a strict evaluation context, such that $fv(C[0]) = \emptyset$. We have*

$$A \preceq_a B \implies C[A] \preceq_a C[B].$$

Proof. We show this result by induction on C. The base case ($C = \_$) is trivial. Then, we need to show:

1. $A \preceq_a B \implies \nu n.A \preceq_a \nu n.B$ for any name n,
2. $A \preceq_a B \implies A \mid D \preceq_a B \mid D$ for any closed extended process D.

First case: $A \preceq_a B \implies \nu n.A \preceq_a \nu n.B$.
By hypothesis, we know that $A \preceq_a B$. Let $\mathcal{R}$ be a relation witnessing this adpative simulation. We define a relation $\mathcal{R}'$ in the following way:

$A \;\mathcal{R}'\; B$ if and only if $A \;\mathcal{R}\; B$ or there exist two extended processes $\tilde{A}$ and $\tilde{B}$ such that $A \equiv \nu n.\tilde{A}$, $B \equiv \nu n.\tilde{B}$ and $\tilde{A} \,\mathcal{R}\, \tilde{B}$.

We now have to show that $\mathcal{R}'$ is a relation witnessing $\nu n.A \preceq_a \nu n.B$. Firstly, we need to show that $\nu n.A \;\mathcal{R}'\; \nu n.B$. This follows from the fact that $A \;\mathcal{R}\; B$ and by definition of $\mathcal{R}'$. Secondly, we show that $\mathcal{R}'$ is an adaptive simulation, *i.e.* $\mathcal{R}'$ is a relation satisfying the three points of Definition 12.

Let A and B be two extended processes such that $A \;\mathcal{R}'\; B$. Either we have $A \;\mathcal{R}\; B$ and in such a case, we easily conclude since $\mathcal{R}$ has the three required properties. Or, there exist two extended processes $\tilde{A}$ and $\tilde{B}$ such that $A \equiv \nu n.\tilde{A}$, $B \equiv \nu n.\tilde{B}$ and $\tilde{A} \;\mathcal{R}\; \tilde{B}$.

1. (static equivalence) We need to show that for all σ_A there exists σ_B such that $\phi(A\sigma_A) \approx_s \phi(B\sigma_B)$, *i.e.* for all σ_A there exists σ_B such that $\phi(\nu n.\tilde{A}\sigma_A) \approx_s \phi(\nu n.\tilde{B}\sigma_B)$. This is obvious since we know that for all σ_A there exists σ_B such that $\phi(\tilde{A}\sigma_A) \approx_s \phi(\tilde{B}\sigma_B)$ by the fact that $A \;\mathcal{R}\; B$.
2. (internal reduction) Let σ_A be a substitution such that $\nu n.\tilde{A}\sigma_A \to A'\sigma_A$. We need to show that there exists σ_B and B' such that $\nu n.\tilde{B}\sigma_B \xrightarrow{*} B'\sigma_B$ and $A'\theta_A \;\mathcal{R}'\; B'\theta_B$ where $\theta_A = \sigma_A|_{X_{\nu n.\tilde{A}}(\nu n.\tilde{A}\sigma_A \to A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \xrightarrow{*} B'\sigma_B)}$. By case analysis on the reduction step, we can show that $A' \equiv \nu n.\tilde{A}'$ for some extended process $\tilde{A}'$ and that $\tilde{A}\sigma_A \to \tilde{A}'\sigma_A$. Since $\tilde{A} \;\mathcal{R}\; \tilde{B}$, there exist $\tilde{B}'$ and $\tilde{\sigma}_B$ such that $\tilde{B}\tilde{\sigma}_B \xrightarrow{*} \tilde{B}'\tilde{\sigma}_B$ and $\tilde{A}'\tilde{\theta}_A \;\mathcal{R}\; \tilde{B}'\tilde{\theta}_B$ where $\tilde{\theta}_A = \sigma_A|_{X_{\tilde{A}}(\tilde{A}\sigma_A \to \tilde{A}'\sigma_A)}$ and $\tilde{\theta}_B = \tilde{\sigma}_B|_{X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \xrightarrow{*} \tilde{B}'\tilde{\sigma}_B)}$. Hence we have, $\nu n.\tilde{B}\tilde{\sigma}_B \xrightarrow{*} \nu n.\tilde{B}'\tilde{\sigma}_B$ and $\nu n.\tilde{A}'\tilde{\theta}_A \;\mathcal{R}'\; \nu n.\tilde{B}'\tilde{\theta}_B$. Let $\sigma_B = \tilde{\sigma}_B$ and $B' = \nu n.\tilde{B}'$. As $X_{\nu n.\tilde{A}}(\nu n.\tilde{A}\sigma_A \to A'\sigma_A) = X_{\tilde{A}}(\tilde{A}\sigma_A \to \tilde{A}'\sigma_A)$, we have $\theta_A = \tilde{\theta}_A$. As $X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \xrightarrow{*} B'\sigma_B) = X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \xrightarrow{*} \tilde{B}'\tilde{\sigma}_B)$, we have $\theta_B = \tilde{\theta}_B$. Hence, we have $\nu n.\tilde{B}\sigma_B \xrightarrow{*} B'\sigma_B$ and $A'\theta_A \;\mathcal{R}'\; B'\theta_B$.

   We need also to show that for all B' such that $\nu n.\tilde{B}\sigma_B \to B'\sigma_B$ there exists A' such that $\nu n.\tilde{A}\sigma_A \xrightarrow{*} A'\sigma_A$ and $A'\theta'_A \;\mathcal{R}'\; B'\theta'_B$ where $\theta'_A = \sigma_A|_{X_{\nu n.A}(\nu n.\tilde{A}\sigma_A \xrightarrow{*} A'\sigma_A)}$ and $\theta'_B = \sigma_B|_{X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \to B'\sigma_B)}$. By case analysis on the reduction step, we can show that $B' \equiv \nu n.\tilde{B}'$ for some extended process $\tilde{B}'$ and that $\tilde{B}\sigma_B \to \tilde{B}'\sigma_B$.

Since $\tilde{A} \ \mathcal{R} \ \tilde{B}$, there exist $\tilde{A}'$ such that $\tilde{A}\sigma_A \xrightarrow{*} \tilde{A}'\sigma_A$ and $\tilde{A}'\theta'_A \ \mathcal{R} \ \tilde{B}'\theta'_B$ where $\tilde{\theta}'_A = \sigma_A|_{X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{*} \tilde{A}'\sigma_A)}$ and $\tilde{\theta}'_B = \sigma_B|_{X_{\tilde{B}}(\tilde{B}\sigma_B \rightarrow \tilde{B}'\sigma_B)}$.
Hence we have, $\nu n.\tilde{A}\sigma_A \xrightarrow{*} \nu n.\tilde{A}'\sigma_A$. Let $A' = \nu n.\tilde{A}'$. We have $\nu n.\tilde{A}\sigma_A \xrightarrow{*} A'\sigma_A$. As $X_{\nu n.A}(\nu n.\tilde{A}\sigma_A \xrightarrow{*} A'\sigma_A) = X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{*} \tilde{A}'\sigma_A)$, we have $\theta'_A = \tilde{\theta}'_A$. As $X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \rightarrow B'\sigma_B) = X_{\tilde{B}}(\tilde{B}\sigma_B \rightarrow \tilde{B}'\sigma_B)$, we have $\theta'_B = \tilde{\theta}'_B$. Hence, we have $A'\theta'_A \ \mathcal{R}' \ B'\theta'_B$.

3. (labeled transition) Let σ_A be a substitution such that $\nu n.\tilde{A}\sigma_A \xrightarrow{\alpha} A'\sigma_A$. We need to show that there exists σ_B and B' such that $\nu n.\tilde{B}\sigma_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B$ and $A'\theta_A \ \mathcal{R}' \ B'\theta_B$ where $\theta_A = \sigma_A|_{X_{\nu n.\tilde{A}}(\nu n.\tilde{A}\sigma_A \xrightarrow{\alpha} A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B)}$. By case analysis on the labeled transition $\xrightarrow{\alpha}$, we can show that we are in one of the following cases:
   (a) $\nu n.\tilde{A}\sigma_A \xrightarrow{\alpha} \nu n.\tilde{A}'\sigma_A$ and n does not occur in α. In such a case, we have $\tilde{A}\sigma_A \xrightarrow{\alpha} \tilde{A}'\sigma_A$ and $A'\sigma_A \equiv \nu n.\tilde{A}'\sigma_A$,
   (b) $\nu n.\tilde{A}\sigma_A \xrightarrow{\nu n.out(a,n)} A'\sigma_A$ with $n \neq a$ and we have $\tilde{A}\sigma_A \xrightarrow{out(a,n)} A'\sigma_A$.

   Firstly, we deal with the case (a). Since $\tilde{A} \ \mathcal{R} \ \tilde{B}$, there exist $\tilde{B}'$ and $\tilde{\sigma}_B$ such that $\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B$ and $\tilde{A}'\tilde{\theta}_A \ \mathcal{R} \ \tilde{B}'\tilde{\theta}_B$ where $\tilde{\theta}_A = \sigma_A|_{X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{\alpha} \tilde{A}'\sigma_A)}$ and $\tilde{\theta}_B = \tilde{\sigma}_B|_{X_{\tilde{B}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)}}$. Hence we have, $\nu n.\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \nu n.\tilde{B}'\tilde{\sigma}_B$ and $\nu n.\tilde{A}'\tilde{\theta}_A \ \mathcal{R}' \ \nu n.\tilde{B}'\tilde{\theta}_B$. Let $\sigma_B = \tilde{\sigma}_B$ and $B' = \nu n.\tilde{B}'$. As $X_{\nu n.\tilde{A}}(\nu n.\tilde{A}\sigma_A \xrightarrow{\alpha} A'\sigma_A) = X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{\alpha} \tilde{A}'\sigma_A)$, we have $\theta_A = \tilde{\theta}_A$. As $X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B) = X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)$, we have $\theta_B = \tilde{\theta}_B$. We have $\nu n.\tilde{B}\sigma_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B$ and $A'\theta_A \ \mathcal{R}' \ B'\theta_B$.

   Now, we consider the case (b). Since $\tilde{A} \ \mathcal{R} \ \tilde{B}$, we have that there exist $\tilde{B}'$ and $\tilde{\sigma}_B$ such that $\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{out(a,n)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B$ and $A'\tilde{\theta}_A \ \mathcal{R} \ \tilde{B}'\tilde{\theta}_B$ where $\tilde{\theta}_A = \sigma_A|_{X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{out(a,n)} A'\sigma_A)}$ and $\tilde{\theta}_B = \tilde{\sigma}_B|_{X_{\tilde{B}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{out(a,n)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)}}$. Hence we have, $\nu n.\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\nu n.out(a,n)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B$ and $A'\tilde{\theta}_A \ \mathcal{R}' \ \tilde{B}'\tilde{\theta}_B$. Let $\sigma_B = \tilde{\sigma}_B$ and $B' = \tilde{B}'$. As $X_{\nu n.\tilde{A}}(\nu n.\tilde{A}\sigma_A \xrightarrow{\nu n.out(a,n)} A'\sigma_A) = X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{out(a,n)} A'\sigma_A)$, we have $\theta_A = \tilde{\theta}_A$. As $X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \rightarrow^* \xrightarrow{\nu n.out(a,n)} \rightarrow^* B'\sigma_B) = X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{out(a,n)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)$, we have $\theta_B = \tilde{\theta}_B$. We have $\nu n.\tilde{B}\sigma_B \rightarrow^* \xrightarrow{\nu n.out(a,n)} \rightarrow^* B'\sigma_B$ and $A'\theta_A \ \mathcal{R}' \ B'\theta_B$.

   We need also to show that for all B' such that $\nu n.\tilde{B}\sigma_B \xrightarrow{\alpha} B'\sigma_B$ there exists A' such that $\nu n.\tilde{A}\sigma_A \rightarrow^* \xrightarrow{\alpha} \rightarrow^* A'\sigma_A$ and $A'\theta'_A \ \mathcal{R}' \ B'\theta'_B$ where $\theta'_A = \sigma_A|_{X_{\nu n.\tilde{A}}(\nu n.\tilde{A}\sigma_A \rightarrow^* \xrightarrow{\alpha} \rightarrow^* A'\sigma_A)}$ and $\theta'_B = \sigma_B|_{X_{\nu n.\tilde{B}}(\nu n.\tilde{B}\sigma_B \xrightarrow{\alpha} B'\sigma_B)}$. To do this, we distinguish two cases depending on the labeled transition α and we conclude as above.

Second case: $A \preceq_a B \implies A \mid D \preceq_a B \mid D$.
By hypothesis, we know that $A \preceq_a B$. Let $\mathcal{R}$ be a relation witnessing this adpative simulation. We define a relation $\mathcal{R}'$ in the following way:

$A \ \mathcal{R}' \ B$ if and only if there exist two extended processes $\tilde{A}$, $\tilde{B}$ and a closed extended process $\tilde{D}$ such that $A \equiv \tilde{A} \mid \tilde{D}$, $B \equiv \tilde{B} \mid \tilde{D}$ and $\tilde{A} \ \mathcal{R} \ \tilde{B}$.

We now have to show that $\mathcal{R}'$ is a relation witnessing $A \mid D \preceq_a B \mid D$. Firstly, we need to show that $A \mid D \ \mathcal{R}' \ B \mid D$. This follows from the fact that $A \ \mathcal{R} \ B$ and by the definition of $\mathcal{R}'$. Secondly, we show that $\mathcal{R}'$ is an adaptive simulation, *i.e.* $\mathcal{R}'$ is a relation satisfying the three points of Definition 12.

Let A and B be two extended processes such that $A \ \mathcal{R}' \ B$. By definition of $\mathcal{R}'$, there exist two extended processes $\tilde{A}$, $\tilde{B}$ and a closed extended process $\tilde{D}$ such that $A \equiv \tilde{A} \mid \tilde{D}$, $B \equiv \tilde{B} \mid \tilde{D}$ and $\tilde{A} \ \mathcal{R} \ \tilde{B}$.

1. (static equivalence) We need to show that for all σ_A there exists σ_B such that $\phi(A\sigma_A) \approx_s \phi(B\sigma_B)$, *i.e.* for all σ_A there exists σ_B such that $\phi(\tilde{A}\sigma_A \mid \tilde{D}) \approx_s \phi(\tilde{B}\sigma_B \mid \tilde{D})$. This is due to the fact that $\phi(\tilde{A}\sigma_A) \approx_s \phi(\tilde{B}\sigma_B)$ since $A \ \mathcal{R} \ B$.
2. (labeled transition) Let σ_A be a substitution such that $\tilde{A}\sigma_A \mid \tilde{D} \xrightarrow{\alpha} A'\sigma_A$. We need to show that there exists σ_B and B' such that $\tilde{B}\sigma_B \mid \tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B$ and $A'\theta_A \ \mathcal{R}' \ B'\theta_B$ where $\theta_A = \sigma_A|_{X_{\tilde{A}|\tilde{D}}(\tilde{A}\sigma_A|\tilde{D} \xrightarrow{\alpha} A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_{\tilde{B}|\tilde{D}}(\tilde{B}\sigma_B|\tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B)}$. By case analysis on the labeled transition $\xrightarrow{\alpha}$, we can show that we are in one of the following cases:
   (a) $\tilde{A}\sigma_A \mid \tilde{D} \xrightarrow{\alpha} \tilde{A}'\sigma_A \mid \tilde{D}$ and we have $\tilde{A}\sigma_A \xrightarrow{\alpha} \tilde{A}'\sigma_A$ and $A'\sigma_A \equiv \tilde{A}'\sigma_A \mid \tilde{D}$,
   (b) $\tilde{A}\sigma_A \mid \tilde{D} \xrightarrow{\alpha} \tilde{A}\sigma_A \mid \tilde{D}'$ and we have $\tilde{D} \xrightarrow{\alpha} \tilde{D}'$ and $A'\sigma_A \equiv \tilde{A}\sigma_A \mid \tilde{D}'$.

   Firstly, we deal with the case (a). Since $\tilde{A} \ \mathcal{R} \ \tilde{B}$, there exist $\tilde{B}'$ and $\tilde{\sigma}_B$ such that $\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B$ and $\tilde{A}'\tilde{\theta}_A \ \mathcal{R} \ \tilde{B}'\tilde{\theta}_B$ where $\tilde{\theta}_A = \sigma_A|_{X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{\alpha} \tilde{A}'\sigma_A)}$ and $\tilde{\theta}_B = \tilde{\sigma}_B|_{X_{\tilde{B}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)}}$. Hence we have, $\tilde{B}\tilde{\sigma}_B \mid \tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B \mid \tilde{D}$ and $\tilde{A}'\tilde{\theta}_A \mid \tilde{D}' \ \mathcal{R}' \ \tilde{B}'\tilde{\theta}_B \mid \tilde{D}'$. Let $\sigma_B = \tilde{\sigma}_B$ and

$B' = \tilde{B}' \mid \tilde{D}'$. As $X_{\tilde{A}|\tilde{D}}(\tilde{A}\sigma_A \mid \tilde{D} \xrightarrow{\alpha} A'\sigma_A) = X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{\alpha} \tilde{A}'\sigma_A)$, we have $\theta_A = \tilde{\theta}_A$. As $X_{\tilde{B}|\tilde{D}}(\tilde{B}\sigma_B \mid \tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B) = X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)$, we have $\theta_B = \tilde{\theta}_B$. We have $\tilde{B}\sigma_B \mid \tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'\sigma_B$ and $A'\theta_A \ \mathcal{R}' \ B'\theta_B$.

Now, we consider the case (b). We have $\tilde{B}\sigma_B \mid \tilde{D} \xrightarrow{\alpha} \tilde{B}\sigma_B \mid \tilde{D}'$ for any σ_B. Since $\tilde{A} \ \mathcal{R} \ \tilde{B}$, we have $\tilde{A} \mid \tilde{D}' \ \mathcal{R}' \ \tilde{B} \mid \tilde{D}'$.

We need also to show that for all B' such that $\tilde{B}\sigma_B \mid \tilde{D} \xrightarrow{\alpha} B'\sigma_B$ there exists A' such that $\tilde{A}\sigma_A \mid \tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* A'\sigma_A$ and $A'\theta'_A \ \mathcal{R}' \ B'\theta'_B$ where $\theta'_A = \sigma_A|_{X_{\tilde{A}|\tilde{D}}(\tilde{A}\sigma_A|\tilde{D} \rightarrow^* \xrightarrow{\alpha} \rightarrow^* A'\sigma_A)}$ and $\theta'_B = \sigma_B|_{X_{\tilde{B}|\tilde{D}}(\tilde{B}\sigma_B|\tilde{D} \xrightarrow{\alpha} B'\sigma_B)}$.
We need also to distinguish two cases depending on the fact that the labeled transition allows the $\tilde{B}\sigma_B$ or $\tilde{D}$ to evolve and we conclude as above.

3. (internal reduction) Let σ_A be a substitution such that $\tilde{A}\sigma_A \mid \tilde{D} \rightarrow A'\sigma_A$. We need to show that there exists σ_B and B' such that $\tilde{B}\sigma_B \mid \tilde{D} \xrightarrow{*} B'\sigma_B$ and $A'\theta_A \ \mathcal{R}' \ B'\theta_B$ where $\theta_A = \sigma_A|_{X_{\tilde{A}|\tilde{D}}(\tilde{A}\sigma_A|\tilde{D} \rightarrow A'\sigma_A)}$ and $\theta_B = \sigma_B|_{X_{\tilde{B}|\tilde{D}}(\tilde{B}\sigma_B|\tilde{D} \xrightarrow{*} B'\sigma_B)}$. By case analysis on the internal reduction, we can show that we are in one of the following cases:
   (a) $\tilde{A}\sigma_A \mid \tilde{D} \rightarrow \tilde{A}'\sigma_A \mid \tilde{D}$ and we have $\tilde{A}\sigma_A \rightarrow \tilde{A}'\sigma_A$ and $A'\sigma_A \equiv \tilde{A}'\sigma_A \mid \tilde{D}$,
   (b) $\tilde{A}\sigma_A \mid \tilde{D} \rightarrow \tilde{A}\sigma_A \mid \tilde{D}'$ and we have $\tilde{D} \rightarrow \tilde{D}'$ and $A'\sigma_A \equiv \tilde{A}\sigma_A \mid \tilde{D}'$,
   (c) (communication) $\tilde{A} \equiv \nu\tilde{u}.(\mathsf{out}(w, M).P_1 \mid \tilde{A}')$ and $\tilde{D} \equiv \nu\tilde{v}.(\mathsf{in}(a, x).P_2 \mid \tilde{D}')$. In such a case, we have $w\sigma_A \notin \tilde{u}$ and $A'\sigma_A \equiv \nu\tilde{u}.(P_1 \mid \tilde{A}')\sigma_A \mid \nu\tilde{v}.(P_2\{^{M\sigma_A}/_x\} \mid \tilde{D}')$.
   (d) (communication) $\tilde{A} \equiv \nu\tilde{u}.(\mathsf{in}(w, x).P_1 \mid \tilde{A}')$ and $\tilde{D} \equiv \nu\tilde{v}.(\mathsf{out}(a, M).P_2 \mid \tilde{D}')$. In such a case, we have $A'\sigma_A \equiv \nu\tilde{u}.(P_1\{^M/_x\} \mid \tilde{A}')\sigma_A \mid \nu\tilde{v}.(P_2 \mid \tilde{D}')$.

   The two first cases can be handled as before. We deal with the case (c) (case d is rather similar).
   We have $\tilde{A}\sigma_A \xrightarrow{out(a,M\sigma_A)} \nu\tilde{u}.(P_1 \mid \tilde{A}')\sigma_A$. As $\tilde{A} \ \mathcal{R} \ \tilde{B}$, there exists $\tilde{B}'$ and $\tilde{\sigma}_B$ such that $\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{out(a,M\sigma_A)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B$ and $\nu\tilde{u}(P_1 \mid \tilde{A}')\tilde{\theta}_A \ \mathcal{R} \ \tilde{B}'\tilde{\theta}_B$ where $\tilde{\theta}_A = \sigma_A|_{X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{out(a,M\sigma_A)} \nu\tilde{u}.(P_1|\tilde{A}')\sigma_A)}$ and $\tilde{\theta}_B = \tilde{\sigma}_B|_{X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{out(a,M\sigma_A)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)}$.
   We have $\tilde{B}\tilde{\sigma}_B \mid \tilde{D} \rightarrow^* \tilde{B}'\tilde{\sigma}_B \mid \nu\tilde{v}.(P_2\{^{M\sigma_A}/_x\} \mid \tilde{D}')$. Let $\sigma_B = \tilde{\sigma}_B$ and $B' \equiv \tilde{B}' \mid \nu\tilde{v}.(P_2\{^{M\sigma_A}/_x\}|\tilde{D}')$. As $X_{\tilde{A}|\tilde{D}}(\tilde{A}\sigma_A \mid \tilde{D} \rightarrow A'\sigma_A) = X_{\tilde{A}}(\tilde{A}\sigma_A \xrightarrow{out(a,M\sigma_A)} \nu\tilde{u}.(P_1 \mid \tilde{A}')\sigma_A)$, we have $\tilde{\theta}_A = \theta_A$. As $X_{\tilde{B}|\tilde{D}}(\tilde{B}\sigma_B \mid \tilde{D} \xrightarrow{*} B'\sigma_B) = X_{\tilde{B}}(\tilde{B}\tilde{\sigma}_B \rightarrow^* \xrightarrow{out(a,M\sigma_A)} \rightarrow^* \tilde{B}'\tilde{\sigma}_B)$, we have $\tilde{\theta}_B = \theta_B$.
   Since $\nu\tilde{u}(P_1 \mid \tilde{A}')\tilde{\theta}_A \ \mathcal{R} \ \tilde{B}'\tilde{\theta}_B$, we have $\nu\tilde{u}(P_1 \mid \tilde{A}')\theta_A \mid \nu\tilde{v}.(P_2\{^{M\sigma_A}/_x\}|\tilde{D}') \ \mathcal{R}' \ \tilde{B}'\theta_B \mid \nu\tilde{v}.(P_2\{^{M\sigma_A}/_x\}|\tilde{D}')$, *i.e.* $A'\theta_A \ \mathcal{R}' \ B'\theta_B$.

   We need also to show that for all B' such that $\tilde{B}\sigma_B \mid \tilde{D} \rightarrow B'\sigma_B$ there exists A' such that $\tilde{A}\sigma_A \mid \tilde{D} \rightarrow^* A'\sigma_A$ and $A'\theta'_A \ \mathcal{R}' \ B'\theta'_B$ where $\theta'_A = \sigma_A|_{X_{\tilde{A}|\tilde{D}}(\tilde{A}\sigma_A|\tilde{D} \rightarrow^* A'\sigma_A)}$ and $\theta'_B = \sigma_B|_{X_{\tilde{B}|\tilde{D}}(\tilde{B}\sigma_B|\tilde{D} \rightarrow B'\sigma_B)}$. To do this, we need also to distinguish three cases depending on the fact that the internal reduction allows the $\tilde{B}\sigma_B$, $\tilde{D}$ or both to evolve. We conclude as above. □