Software System Engineering
Spring 2006

Digital Signature and Blocking in Mobile Ambients

Supervisor: Hans Hüttel

Group: B1-215
Anupam Palit
Bin Ren
Sagar Bingi
Yepeng Sun

Aalborg University
29th June 2006
Group: B1-215

Anupam Palit

Bin Ren

Sagar Bingi

Yepeng Sun

Preface

The calculus of Mobile Ambients derives its process primitives from the π-calculus. It introduces the notion of a bounded environment(the ambient) where processes or mobile agents co-operate. An ambient consists of a set of local agents and possibly other subambients. Ambients are moved as a whole under the control of the enclosed agents, which are confined to their ambients. This report presents a variant of Mobile Ambients. It addresses the security issues of robustness against malicious tampering, access control and execution safety by introducing the concept of *digital signature* and *blocking* into ambients. The model of Digitally Signed Ambients with Blocking on Capabilities(DSABC) is similar to the Java Sandbox model for JDK2.0 and Umbrella project developed for Linux security. A type system which captures various security policies has been presented based on the proposed calculus. This project report is developed as our master's thesis at Department of Computer Science, Aalborg University, Denmark.

We would like to thank Mr.Hans Hüttel for encouraging and challenging us throughout the development of this project and constantly providing us with new ideas and invaluable suggestions, never accepting less than our best efforts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This report describes two security mechanisms introduced into Mobile Ambients[1]: digital signature and access control for migrating processes. This calculus is named as *Digitally Signed Ambients with Blocking on Capabilities* (DSABC). This idea emerged while investigating the concept of mobile ambients and the security mechanism in Internet.

1.1 Mobility and Security

Mobile Computing is a generic term used to represent the ability to handle information access, communication and business transaction, that wirelessly connect and use centrally located information or application software while in the state of motion. The devices that are involved in mobile computation could be portable wireless computing and communication devices such as laptops, mobile phones, PDA's etc.

Mobile computation is the notion that running programs need not be forever tied to a single network node. A program can cross barriers, move between different networks and be able to execute at different virtual locations. One of the major concerns in the concept of mobile computation is security. Mobile computation has to do with virtual mobility that is mobile software, while mobile computing has to do with physical mobility that is mobile hardware.

The major security concerns of mobile computation are:

1. **Confidentiality:** Confidential information such as passwords, credit card information etc, should not be intercepted by third parties when transmitted over the Internet.

2. **Integrity:** It ensures that confidential information should not be gained access, modified or destroyed during the transit.

3. **Accountability:** It ensures that all the entities in the network are held responsible for their activities. For example we would not want migrating processes over the network to wreak havoc at the receiving site. In such a case we should be able to track the responsible entity.

4. **Availability:** It ensures that the authorized entities always have access to the desired services. Attacks like denial of service must be prevented.

5. **Legitimate use:** It must ensure that only intended users are allowed to gain access to the desired resources. For example, network servers should be safeguarded against intrusion from unwanted parties.

6. **Access Control:** It refers to practice of restricting entrance to a property. It is implemented by using access rights. In distributed systems some parts of the system have restricted access rights that is not every process should be allowed to have direct access to all the resources. For example, we would not want all the processes to be able to gain access to the operating system kernel.

1.2 Mobile Ambients(Review)

Ambient calculus [1] is a process calculus devised by Luca Cardelli and Andrew D.Gordon in 1998. The main aim of Ambient calculus is to model new computational phenomena over wide areas network. Ambients are used to represent bounded places for computation such as a web page (bounded by a file), a virtual address space (bounded by an addressing range), a laptop (bounded by its case and data ports) etc. Ambients move into and out of other ambients bringing along moving processes, static processes and possibly other ambients. The three basic ambient primitives are in , out and open . The following are the main characteristics of mobile ambients:

- Each ambient has a name.
- An ambient boundary defines the scope of a computation, therefore establish a container, which may be easily identified.
- An ambient is a collection of agents, i.e. processes, which run directly inside the ambient. $n[P]$ is an ambient named n with the running process P inside.
- Ambients can move, i.e. they can enter or exit other ambients.
- An ambient moves with all the subambients and processes inside it. This may occur by performing an in or out operation.

n	names
$P, Q ::=$	processes
$(\nu n)P$	restriction
0	inactivity
$P\|Q$	composition
$!\,P$	replication
$n[P]$	ambient
$M.P$	action
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n

Table 1.1: The syntax of Mobile Ambients

- Movement operations initiated by an ambient on itself are called *subjective moves*, while *objective moves* are performed by an ambient on the other ambient.
- Ambients can be nested inside other ambients. Each subambient has its own name and behaves as an independent ambient. One example of nesting is: A Java applet is running within a Java Virtual Machine, the Java Virtual Machine is running within a Web Browser, the Web Browser is running within an operating system, the operating system is running on a workstation.

1.2.1 The Syntax

In the syntax we look at the mobility primitives. The main categories in the syntax are the processes and their capabilities. The syntax is shown in Table 1.2.1:

Restriction, written as $(\nu n)P$, makes the name n private and unique to P. No other process can use this name for interacting with P. Restriction is a binder and P is its scope. *Inactivity*, is the process that does not reduce. *Composition*, written as $P|Q$ means that P and Q are running in parallel. *Replication*, written as $!\,P$, simulates recursion by spinning off copies of P. *Ambient*, written as $n[P]$, is composed of two parts (it is a binary operator) where n is the name of the ambient and P is the active process inside it. The square brackets around P indicate the perimeter of the ambient. If the ambient moves, everything inside moves with it. *Action* prefix written as $M.P$, represents a process where P is enabled only if the prefix M has been consumed. Capabilities can be thought as terms that enable the ambients to perform some actions. An ambient gains the ability to go inside ambient n with $\mathsf{in}\ n$ capability. An ambient gains the ability

$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(\mathsf{in}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(\mathsf{out}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(P\|Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{open}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\,P)$	$\triangleq$	$\mathrm{fn}(P)$			
$\mathrm{fn}(n[P])$	$\triangleq$	$\{n\} \cup \mathrm{fn}(P)$			
$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$			

Table 1.2: The free names of Mobile Ambients

to leave its parent ambient n with $\mathsf{out}\ n$ capability. Ambient n can be dissolved by the means of $\mathsf{open}\ n$ capability.

The only ambient name binding operator is restriction. The names that are not bound by a restriction operator are thus free names. We denote by $\mathrm{fn}(P)$ the set of free names of process P, Table 1.2 gives the set of free names $\mathrm{fn}(P)$ for Mobile Ambients.

We define the bound names $\mathrm{bn}(P)$ as those names created by a restriction within a process P. For example, in $(\nu n)n[Q]$, n is a bound name. It is possible that a name is a bound name and a free name at the same time. For example, in $n[0]|(\nu n)n[0]$, name n is a free name and also a bound name.

Safe Systems

The main characteristics of safe systems is that the system can avoid harmful foreign codes over the Internet to enter or be executed. A safe system usually provides access control on the incoming processes based on their source. We model the safe system using various variants of the calculi of mobile ambients during the evolution of this project.

Example 1.2.1 *Modeling of Safe System using Mobile Ambients:*

$$app_1[\mathit{in}\ sys.P]|app_2[\mathit{in}\ sys.Q]|sys[T] \rightarrow sys[app_1[P]|app_2[Q]|T]$$

In Example 1.2.1, sys represents a safe system containing the internal resource T. Two applets app_1 and app_2 can get into the system by exercising $\mathsf{in}\ sys$ capability. We can observe that the system can be modeled using mobile ambients to show the movement of the applets but the security mechanism cannot be modeled effectively using mobile ambients due to the following reasons:

- If $app1$ is trusted whereas $app2$ is not an trusted ambient. The system cannot distinguish the two kinds of ambients.

- The system does not provide access control on the incoming codes.

The diagrammatic representation of the model of safe system is shown in Figure 1.1.

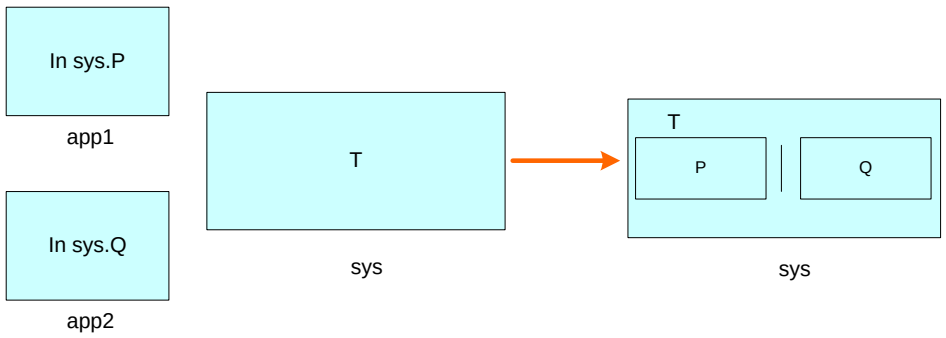

Figure 1.1: Modeling Safe System using Mobile Ambients

1.3 The Characteristics of Digital Signature

A digital signature [2] is a string of bits that is computed from the data transmitted and the private key of an entity. The signature can be used to verify if the data comes from the correct entity and is not tampered during transit.

The main characteristics of digital signature are:

1. **Authenticity:** It ensures the correct sender sends the message. This can be verified through authentication by using the public key corresponding with the private key used to generate the signature. For example executables transmitted over the Internet. Authenticity is required when an update claims to be a security update from Microsoft, the users should be able to verify that the executable does come from Microsoft before they install it .

2. **Integrity:** It ensures that the message is not altered accidentally or maliciously during the transit. For example it should not be possible for some third party to be able to insert malicious behavior in a security update from Microsoft.

3. **Non-repudiation:** The word non-repudiation is a synonym of non-denial which means that it can be verified that the sender and the receiver were

the parties who claimed to send or receive the message in case the ownership is denied by one of the parties. For example it could happen that when an user downloads some security update from Microsoft website which has some malicious behaviors in that case Microsoft should not be able to defend themselves by proving that those updates were not downloaded from their website.

1.4 Java Sandbox Model

Sandbox Model JDK1.0

The first sandbox model[3] was introduced in JDK1.0 to provide a restricted environment to run untrusted code. This restricted environment was named as sandbox and the code running within it was provided only limited access to resources. The local code was provided complete access to all the system resources. This model is shown in the Figure 1.2.

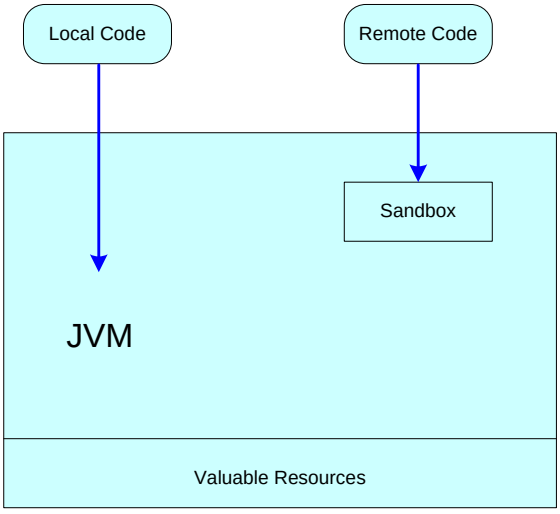

Figure 1.2: JDK 1.0 Security Model

Sandbox Model JDK1.1

The notion of Signed Applet was introduced in JDK1.1. A digitally signed applet whose signature could be verified by the receiver is treated as the local code. Unsigned applets or applets whose signature cannot be verified are allowed to run only within the sandbox. The applets along with their signatures, are delivered in the JAR (Java Archive) format. The concept of signed applet is

shown in the Figure 1.3

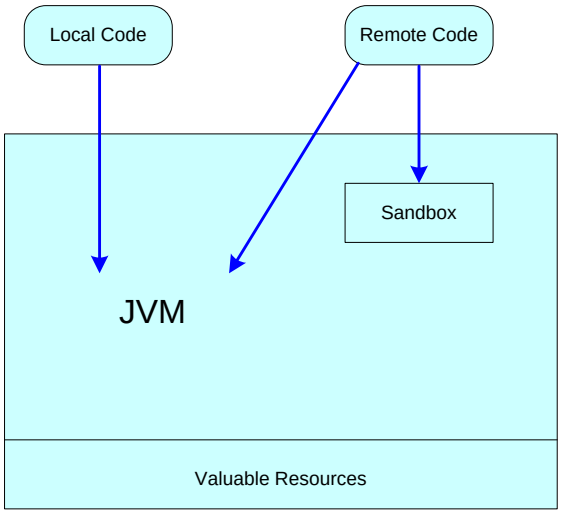

Figure 1.3: JDK 1.1 Security Model

Java uses the concept of digital signature[4] to recognize the source of the code, and restricts the running of codes by using *Protection Domain* in sandbox model.

- **Protection Domain:** It consists of a set of classes whose instances are granted the same set of permissions. Those classes coming from the same *CodeSource* which is a set of public keys together with a *codebase*(in the form of *url*), are usually placed into the same domain according to the public key which identifies where the classes come from. All code shipped as part of the JDK is treated as trusted code, and is placed into the system domain. All other applets and applications are placed into appropriate domain, determined by their *CodeSource*.

- **Permission:** An applet needs permission for accessing a particular resource. The security policy in a Java application environment specifies which permissions are available for code from a specified *CodeSource*. Each permission typically has a name. In JDK there are a number of built-in permission types, and additional ones can be configured by the users.

- **Policy:** It specifies which permissions are available for code from various code sources. The source location for the policy information is decided

by the policy implementation. The JDK contains a default Policy implementation that obtains its information from static policy configuration files.

- **Security Manager:** The main functionality of the security manager is to check the policy currently in effect and perform access control checks. The security manager prevents applet code from accessing resources unless it is explicitly granted permission to do so by an entry in a policy file.

The analysis of the calculus of mobile ambients, advantages and applications of digital signatures including the Java Sandbox model, provides us with the motivation to introduce the concept of digital signature into the calculus of mobile ambients in the Chapter 2. We introduce a new notion like Java applet into our calculus by introducing the concept of code along with two new capabilities auth and enter.

1. A new notion of code is introduced. A code is a signed piece of information which needs a carrier that is another ambient for its movement.
2. All the information(processes) contained within the code are passive i.e, it cannot undergo any changes within the boundaries of the code.
3. The code is considered to be signed by default, and its signature is generated by encrypting the hash value of its contents with the key k.
4. A code is represented as $m\{P\}_{sig}$, where P is the passive process enclosed within the code.
5. The auth capability is introduced. The main functionality of this capability is to authenticate a piece of code. This is used to ensure that the code comes from the intended sender. In the following example:

$$m\{P\}_{h(k,P)}|a[\mathsf{auth}(m,k).Q] \rightarrow a[P|Q]$$

   The code m is allowed to enter the ambient a if it can be authenticated i.e, if the key k contained within the ambients auth capability matches the key using which the code was signed.

6. The enter capability is introduced. The main functionality of this capability is to sign a piece of code. The ambient holding the enter capability is the holder of the private key k and hence can sign itself through entering a code with the matched key in the enter capability. So that the digital signature of the code is recalculated. In the following example:

$$m\{P\}_{h(k,P)}|a[\mathsf{enter}(m,k).Q] \rightarrow m\{P|a[Q]\}_{h(k,P|a[Q])}$$

The ambient a can enter a code m if it holds the private key k with which the code m was originally signed, once ambient enters into the code, the signature of the code has to be recalculated and hence its digital signature changes.

1.5 Blocking

For implementing the concept of access control in mobile ambients, we investigate the concept of blocking in π-calculus [5] by Vivas Frontana in his Phd thesis. The concept of blocking is similar to access control on resources in the real world. Unix restricts the access to its files using read, write and execute permissions. The Umbrella model [6] proposed for Linux enforces authentication of files along with process based mandatory access control at process level, by a set of restrictions for each process, where every process has at least the restrictions of its parent. The further details of the Umbrella model are mentioned in Chapter 5.

After investigating the concept of blocking in Mobile Ambients, three kinds of possible blocking were introduced:

- Blocking on Names
- Blocking on Capabilities
- Blocking at Ambient level

Out of the three kinds of blocking, the Blocking on Capability is flexible and is similar to the access control mechanism of Umbrella. The capabilities which are blocked cannot be exercised within the blocking scope. The concept of blocking is explained in detail in Chapter 4.

Ambient calculus is a process calculus used to effectively model the mobile computations over the network. Mobile ambients in the real world are like agents which are autonomous and intelligent programs that move through the network interacting with other services over the network. Security in mobile computation is one of the major concerns which the calculus of Mobile Ambients cannot address effectively. That is the main reason we introduce the concept of digital signature and access control on migrating processes to ensure more security and security. We develop a new model very similar to sandbox model for Java2 and the Umbrella project developed for Linux with a few variations. The concept of digital signature is introduced in the calculus of DSA by signing ambients directly using a private key. Access control on process is captured by introducing the concept of blocking into mobile ambients. The calculus of Digitally Signed

Ambients(DSA), captures the concept of *digital signature,* while the calculus of Mobile Ambients with Blocking on Capabilities(MABC) captures the property of *access control* on processes. For capturing both these properties we combine the two calculi into one hybrid calculus and name it as Digitally Signed Ambients with Blocking on Capabilities(DSABC). Advance safety properties like *key distribution*, *who can visit whom* and *minimal blocking* for access control on migrating processes are captured in its type system.

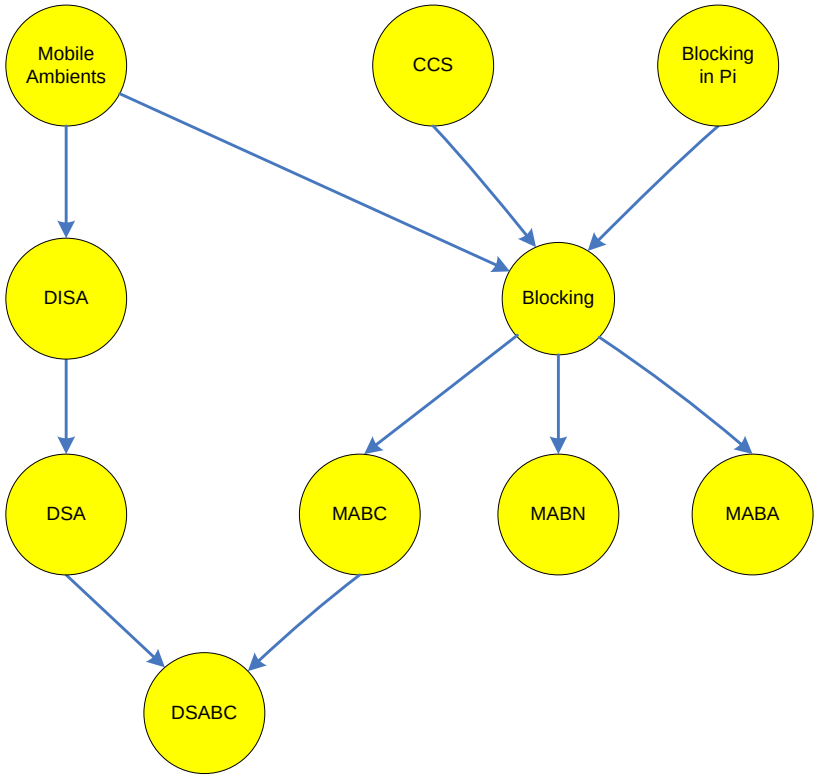

Figure 1.4: Organization of the report

The report is organized as follows: Chapter 2 describes the review of our previous work on Digital Signature for Mobile Ambients(DISA) along with its syntax and semantics. In DISA we consider a piece of code to be digitally signed by default. Chapter 3 is a discussion of the extension to the concept of DISA into Digitally Signed Ambients(DSA). In the new concept of DSA we directly sign an ambient instead of using code and describe its syntax and semantics. Chapter 4 introduces a new concept of blocking into mobile ambients. In this chapter we investigate the three different kinds of blocking in mobile ambients namely Blocking on Names(MABN), Blocking on Capabilities(MABC) and Blocking at Ambient Level(MABA) and describe their syntax and semantics. Chapter 5 describes a hybrid calculus formed by merging the calculus of DSA along with the

calculus of MABC and describe its syntax and semantics along with its advantages. In Chapter 6 we give a detailed description of type system for the calculus of DSABC. Finally, we conclude in Chapter 7. The organization of this report is shown in Figure 1.4. The arrows in the figure indicate the evolution of the report.

Chapter 2

Digital Signature for Mobile Ambients

The calculus of Mobile Ambients is effective in representing mobile computations over complex network structures but has a few security breaches over the open network. It is not secure enough to prevent intrusions and message tampering. The concept of digital signature has been effectively implemented in the real world to resolve this issue. The calculus of DISA [7] was developed in the previous semester with the main aim to introduce the concept of Digital Signature into mobile ambients. This serves as the foundation for our present work on digital signature and blocking in mobile ambients.

2.1 Syntax and Semantics

In this section, we introduce syntax and semantics for DISA .

2.1.1 Syntax

We now introduce the primitives for DISA . We assume there is an infinite set of names, $\mathscr{N}$, ranged over by n and m, where n and m are the ambient and code names respectively, k denotes the key in the digital signature, and C denotes a code group name. Table 2.1 shows the syntax of DISA . The main syntactic categories are processes, digital signature and capabilities.

All the processes have the same informal meaning as described in mobile ambients. Two new processes primitives, restriction on code name, $(\nu m : C)P$, and code $m\{P\}_{sig}$ are introduced.

The process $(\nu m : C)P$, creates a new code name m within a scope P, makes

the code name private and unique to P, C is a code group name, m belong to code group C. The new code name can be used to name codes and to be operated within ambients by name. No other process can use this code name for interacting with P.

A code is a bounded area surrounded by delimiters consisting of special kind of passive process which is represented as $m\{P\}_{sig}$, where P is the passive process, m is the code name, sig is the digital signature which is obtained by encrypting the hash value of P with a key k. Each code has a name. A code may contain other ambients or processes within it. A code can enter an ambient through authentication and a code can allow an ambient to enter it through signing.

We define the form of digital signature, $h(k, P)$ where h is a digital signature creation algorithm. It first calculates the hash value of process P, and encrypts hash value with the key k. For code $m\{P\}_{h(k,P)}$, in $h(k, P)$, P is always the same as the P inside the code m.

Two new capabilities are introduced, $\mathsf{enter}(m, k)$ can sign m with key k, and $\mathsf{auth}(m, k)$ can authenticate m with key k. We will describe the two capabilities in the reduction rules.

Table 2.2 gives the set of free names $\mathrm{fn}(P)$ for DISA .

Table 2.3 shows the syntactic conventions of DISA .

2.1.2 Semantics

Structural congruence is a way of identifying processes that we do not want to differentiate for any semantic reason. Processes of calculus are grouped into equivalence classes by the relation $\equiv$, which denotes structural congruence to identify the processes which intuitively represent the same thing. In particular, structural congruence is an important relation because it is used to rearrange the process so that they can reduce.

Table 2.4 defines the structural congruence of DISA which inherits seventeen rules from the calculus of mobile ambients, we only comment on the new clauses in the definition.

(Struct ResC): The restriction on code name has no effect on structurally congruent processes.

k	key
m, n	names
C	code group name
$P, Q ::=$	processes
$(\nu m : C)P$	restriction on code name
$(\nu n)P$	restriction on ambient name
0	inactivity
$P \mid Q$	composition
$!\, P$	replication
$n[P]$	ambient
$M.P$	action
$m\{P\}_{sig}$	code
$sig ::= h(k, P)$	digital signature
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n
$\mathsf{enter}(m, k)$	can sign m with key k
$\mathsf{auth}(m, k)$	can authenticate m with key k

Table 2.1: The syntax of DISA

$\mathrm{fn}((\nu m : C)P)$	$\triangleq$	$\mathrm{fn}(P) - \{m\}$	$\mathrm{fn}(h(k, P))$	$\triangleq$	$\{k\} \cup \mathrm{fn}(P)$
$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(\mathsf{in}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(\mathsf{out}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(P \mid Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{open}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\, P)$	$\triangleq$	$\mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{enter}(m, k))$	$\triangleq$	$\{m, k\}$
$\mathrm{fn}(n[P])$	$\triangleq$	$\{n\} \cup \mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{auth}(m, k))$	$\triangleq$	$\{m, k\}$
$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$			
$\mathrm{fn}(m\{P\}_{sig})$	$\triangleq$	$\{m\} \cup \mathrm{fn}(P) \cup \mathrm{fn}(sig)$			

Table 2.2: The free names of DISA

$(\nu m : C)P\|Q$	is read	$((\nu m : C)P)\|Q$
$(\nu n)P\|Q$	is read	$((\nu n)P)\|Q$
$M.P\|Q$	is read	$(M.P)\|Q$
$(\nu n_1 \ldots n_j)P$	$\triangleq$	$(\nu n_1)\ldots(\nu n_j)P$
$n[]$	$\triangleq$	$n[0]$
$m\{\}_{sig}$	$\triangleq$	$m\{0\}_{sig}$
M	$\triangleq$	$M.0$

Table 2.3: The syntactic conventions of DISA

(Struct Res ResC): More than one restriction of different names is applied on a process consecutively, the order of restrictions does not matter.

(Struct Code): If $P \equiv Q$ then $m\{P\}_{h(k,P)}$ and $m\{Q\}_{h(k,Q)}$ shows the same behavioral properties, even if they have different hash values.

(Struct ResC Par): $m \notin \text{fn}(P)$, means m is not bound within process P, therefore $(\nu m : C)(P|Q) \equiv P|(\nu m : C)Q$ is obtained.

(Struct Zero ResC): The restriction on code name has no effect on inactivity.

We now introduce reduction relation $\rightarrow$ to identify the behavior of processes.

The calculus of DISA expresses mobility through the migration of processes, from one locality to another. The meaning of computation is represented by the interaction among codes and ambients. Ambients, with the appropriate capability, can enter other ambients, an ambient from inside can exit or a boundary can be dissolved, an ambient can allow a code to enter it or an ambient can enter a code. These are the basic actions that define computation in the calculus of DISA . The reduction relations as mathematical tool capture those basic movements.

The Reduction rules of DISA inherits seven rules from the calculus of mobile ambients, table 2.5 gives all the reduction rules.

For the two rules (Red Enter) and (Red Auth), the code and ambient must be at the same level.

(Red Enter) It expresses the ambient n can enter a code m if n has the same key k with which the code m was signed, after the ambient enters the code, the

$P \equiv P$	(Struct Refl)
$\dfrac{P \equiv Q}{Q \equiv P}$	(Struct Symm)
$\dfrac{P \equiv Q, Q \equiv R}{P \equiv R}$	(Struct Trans)
$\dfrac{P \equiv Q}{(\nu m : C)P \equiv (\nu m : C)Q}$	(Struct ResC)
$\dfrac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$	(Struct Res)
$\dfrac{P \equiv Q}{P \mid R \equiv Q \mid R}$	(Struct Par)
$\dfrac{P \equiv Q}{!\,P \equiv !\,Q}$	(Struct Repl)
$\dfrac{P \equiv Q}{n[P] \equiv n[Q]}$	(Struct Amb)
$\dfrac{P \equiv Q}{M.P \equiv M.Q}$	(Struct Action)
$\dfrac{P \equiv Q}{m\{P\}_{h(k,P)} \equiv m\{Q\}_{h(k,Q)}}$	(Struct Code)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$!\,P \equiv P \mid !\,P$	(Struct Repl Par)
$(\nu m : C)(\nu m' : C')P \equiv (\nu m' : C')(\nu m : C)P$	(Struct Res ResC)
$(\nu n)(\nu n')P \equiv (\nu n')(\nu n)P$	(Struct Res Res)
$(\nu m : C)(P \mid Q) \equiv P \mid (\nu m : C)Q$ if $m \notin \mathrm{fn}(P)$	(Struct ResC Par)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\nu n)(n'[P]) \equiv n'[(\nu n)P]$ if $n \neq n'$	(Struct Res Amb)
$P \mid 0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$(\nu m : C)0 \equiv 0$	(Struct Zero ResC)
$!\,0 \equiv 0$	(Struct Zero Repl)

Table 2.4: The structural congruence of DISA

$a[\mathsf{in}\ b.P\|Q] \mid b[R] \ \rightarrow\ b[R\|a[P\|Q]]$	(Red In)
$b[a[\mathsf{out}\ b.P\|Q]\|R] \ \rightarrow\ b[R]\ \|a[P\|Q]$	(Red Out)
$\mathsf{open}\ a.P \mid a[Q] \ \rightarrow\ P \mid Q$	(Red Open)
$m\{P\}_{h(k,P)} \mid a[\mathsf{enter}(m,k).Q\|R] \ \rightarrow\ m\{P\|a[Q\|R]\}_{h(k,P\|a[Q\|R])}$	(Red Enter)
$m\{P\}_{h(k,P)} \mid a[\mathsf{auth}(m,k).Q\|R] \ \rightarrow\ a[P\|Q\|R]$	(Red Auth)
$\dfrac{P \rightarrow Q}{(\nu n)P \rightarrow (\nu n)Q}$	(Red Res)
$\dfrac{P \rightarrow Q}{(\nu m : C)P \rightarrow (\nu m : C)Q}$	(Red ResC)
$\dfrac{P \rightarrow Q}{n[P] \rightarrow n[Q]}$	(Red Amb)
$\dfrac{P \rightarrow Q}{P\|R \rightarrow Q\|R}$	(Red Par)
$\dfrac{P' \equiv P, P \rightarrow Q, Q \equiv Q'}{P' \rightarrow Q'}$	(Red $\equiv$)

Table 2.5: The reduction rules of DISA

signature of the code has to be recalculated and hence its signature changes.

(Red Auth) It expresses the ambient n allows the code m to enter it and activate processes P if the key k contained in its auth capability matches the key k with which the code m was signed.

(Red ResC) It means the restriction has no effect on the internal event within the restriction scope.

2.2 Discussion

In the DISA , we introduce the concept of code. We also introduced the interaction among codes and ambients, an ambient can allow a code to enter it or vice versa. In practice, one interesting feature of the calculus of DISA is the possibility to capture problems related to network security.

An ambient can be considered to be an administrative domain, such as a PDA, and the process inside it accepts only trusted codes. Consider the follow-

ing process:

$$P = m\{Q\}_{h(k,Q)} \mid m\{R\}_{h(k',R)} \mid b[\mathsf{auth}(m,k)|S]$$

There are two codes having the same name m signed using two different keys, k and k'. If the two codes want to enter ambient b, according to the (RED AUTH) rule, if the k in m is the same as the k in auth capability, then the code m with the k can enter ambient b, another code m with k' is rejected. If m with k added to PDA that the process Q can interact with the process S. All data held on PDA is fully protected, because only the trusted code is allowed to enter the PDA. DISA has been developed to capture the notion of digital signatures. In the real world we assume signed applets to be similar to code in our calculus. An attempt was made to model the concept of digital signature into mobile ambients. There are still scope of improvements as follows:

1. The code is static that is it cannot move by itself. Movement is possible only with the use of a carrier.

2. All the processes within the code are passive that means reductions cannot happen within the code which may not be true in the real world.

3. The model of DISA is based on the sandbox model of Java1 and not the latest sandbox model of Java2. In DISA , the untrusted code is not allowed to enter the system where as trusted code is given complete access to all the system resources.

4. There is no key distribution i.e, it cannot be made sure if the code was indeed signed by the desired signatory.

5. The concept of code and ambients is not unified. That is we consider code and ambients to be two separate entities.

Chapter 3

Digitally Signed Ambients

In this chapter, we introduce the calculus of Digitally Signed Ambients. In Section 3.1, we investigate the applications of DISA , its weakness, and introduce the new concept of digitally signed ambients based on mobile agents. In Section 3.2, we give the formal definition of the calculus of digitally signed ambients, and discuss its security concerns.

3.1 Signing Ambients

In DISA , an ambient can be signed by a code by exercising the enter capability, as a result the ambient enters the code and becomes a static process. If this code is authenticated by another ambient, then the static processes contained within the code become active, enters the authenticating ambient and becomes a part of it. A code is a passive process which cannot move by itself but can only be carried by other ambients. A code can be used to store ambients in passive state, and guarantee integrity of the stored ambients by signing them using digital signature.

DISA can be used to model the traditional view of remote code: The source site signs the code with its private key, and sends the signed code to the destination site; the destination site authenticates the code with the public key of the source site, then activates the code as processes in case of successful authentication; during transit, the code cannot be altered, hence it cannot execute by itself. For example, a signed applet downloaded from a website is a static Java byte code, which can be executed after its authentication at the destination. Applets are always treated as passive objects until they are executed.

However, it is believed that the current trends in Internet technology lead to the use of *mobile agents*[8], which are different from Java applets. Mobile agents are programs that can migrate from host to host in a network, decided by their

own choosing according to the times, places and other environmental factors. The state of the running program is saved, transported to the new host, and restored, allowing the program to continue where it left. Because mobile agents can make decision and move by themselves, they can run as they move. Mobile agents can be treated as some autonomous subjects.

The calculus of mobile ambients is a mathematic model for mobile agents. In mobile ambients, a transporting ambient containing some critical information or processes can be a subambient of a source host ambient; then the transporting ambient moves itself into a destination host ambient through a series of reductions, and maybe travel across other untrustful ambients during the transit. This arises some security concerns as following:

- How does the destination host ambient identify that the transporting ambient comes from the source host ambient it expects?
- How dose the destination host ambient guarantee that the transporting ambient is not inserted some harmful processes by those untrustful ambients during the transit?

As a solution for these security concerns, instead of signing ambients through static code, we can sign ambients directly.

$n[P]_k$ denotes a digitally signed ambient. n is the ambient name, P is the process, k is the key which is used to sign process P. Traditionally if a process is digitally signed, it is supposed not to be changed during transit. However, a digitally signed ambient should keep the capabilities of autonomous mobility, so that the process enclosed by the signed ambient always changes, how do we explain this? We can understand it like this: if an action is initiated by an ambient, the ambient should be able to recalculate its own digital signature since the process enclosed by the ambient is changed; on the contrary, a digitally signed ambient does not allow to be changed by other ambients. For example:

$$P = a[\mathsf{in}\ b.S]_k | b[T]$$
$$Q = a[\mathsf{in}\ b.S]_k | b[T]_{k'}$$

P can be reduced to $b[T|a[S]_k]$, ambient a can enter ambient b. After $\mathsf{in}\ b$ is consumed, the process enclosed by ambient a is naturally changed, the signature is recalculated by ambient a and it becomes $a[S]_k$. On the contrary, in process Q, $\mathsf{in}\ b$ cannot be executed, because ambient b is digitally signed, it will prevent the change initiated by ambient a.

The only way to change a digitally signed ambient is that the action is initiated by this digitally signed ambient. auth capability can be initiated by a digitally signed ambient. For example:

$$a[S]_k|b[\mathsf{auth}(a,k).T]_{k'} \rightarrow b[T|a[S]]_{k'}$$

Ambient b authenticate digitally signed ambient a with key k, then ambient a become unsigned and trusted by ambient b, so that ambient a become a subambient of ambient b. $\mathsf{auth}(a,k)$ is initiated by signed ambient b, so that it can accept a trustful ambient through authentication and changes its own digital signature.

There is another case we have to consider: if an event only happen within a digitally signed ambient, how dose this ambient deal with it? For example, in $n[a[\mathsf{in}\ b]|b[0]]_k$, ambient a can enter ambient b, this is an internal event for ambient n. Because ambient a and ambient b are subambients of ambient n, we can take them as parts of ambient n, so that the internal event can be regarded as an internal action of ambient n. Since we understand the internal action within an ambient is also initiated by the ambient, the ambient can still recalculate the digital signature after its enclosed process changes.

In general, an action is related with a subject and an object(In case of the internal event, it is only related a subject), the action is initiated by the subject, the subject can change its own digital signature. And if the object is digitally signed, its digital signature cannot be changed by the subject, so that it can prevent any change caused by the environment and guarantee the integrity of the process. We will give the formal definition of the calculus of *Digitally Signed Ambient*(DSA) in Section 3.2.

3.2 Syntax and Semantics

Table 3.1 gives the formal definition of the syntax of Digitally Signed Ambients. All the processes have the same meaning as described in section 5.2 except, for one new process, i.e, signed ambient. Ambient is signed directly and it is written as:

$$n[P]_s$$

Where n is the name of the ambient, P is the process running inside the ambient and s can be either key k or an empty key. When $s = k$, n is signed ambient. As we have seen in DISA , signed ambient can be written as $n[P]_{h(k,P)}$, where $h(k,P)$ is the digital signature obtained by encrypting the hash value of the process P using key k. But for simplicity, we write signed ambient as $n[P]_k$. When $s = \epsilon$, n is normal ambient, which can be opened by any process who holds open n capability. The digital signature of a signed ambient can be changed by itself or by its enclosed subambients or by any process who holds the sign capability with the corresponding key.

n	names
$s ::= k \mid \epsilon$	key
$P, Q ::=$	processes
$(\nu n)P$	restriction
0	inactivity
$P \mid Q$	composition
$!\,P$	replication
$n[P]_s$	signed ambient
$M.P$	action
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n
$\mathsf{auth}(n, k)$	can authenticate n with key k
$\mathsf{sign}(n, k)$	can sign n with key k

Table 3.1: The syntax of DSA

Table 3.2 shows the definition of free names of DSA. The only difference from that of DISA is $\mathrm{fn}(n[P]_s)$. In DISA , the ambient is a normal ambient, but here it is a signed ambient. Therefore, we should also consider the free names of s.

Table 3.3 shows the definition of syntactic conventions of DSA. Most of them are similar to the syntactic conventions of mobile ambients. The only difference is with signature. If $s = k$ then that ambient is read as a singed ambient. If $s = \epsilon$, the ambient is read as a normal ambient.

$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(\mathsf{in}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(\mathsf{out}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(P \mid Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{open}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\,P)$	$\triangleq$	$\mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{sign}(m, k))$	$\triangleq$	$\{m, k\}$
$\mathrm{fn}(n[P]_s)$	$\triangleq$	$\{n\} \cup \mathrm{fn}(P) \cup \mathrm{fn}(s)$	$\mathrm{fn}(\mathsf{auth}(m, k))$	$\triangleq$	$\{m, k\}$
$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$	$\mathrm{fn}(\epsilon)$	$\triangleq$	$\emptyset$
$\mathrm{fn}(k)$	$\triangleq$	$\{k\}$			

Table 3.2: The free names of DSA

$(\nu n)P\|Q$	is read	$((\nu n)P)\|Q$
$M.P\|Q$	is read	$(M.P)\|Q$
$(\nu n_1 ... \nu n_m)P$	$\triangleq$	$(\nu n_1)...(\nu n_m)P$
$n[P]_\epsilon$	$\triangleq$	$n[P]$
$n[]_s$	$\triangleq$	$n[0]_s$
M	$\triangleq$	$M.0$

Table 3.3: The syntactic conventions of DSA

Table 3.4 shows the definition of structural congruence of DSA. The new rules are explained as follows:

(Struct Amb) : If processes P and Q which are enclosed in ambient n, are structurally equivalent, then even though the digital signatures of the ambients are different, the two ambients show the same behavioral properties.

(Struct Res Amb) : This rule is the same as in Mobile Ambients. In addition to the Struct Res Amb, the following should be emphasized:

$$(\nu n)(n^{'}[P]_k) \not\equiv n^{'}[(\nu n)P]_k$$

The two processes $(\nu n)(n^{'}[P]_k)$ and $n^{'}[(\nu n)P]_k$ are not structurally equivalent. Since the restriction present in $(\nu n)(n^{'}[P]_k)$ is not signed, it can extend its scope. Where as the restriction present in $n^{'}[(\nu n)P]_k$ is signed, therefore it can not extend its scope.

Table 3.5 shows the reduction rules of DSA.

(Red In): Reduction In can be performed only when an ambient enters an unsigned ambient. As shown in Table 3.5 an ambient n tries to enter unsigned ambient m. The reason is when the capability in m is exercised within the ambient n, the contents of n changes accordingly. However, a subject can change its own digital signature, by exercising the in capability which will changes the digital signature of n. In contrast, subject can not change the digital signature of the object. When n enters m, the contents of m will automatically be changed. Hence, if m is a signed ambient, the digital signature of m has to be changed by n, which is not reasonable and hence the In reduction is not possible.

(Red Out): Reduction Out can be performed only when an ambient moves out of an unsigned ambient. In Red Out rule the ambient n tries to move out of an ambient m. Here ambient n is subject, which can change its own contents by exercising out capability. When ambient n moves out from m, if m is a

$P \equiv P$	(Struct Refl)
$\dfrac{P \equiv Q}{Q \equiv P}$	(Struct Symm)
$\dfrac{P \equiv Q, Q \equiv R}{P \equiv R}$	(Struct Trans)
$\dfrac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$	(Struct Res)
$\dfrac{P \equiv Q}{P \mid R \equiv Q \mid R}$	(Struct Par)
$\dfrac{P \equiv Q}{!\,P \equiv !\,Q}$	(Struct Repl)
$\dfrac{P \equiv Q}{n[P]_s \equiv n[Q]_s}$	(Struct Amb)
$\dfrac{P \equiv Q}{M.P \equiv M.Q}$	(Struct Action)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$!\,P \equiv P \mid !\,P$	(Struct Repl Par)
$(\nu n)(\nu n^{'})P \equiv (\nu n^{'})(\nu n)P$	(Struct Res Res)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\nu n)(n^{'}[P]) \equiv n^{'}[(\nu n)P]$ if $n \neq n^{'}$	(Struct Res Amb)
$P \mid 0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$!\,0 \equiv 0$	(Struct Zero Repl)

Table 3.4: The definition of structural congruence

signed ambient, digital signature of m has to be changed by subject n, which is not reasonable. Therefore, if m is signed ambient, RED OUT can not be possible.

(RED OPEN): Only an unsigned ambient can be opened. In RED OPEN rule process P tries to open the ambient n through open capability. Therefore, if n is signed ambient, it can not allow other processes to dissolve its boundary.

(RED AUTH): Reduction AUTH is special type of reduction rule, the only way in which subject can accept the signed ambient. Here ambient n tries to authenticate a signed ambient m through auth capability. In general, a subject can not change the digital signature of an object. But in this rule, since ambient n holds valid public key, it can accept the signed ambient m. Therefore, after auth capability is exercised, ambient n changes its own contents, hence digital signature of ambient n is recalculated.

(RED SIGN): Reduction SIGN is performed only when a process who holds valid private key, signs an unsigned ambient.

(RED AMB): In $n[P]_k$, it is understood that P is actively running, and that P can be the parallel composition of several processes. We emphasize that P is running even when the surrounding ambient is moving. As we know the digital signature of an ambient can be changed by itself or by its enclosed ambients, we express the fact that any reduction of the process which is enclosed in an signed ambient will lead to recalculation of the hash value of that ambient.

Example 3.2.1 *Modeling of the Secure Systems using DSA.*

$$app_1[\mathsf{auth}(app_3, k_3)|P]_{k_1}|app_2[Q]_{k_2}|app_3[R]_{k_3}|app_4[R]_{k_4}|sys[\mathsf{auth}(app_1, k_1)|\mathsf{auth}(app_4, k_4)|\mathsf{auth}(app_2, k_5)|\mathsf{auth}(app_2, k_5).T_3|pri[X]|pub[Y]]_{k_j}$$

$$\rightarrow app_1[app_3[R]|P]_{k_1}|app_2[Q]_{k_2}|app_4[R]_{k_4}|sys[\mathsf{auth}(app_1, k_1)|\mathsf{auth}(app_4, k_4)|\mathsf{auth}(app_2, k_5)|pri[X]|pub[Y]]_{k_j}$$

$$\rightarrow app_2[Q]_{k_2}|sys[app_1[app_3[R]]|\mathsf{auth}(app_2, k_5)|app_4[Q]|pri[X]|pub[Y]]_{k_j}$$

In this model we consider four applets namely $app1$ to $app4$. The resource T of Secure Systems can be considered to be a combination of a private resource($pri[X]$), public resource($pub[Y]$) and some additional processes. The model of the system using DSA is shown in the Figure 3.1.

In the first reduction the $app3$ is authenticated by $app1$. In the second reduction $app3$ and $app4$ are authenticated and hence have access to all the resources

$n[\mathsf{in}\ m.P\|Q]_s\|m[R]_\epsilon \to m[n[P\|Q]_s\|R]_\epsilon$	(Red In)
$m[P\|n[\mathsf{out}\ m.Q\|R]_s]_\epsilon \to m[P]_\epsilon\|n[Q\|R]_s$	(Red Out)
$\mathsf{open}\ n.P\|n[Q]_\epsilon \to P\|Q$	(Red Open)
$m[P]_k\|n[\mathsf{auth}(m,k).Q\|R]_s \to n[R\|Q\|m[P]_\epsilon]_s$	(Red Auth)
$\mathsf{sign}(m,k).P\|m[Q]_\epsilon \to P\|m[Q]_k$	(Red Sign)
$\dfrac{P \to Q}{(\nu n)P \to (\nu n)Q}$	(Red Res)
$\dfrac{P \to Q}{n[P]_s \to n[Q]_s}$	(Red Amb)
$\dfrac{P \to Q}{P\|R \to Q\|R}$	(Red Par)
$\dfrac{P' \equiv P, P \to Q, Q \equiv Q'}{P' \to Q'}$	(Red $\equiv$)

Table 3.5: The reduction Rules of DSA

within the ambient where as *app*2 cannot be authenticated as the key does not match. From the reductions above it is evident that digitally signed ambients can distinguish between trusted and untrusted code. There are a few problems which still persist such as:

- In the first reduction we can see that the ambient app_1 trusts the key k_3 of the ambient app_3 and hence authenticates it. There could be a case where the key k_3 is not trusted by the Secure System but we see from the reduction that even then app_3 ends up within system gaining access to all the system resources. This could be taken as a Trojan horse example where ambient app_1 acts as the horse.

- In the second reduction, we can see that the system trusts the key of app_1 and *app*4 which are allowed access to all the resources within the system, but this mechanism is not flexible as it is similar to the Java sandbox model of JDK1.1 where the applet is either given complete access to the resource or no access at all. We need to device a more flexible mechanism for access control on processes. That is to provide different access control for different processes based on their sources.

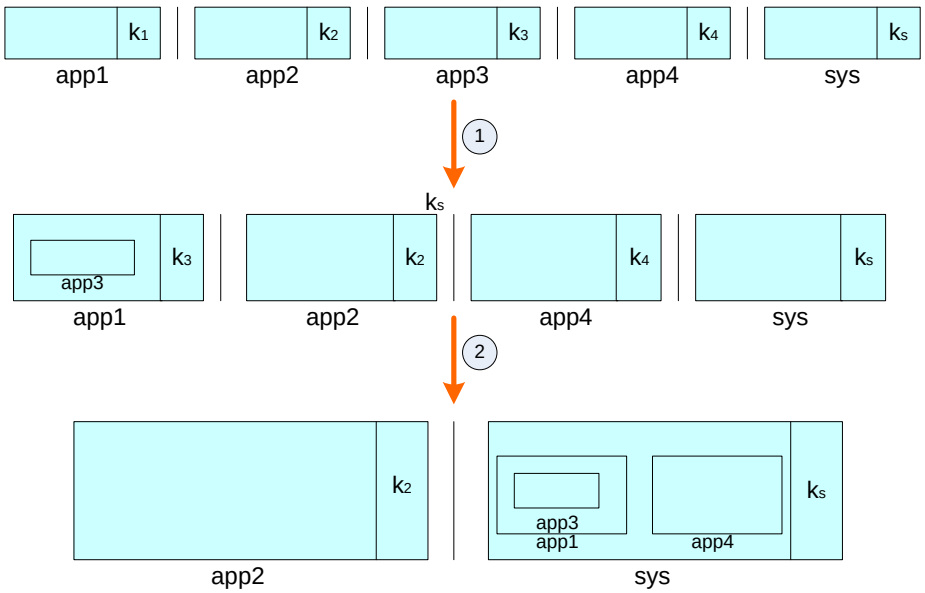

Figure 3.1: Modeling of Safe System using DSA

Security Concerns

Security is a major concern in mobile agents. For example in the following reduction:

$$\mathsf{open}\ n.P|n[Q]_\epsilon \rightarrow P|Q$$

An open operation dissolves the boundary of the ambient n. From the point of view of P, we cannot predict the behavior of Q when it is unleashed. From the point of view of Q, its environment is being ripped open. This is a security concern as there could be a case where the ambient n does not wish its boundary to be dissolved but there is no mechanism to prevent it. This is one of the reasons why we introduce the concept of *blocking* which will be described in the chapter 4.

Chapter 4

Blocking in Mobile Ambients

In this chapter, we introduce the three kinds of blockings into Mobile Ambients. In Section 4.1, the concept of blocking in Calculus for Communicating System(CCS) and π calculus is investigated; in Section 4.2, the calculus of Mobile Ambients with Blocking on Names is introduced; in Section 4.3, the calculus of Mobile Ambients with Blocking on Capabilities is introduced; in Section 4.4, the calculus of Mobile Ambients with Blocking at Ambient Level is introduced.

4.1 Blocking in Process Calculus

In CCS[9], the concept of blocking is introduced as *restriction*. The following is the syntax of CCS:

$$P ::= \ K \ \mid \ \alpha.P \ \mid \ P + P \ \mid \ P|Q \ \mid \ P[f] \ \mid \ P\backslash L$$

K is the process name; $\alpha.P$ is the prefix, that means P can be executed only after action α is executed; $P + P$ is the nondeterministic process; $P|Q$ is parallelism; $P[f]$ is relabeling, which is equivalent with substitution defined by f; $P\backslash L$ is restriction, which blocks any process out of P to use port names in L. The following is one example to represent the usage restriction[10]:

$$\begin{aligned} CM &\stackrel{def}{=} coin.\overline{coffee}.CM \\ CS &\stackrel{def}{=} \overline{pub}.\overline{coin}.coffee.CS \\ CS' &\stackrel{def}{=} \overline{pub}.\overline{coin}.coffee.CS \\ SmUni &\stackrel{def}{=} (CM|CS)\backslash coin\backslash coffee \end{aligned}$$

CM is a process that represents a coffee machine which accept a coin and then output a coffee; CS is a process that represents a computer scientist who is initially keen to produce a publication, then needs to get coffee from CM and make the next publication. CS' is another computer scientist who competes for the

resource coffee machine with CS inside process $CM|CS|CS'$. In order to make the coffee machine private for CS, the restriction on port names *coin* and *coffee* is used to block any process other than CM and CS to know these port names, therefore the port names *coin* and $coffee$ in CS' do not mean the same channels expressed by the same port names *coin* and *coffee* in $SmUni$.

In his PhD thesis [5], Vivas Frontana introduced the concept of blocking into π-calculus, which is different from the concept of restriction in π-calculus, but it is similar as the concept of restriction in *CCS*. The following is the syntax of the first-order π-calculus with blocking:

$$\begin{aligned} P ::= \quad & 0 \mid \overline{x}y.P \mid x(y).P \mid \tau.P \mid \nu(x)P \mid [x=y]P \mid \\ & P|P \mid P+Q \mid A(y_1,\cdots,y_n) \mid P\backslash z \end{aligned} \tag{4.1}$$

0 is the inactivity; $\overline{x}y.P$ is the output prefix, which sends the channel name y along the channel x before continuing as P; $x(y).P$ is the input prefix, which inputs an arbitrary name z from the channel x before continuing as $P\{z\backslash y\}$; $\tau.P$ is the prefix whose action is a silent action; $\nu(x)P$ denotes a process P in which occurrences of x in P are bound by $\nu(x)$; the match $[x = y]P$ acts as 0 if it is not the case that $x = y$, otherwise as P; $P|P$ is the parallelism; $P + Q$ is the summation which represents alternative choice; $A(y_1, \cdots, y_n)$ is the identifier which defines a process template.

The blocking operator in π-calculus is denoted by $P\backslash a$. It works as a firewall which prevents any kind of communication through the channel a between process P and its environment, but allows the communication of the channel name a across the firewall. Therefore the blocking operator merely restricts the communication capabilities of channel a by the means of name space management, that the name a appeared out of the scope of the blocking does not bind with the name a appeared within the scope of the blocking. In addition, the most significant difference between restriction and blocking is that the scope of restriction can be extruded through communication of the restricted name but the scope of blocking cannot be extruded in any way, even if the blocked names are leaked out of the blocking scope through communications. The following is an example to explain the above description:

$$P = \overline{x}a.a(y).R$$
$$Q = x(z).\overline{z}b.S$$

For process $\nu(a)P|Q$, it is structural equivalent with process $\nu(a)(P|Q)$. When channel name a is transferred from P to Q, the scope of the restriction is expanded to Q according to the semantic rules of π-calculus, which is a mechanism called extrusion of the restriction scope. $(\nu a)(P|Q)$ have the following reductions:

$$\nu(a)(\overline{x}a.a(y).R|x(z).\overline{z}b.S) \rightarrow \nu(a)(a(y).R|\overline{a}b.S\{a\backslash z\}) \rightarrow \nu(a)(R\{b\backslash y\}|S\{a\backslash z\})$$

On the contrary, process $P\backslash a|Q$ is not structurally equivalent to the process $(P|Q)\backslash a$. In the first reduction step, channel name a can go through the blocking through the communication along channel x, so that name a is leaked out of the blocking scope:

$$\overline{x}a.a(y).R\backslash a|x(z).\overline{z}b.S \rightarrow a(y).R\backslash a|\overline{a}b.S\{a\backslash z\}$$

After this step, the communication along a can not happen, because the blocking on name a divides the process into two different name spaces, where name a within the blocking scope cannot bind with name a out of the blocking scope.

We can see that the two calculi above have the similar constructs in terms of blocking, and they also have the similar constructs as mobile ambients, such as inactivity, parallelism and prefix, so that it is feasible to introduce the concept of blocking into the calculus of Mobile Ambients. In this chapter, we try to introduce the concept of blocking into the calculus of Mobile Ambients which is more complex than introducing blocking in CCS and π-calculus. In this chapter, we propose the three different calculi for blocking in the following sections: in Section 4.2, we introduce *Mobile Ambients with Blocking on Names*(MABN); in Section 4.3, we introduce *Mobile Ambients with Blocking on Capabilities*(MABC); in Section 4.4, we introduce *Mobile Ambients with Blocking defined at Ambient Level*(MABA).

4.2 Blocking on Names: MABN

In mobile ambients, any action in terms of movement has a subject and an object. For example, in $a[\mathsf{in}\ b]$, for the action $\mathsf{in}\ b$, ambient a is the subject, ambient b is the object, a takes the action on b, finally it will enter b. The blocking in π-calculus[11][5] prohibits any communication between processes within and out of the scope of blocking along any blocked channel name. Similarly the blocking on names in mobile ambients can prohibit the actions applied on the objects whose names are out of the scope of the blocking, and we denote $P\backslash a$ as name a is blocked on process P . The following is one example:

$$P = a[\mathsf{in}\ b]|b[0]$$
$$Q = (a[\mathsf{in}\ b])\backslash b|b[0]$$
$$R = a[\mathsf{in}\ b]|(b[0])\backslash b$$

In P, it is evident that a can enter b after $\mathsf{in}\ b$ is exercised. In Q, a cannot enter b, because ambient b outside the blocking scope cannot be referenced by name b within the blocking scope, that b within $\mathsf{in}\ b$ does not bind with ambient b. For the same reason, within R, the action $\mathsf{in}\ b$ cannot be taken by ambient a, name b

outside the blocking scope cannot bind with ambient b within the blocking scope. Therefore, we can say that blocking on names can make effect bi-directionally on processes within the scope and out of the scope.

This section introduces the calculus of MABN. The subsection 4.2.1, gives the formal definition of this calculus and its explanation. In subsection 4.2.2, some topics about this calculus are discussed.

4.2.1 Syntax and Semantics

Table 4.1 gives the formal definition of the syntax of MABA. It is assumed that an infinite set of names $\mathscr{N}$ ranges over ambient names. $P\backslash_N$ denotes that the ambient names belonging to N are blocked within the scope of process P. When set N has only one element such as n, it can be written as $P\backslash_n$. The other syntactic elements have the same meaning as the calculus of Mobile Ambients.

Table 4.2 shows the definition of free names in MABN. The rule $\text{fn}(P\backslash_N) \triangleq \text{fn}(P) \cup N$ tries to collect the free names in the blocking set.

Table 4.3 shows the syntactic conventions of MABN.

Table 4.4 gives the definition of structural congruence of MABN. Most of them are the same as in Mobile Ambients. The introduction of blocking operation brings some new rules which are explained as follows:

(Struct Block): Blocking has no effects on the internal events within the scope of the blocking.

(Struct Block Empty): A process without blocking is equivalent with the process with blocking of empty names set. Hence, general reduction rules for this calculus are possible.

(Struct Block Union): Two consecutive blockings can be combined with union of the two blocking name sets. This gives another rule: $P\backslash_N\backslash_{N'} \equiv P\backslash_{N'}\backslash_N$.

(Struct Zero Block): Any blocking on inactivity does not make any effect.

(Struct Block Par Symm): The blocking on names makes the same effect on bi-directions.

(Struct Res Block): If there is no shared name between restriction and blocking on names, they make no effect with each other.

n	names
N	sets of names
$P, Q ::=$	processes
$(\nu n)P$	restriction on ambient name
0	inactivity
$P \mid Q$	composition
$!\,P$	replication
$n[P]$	ambient
$P\backslash_N$	blocking on ambient names
$M.P$	action
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n

Table 4.1: The syntax of MABN

$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(\mathsf{in}\ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(P \mid Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{out}\ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\,P)$	$\triangleq$	$\mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{open}\ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(P\backslash_N)$	$\triangleq$	$\mathrm{fn}(P) \cup N$	$\mathrm{fn}(n[P])$	$\triangleq$	$\mathrm{fn}(P) \cup \{n\}$

Table 4.2: The free names of MABN

$P \mid Q\backslash_N$	is read	$P \mid (Q\backslash_N)$
$(\nu n)P \mid Q$	is read	$((\nu n)P) \mid Q$
$M.P \mid Q$	is read	$(M.P) \mid Q$
$(\nu n_1 \ldots n_m)P$	$\triangleq$	$(\nu n_1) \ldots (\nu n_m)P$
$n[]$	$\triangleq$	$n[0]$
$()\backslash_N$	$\triangleq$	$(0)\backslash_N$
M	$\triangleq$	$M.0$

Table 4.3: The syntactic conventions of MABN

$P \equiv P$	(Struct Refl)
$\dfrac{P \equiv Q}{Q \equiv P}$	(Struct Symm)
$\dfrac{P \equiv Q, Q \equiv R}{P \equiv R}$	(Struct Trans)
$\dfrac{P \equiv Q}{P\backslash_N \equiv Q\backslash_N}$	(Struct Block)
$\dfrac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$	(Struct Res)
$\dfrac{P \equiv Q}{P\|R \equiv Q\|R}$	(Struct Par)
$\dfrac{P \equiv Q}{!\,P \equiv !\,Q}$	(Struct Repl)
$\dfrac{P \equiv Q}{n[P] \equiv n[Q]}$	(Struct Amb)
$\dfrac{P \equiv Q}{M.P \equiv M.Q}$	(Struct Action)
$P\|Q \equiv Q\|P$	(Struct Par Comm)
$(P\|Q)\|R \equiv P\|(Q\|R)$	(Struct Par Assoc)
$!\,P \equiv P\|!\,P$	(Struct Repl Par)
$(\nu n)(\nu n^{'})P \equiv (\nu n^{'})(\nu n)P$	(Struct Res Res)
$(\nu n)(P\|Q) \equiv P\|(\nu n)Q$ if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\nu n)(n^{'}[P]) \equiv n^{'}[(\nu n)P]$ if $n \neq n^{'}$	(Struct Res Amb)
$P\|0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$!\,0 \equiv 0$	(Struct Zero Repl)
$P\backslash_\emptyset \equiv P$	(Struct Block Empty)
$P\backslash_N\backslash_{N'} \equiv P\backslash_{N \cup N'}$	(Struct Block Union)
$0\backslash_N \equiv 0$	(Struct Zero Block)
$(P)\backslash_N\|Q \equiv P\|(Q)\backslash_N$	(Struct Block Par Symm)
$(\nu n)(P\backslash_N) \equiv ((\nu n)P)\backslash_N$ if $n \notin N$	(Struct Res Block)

Table 4.4: The structural congruence of MABN

In addition, the following should be emphasized:

$$n[P\backslash_N] \not\equiv n[P]\backslash_N$$
$$P\backslash_N|Q\backslash_N \not\equiv (P|Q)\backslash_N$$

In $n[P\backslash_N]$, the blocking of N is carried on by ambient n as a part of ambient n. For example, $n[\mathsf{in}\ m.Q\backslash_N]|m[R]$ can be reduced to $m[R|n[Q\backslash_N]]$ in the case of $m \notin N$, the blocking is carried by ambient n. On the contrary, in $n[P]\backslash_N$, after ambient n exercises some capability in terms of movements, it can jump out of the blocking scope of N. For example, $n[\mathsf{in}\ m.Q]\backslash_N|m[R]$ can be reduced to $()\backslash_N|m[R|n[\mathsf{in}\ m.Q]]$. Hence blocking has different effects on an ambient within and out of it.

In $(P|Q)\backslash_N$, P can interact with Q through some names, even though some of these names belong to the blocking N, because P and Q are within the same scope of Blocking, according to (Red Block), this blocking makes no effect on the internal event within the scope of this blocking. On the contrary, for $P\backslash_N|Q\backslash_N$, P cannot interact with Q through names belonging to N.

Table 4.5 shows the reduction rules of MABN.

In order to express that a reduction is managed by multi-level blocks, the following *evaluation context* is defined:

Definition 4.2.1 (Evaluation Context)

$$\mathscr{C}^N(\!|\ |\!) ::= \ (\!|\ |\!)\backslash_N \ \mid \ (\mathscr{C}^{N'}(\!|\ |\!)|P)\backslash_{N''}(N = N' \cup N'') \tag{4.2}$$

where N, N' and N'' are three blocked name sets, P is an arbitrary process. $(\!|\ |\!)$ is a hole which can be filled up with any process, capability or another evaluation context.

From (Red In) (Red Out) (Red Open), an action can occur if there is no name blocking on its object along the way its subject moves. Additionally, the blocking scope is fixed relative to the ambient in which it resides, ambients can move into or out of blocking scopes by entering into or exit from ambients, and the blocking within one ambient has to be moved along with the ambient.

From (Red Block), we can see that, the blocking makes no effect on the event happening within the scope of the blocking.

$\mathscr{C}^N(\!\!\{\, a[\mathscr{C}^{N'}(\!\!\{\, \mathsf{in}\ b.P \,\}\!\!)]\,\vert Q \,\}\!\!)\vert\mathscr{C}^{N''}(\!\!\{\, b[R] \,\}\!\!)$ $\to \mathscr{C}^N(\!\!\{\, Q \,\}\!\!)\vert\mathscr{C}^{N''}(\!\!\{\, b[R\vert a[\mathscr{C}^{N'}(\!\!\{\, P \,\}\!\!)]] \,\}\!\!)(b \notin N \cup N' \cup N'')$	(Red In)
$b[\mathscr{C}^N(\!\!\{\, a[\mathscr{C}^{N'}(\!\!\{\, \mathsf{out}\ b.P \,\}\!\!)]\,\vert Q \,\}\!\!)] \to b[\mathscr{C}^N(\!\!\{\, Q \,\}\!\!)]\vert a[\mathscr{C}^{N'}(\!\!\{\, P \,\}\!\!)](b \notin N \cup N')$	(Red Out)
$\mathscr{C}^N(\!\!\{\, \mathsf{open}\ b.P \,\}\!\!)\vert\mathscr{C}^{N'}(\!\!\{\, b[Q] \,\}\!\!) \to \mathscr{C}^N(\!\!\{\, P \,\}\!\!)\vert\mathscr{C}^{N'}(\!\!\{\, Q \,\}\!\!)(b \notin N \cup N')$	(Red Open)
$\dfrac{P \to Q}{(\nu n)P \to (\nu n)Q}$	(Red Res)
$\dfrac{P \to Q}{n[P] \to n[Q]}$	(Red Amb)
$\dfrac{P \to Q}{P\vert R \to Q\vert R}$	(Red Par)
$\dfrac{P \to Q}{P\backslash_N \to Q\backslash_N}$	(Red Block)
$\dfrac{P' \equiv P, P \to Q, Q \equiv Q'}{P' \to Q'}$	(Red $\equiv$)

Table 4.5: The reduction rules of MABN

4.2.2 Discussion

The introduction of blocking on names into mobile ambients brings some new insights on name space management, and it is also worth comparing blocking with restriction. We will have some discussion on these topics in the following.

1. **Name Space Management and Dynamic Name Binding**

   Blocking on names can be considered as a mechanism of name space management and name dynamic binding. Name space management usually adopts a hierarchical structure to organize names to avoid name collision as *Domain Name System*[12] in Internet. Dynamic name binding is a pattern to associate identifiers or references to entities on runtime. An ambient name can appear in two forms: when n appears as $n[P]$, name n means an entity which is an ambient encapsulates a process; when n appears in a capability such as $\mathsf{in}\ n$, name n means a reference to some ambient. Therefore these two forms of names must be bound dynamically when an action can happen, for example, in $a[0]|b[c[\mathsf{in}\ a]|a[0]]$, when $\mathsf{in}\ a$ is executed, a inside $\mathsf{in}\ a$ is bound with ambient a inside ambient b, not ambient a outside ambient b. Blocking on names can split a process into two separated name spaces, but allow dynamic name binding on execution. For example,

   $$(\overset{1}{a}[\mathsf{in}\ n] \mid \overset{2}{n}[\mathsf{in}\ a])\backslash_n \mid \overset{3}{n}[\mathsf{in}\ a] \mid \overset{4}{a}[\mathsf{in}\ n]$$

   In the above process, the blocking splits the process into two name spaces in which name n can only bind with the entity in the same name space. Every ambient is labeled by a unique number in order to describe name dynamic binding clearly. In the following, we use $\rightarrow$ to express the feasibility of ambients movement, for example $(1) \rightarrow (2)$ means ambient a labeled by 1 can move into ambient n labeled by 2, $(1) \nrightarrow (3)$ means ambient a labeled by 1 can not move into ambient n labeled by 3. According to the semantics, the process has the following feasibilities of movement:

   $$\begin{array}{ll} (1) \rightarrow (2) & (1) \nrightarrow (3) \\ (2) \rightarrow (1) & (2) \rightarrow (4) \\ (3) \rightarrow (4) & (3) \rightarrow (1) \\ (4) \rightarrow (3) & (4) \nrightarrow (2) \end{array}$$

   From the left column, we can see that the blocking makes no effect on the actions within the blocking scope. From the right column, we can see that the movements by $\mathsf{in}\ n$ are blocked by the block on name n, but the blocking on name n makes no effect on the movements by $\mathsf{in}\ a$.

The most important point is: the blocking cannot prohibit (1) → (3) forever, this block can be overcome by name dynamic binding. Suppose that the process has the following starting configuration:

$$(a[\mathsf{in}\ n|\overset{1}{\mathsf{in}}\ a.\mathsf{out}\ a] \mid n[\overset{2}{\mathsf{in}}\ a])\backslash_n|\ n[\overset{3}{\mathsf{in}}\ a] \mid a[\overset{4}{\mathsf{in}}\ n]$$

It has the following reductions:

$$\begin{array}{ll} & (a[\mathsf{in}\ n|\overset{1}{\mathsf{in}}\ a.\mathsf{out}\ a] \mid n[\overset{2}{\mathsf{in}}\ a])\backslash_n|\ n[\overset{3}{\mathsf{in}}\ a] \mid a[\overset{4}{\mathsf{in}}\ n] \\ \\ \rightarrow & (n[\overset{2}{\mathsf{in}}\ a])\backslash_n|\ n[\overset{3}{\mathsf{in}}\ a] \mid a[\mathsf{in}\ n|\ a[\overset{4}{\mathsf{in}}\ n|\overset{1}{\mathsf{out}}\ a]] \\ \rightarrow & (n[\overset{2}{\mathsf{in}}\ a])\backslash_n|\ n[\overset{3}{\mathsf{in}}\ a] \mid a[\overset{4}{\mathsf{in}}\ n] \mid a[\overset{1}{\mathsf{in}}\ n] \\ \\ \rightarrow & (n[\overset{2}{\mathsf{in}}\ a])\backslash_n|\ n[\mathsf{in}\ a|\ \overset{3}{a}[\overset{1}{[]}] \mid a[\overset{4}{\mathsf{in}}\ n] \end{array}$$

Ambient a labeled by 1 can enter the name space where ambient a labeled by 4 through (1) → (4), so that ambient a is in the same name space with ambient n labeled by (3), ambient a labeled by (1) may have a chance to enter ambient n labeled by (3). When ambient a labeled by (1) exits from ambient a labeled by 4, name n in the capability $\mathsf{in}\ n$ of ambient a labeled by 1 binds with ambient n labeled by 3.

2. The Comparison between Blocking on Names and Restriction

Restriction is a mechanism to create a new name which can be used within the scope of the restriction. In contrast, blocking on names does not create new names but defines name spaces which limit dynamic name binding within the same name space. The most obvious difference between the scopes of restriction and blocking on names is: the scope of restriction can be extruded using (Struct Res Par); blocking on names is a fixed structure whose scope cannot be extruded. For example, $(\nu m)m[\mathsf{in}\ n]|n[0] \equiv (\nu m)(m[\mathsf{in}\ n]|n[0])$ so that the scope of the restriction is extruded from ambient m to ambients m and n, as a result $\mathsf{in}\ n$ can be exercised; $m[\mathsf{in}\ n]\backslash_m|n[0]$ is not structural congruent with $(m[\mathsf{in}\ n]|n[0])\backslash_m$ although $\mathsf{in}\ n$ can be exercised in both cases according to our reduction rules.

However, after the difference is explored more deeply, it can be found that restriction can prohibit dynamic name binding, but blocking on names allows dynamic name binding. It is explained by the following example:

$$P = (\nu b)(b[0]|m[0])|n[\mathsf{in}\ b|\mathsf{in}\ m.\mathsf{out}\ m]$$
$$Q = (b[0]|m[0])\backslash_{\{b\}}|n[\mathsf{in}\ b|\mathsf{in}\ m.\mathsf{out}\ m]$$

Process P can be converted to $(\nu b)(b[0]|m[0]|n[\mathsf{in}\ b'|\mathsf{in}\ m.\mathsf{out}\ m])$ using α-conversion with which $\mathsf{in}\ b$ becomes $\mathsf{in}\ b'$, since name b within ambient n is a free name. Therefore, ambient n can not enter ambient b in process P forever, and the restriction on name b prohibits this.

For process Q, it has the following reductions:

$$(b[0]|m[0])\backslash_{\{b\}}|n[\mathsf{in}\ b|\mathsf{in}\ m.\mathsf{out}\ m]$$

$$\begin{array}{ll} \rightarrow & (b[0]|m[n[\mathsf{in}\ b|\mathsf{out}\ m]])\backslash_{\{b\}} \\ \rightarrow & (b[0]|m[0]|n[\mathsf{in}\ b])\backslash_{\{b\}} \\ \rightarrow & (b[n[0]]|m[0])\backslash_{\{b\}} \end{array}$$

Although the scope of the blocking on name b is fixed, ambient n can enter the blocking scope by entering into ambient m which is not blocked, then name b in $\mathsf{in}\ b$ binds with ambient b, as a result it can enter ambient b. Therefore, ambient n can enter ambient b by circumventing through entering ambient m which is not blocked but in the same blocking scope as ambient b.

This comparison make it easier to answer the question why $(\nu a)(P\backslash_{\{a\}}) \equiv ((\nu a)P)\backslash_a$ does not hold. The reason is simply that creating a name within the scope of the blocking on the name is different from blocking a name after creating it.

4.3 Blocking on Capabilities: MABC

In this section we will introduce the calculus of mobile ambients with Blocking on Capabilities (MABC). In MABN, blocking on names prohibits the actions applied on the objects whose names are out of the scope of the blocking. In MABC, blocking on capabilities controls the access capability on the objects. The idea of block on capabilities is an inspiration from the concept of access control.

Access control[13] is a method of restricting access to resources, allowing only privileged entities access. For example, when we access a web resource, access can be granted or denied based on some criteria, such as the browser which we are using. Access control is analogous to having a bouncer at the entrance, it

controls entrance by some arbitrary condition which may or may not have anything to do with the attributes of the particular visitor.

We use $P\wr_C$ to express blocking on capabilities, where C is a set of abstract capabilities, $\wr_C$ can block the interaction taken by those capabilities belonging to C between process P and its environment. For example, consider the following process,

$$\mathsf{open}\ m.P \wr_{\mathsf{open}\ m} \mid m[Q]$$

In the above process, since open capability is blocked and the access can not be executed. Hence, ambient m can not be dissolved. Here the scope of blocking virtually represents the scope of access control.

4.3.1 Syntax and Semantics

Table 4.6 gives the formal definition of the syntax of MABC. It is assumed that an infinite set of names $\mathscr{N}$ ranges over ambient names.

Table 4.7 defines the set of free names $\mathrm{fn}(P)$ for MABC.

Table 4.8 shows the syntactic conventions of MABC.

Table 4.9 gives the structural congruence for MABC. We have the following five new structural congruence rules,

(Struct BC): Blocking has no effect on any processes that are structurally congruent.

(Struct Res BC): If the restricted name does not appear in the block set, the restriction and blocking do not interrupt each other.

(Struct BC Union): Two consecutive blocking can be combined with union of the two blocking capability sets.

(Struct BC Empty): Empty blocking set means no blocking within process.

n	names
$Cap = \{\mathsf{in}\ , \mathsf{out}\ , \mathsf{open}\ \}$	the set of capabilities
Amb	the complete ambient name set
$C \subseteq Cap \times Amb$	the blocking capability set
$P, Q ::=$	processes
$(\nu n)P$	restriction on ambient name
0	inactivity
$P \vert Q$	composition
$!\,P$	replication
$n[P]$	ambient
$M.P$	action
$P \wr_C$	blocking on capabilities
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n

Table 4.6: The syntax of MABC

$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(n[P])$	$\triangleq$	$\{n\} \cup \mathrm{fn}(P)$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(P \wr_C)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(C)$
$\mathrm{fn}(P \vert Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{in}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\,P)$	$\triangleq$	$\mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{out}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{open}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(C)$	$\triangleq$	$\bigcup_{M \in C} \mathrm{fn}(M)$			

Table 4.7: The free names of MABC

$P \wr_C \vert Q$	is read	$(P \wr_C) \vert Q$
$(\nu n)P \vert Q$	is read	$((\nu n)P) \vert Q$
$M.P \vert Q$	is read	$(M.P) \vert Q$
$(\nu n_1 \ldots n_m)P$	$\triangleq$	$(\nu n_1) \ldots (\nu n_m)P$
$n[]$	$\triangleq$	$n[0]$
$()\wr_C$	$\triangleq$	$(0)\wr_C$
M	$\triangleq$	$M.0$

Table 4.8: The syntactic conventions of MABC

$P \equiv P$	(Struct Refl)
$\dfrac{P \equiv Q}{Q \equiv P}$	(Struct Symm)
$\dfrac{P \equiv Q, Q \equiv R}{P \equiv R}$	(Struct Trans)
$\dfrac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$	(Struct Res)
$\dfrac{P \equiv Q}{P \mid R \equiv Q \mid R}$	(Struct Par)
$\dfrac{P \equiv Q}{!\,P \equiv !\,Q}$	(Struct Repl)
$\dfrac{P \equiv Q}{n[P] \equiv n[Q]}$	(Struct Amb)
$\dfrac{P \equiv Q}{M.P \equiv M.Q}$	(Struct Action)
$\dfrac{P \equiv Q}{P \wr_C \equiv Q \wr_C}$	(Struct BC)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$!\,P \equiv P \mid !\,P$	(Struct Repl Par)
$(\nu n)(\nu n')P \equiv (\nu n')(\nu n)P$	(Struct Res Res)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\nu n)(n'[P]) \equiv n'[(\nu n)P]$ if $n \neq n'$	(Struct Res Amb)
$(\nu n)(P \wr_C) \equiv ((\nu n)P) \wr_C$ if $n \notin \mathrm{fn}(C)$	(Struct Res BC)
$P \mid 0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$!\,0 \equiv 0$	(Struct Zero Repl)
$P \wr_C \wr_{C'} \equiv P \wr_{C \cup C'}$	(Struct BC Union)
$P \wr_\emptyset \equiv P$	(Struct BC Empty)
$0 \wr_C \equiv 0$	(Struct Zero BC)

Table 4.9: The structural congruence of MABC

$\mathscr{C}^C(\!\| n[\mathscr{C}^{C'}(\!\| \mathsf{in}\ m.P)\!\|] \mid Q)\!\| \mid \mathscr{C}^{C''}(\!\| m[R]\!\|)$ $\to \mathscr{C}^C(\!\| Q)\!\| \mid \mathscr{C}^{C''}(\!\| m[R \mid n[\mathscr{C}^{C'}(\!\| P)\!\|]]\!\|)$	($\mathsf{in}\ m \notin C \cup C'$)	(Red In)
$m[\mathscr{C}^C(\!\| n[\mathscr{C}^{C'}(\!\| \mathsf{out}\ m.P)\!\|] \mid Q)\!\|]$ $\to m[\mathscr{C}^C(\!\| Q)\!\|] \mid n[\mathscr{C}^{C'}(\!\| P)\!\|]$	($\mathsf{out}\ m \notin C \cup C'$)	(Red Out)
$\mathscr{C}^C(\!\| \mathsf{open}\ m.P)\!\| \mid \mathscr{C}^{C'}(\!\| m[Q]\!\|)$ $\to \mathscr{C}^C(\!\| P)\!\| \mid \mathscr{C}^{C'}(\!\| Q)\!\|$	($\mathsf{open}\ m \notin C$)	(Red Open)
$\dfrac{P \to Q}{(\nu n)P \to (\nu n)Q}$		(Red Res)
$\dfrac{P \to Q}{n[P] \to n[Q]}$		(Red Amb)
$\dfrac{P \to Q}{P\|R \to Q\|R}$		(Red Par)
$\dfrac{P \to Q}{P\wr_C \to Q\wr_C}$		(Red BC)
$\dfrac{P' \equiv P, P \to Q, Q \equiv Q'}{P' \to Q'}$		(Red $\equiv$)

Table 4.10: The reduction rules of MABC

(Struct Zero BC): The blocking on inactivity does not make any effect.

Since there are multi-level of blocking on capability, for example, $n[P\wr_C]$, $(P \wr_C | n[Q])\wr_{C'}$. Hence, in order to express the reduction rules accurately, we introduce the context of Blocking on Capabilities, written as $\mathscr{C}^C(\!\| \,)\!\|$.

Definition 4.3.1 (The context of Blocking on Capabilities) . *Let C, C' and C" be blocked sets of capability, where C = C' ∪ C", P is a process, $\mathscr{C}^C(\!\| \,)\!\|$ denotes the context of blocking on capability, we write $\mathscr{C}^C(\!\| \,)\!\| ::= (\!\| \,)\!\|\wr_C \big| (\mathscr{C}^{C'}(\!\| \,)\!\| | P)\wr_{C''}$ to express multi-level blocking context in MABC. If C is equal to ∅, it means no blocking.*

We can think $(\!\| \,)\!\|$ as a hole which can be filled up with anything, like processes, capabilities and another context.

Table 4.10 shows the all reduction rules for MABC.

There are four new reduction rules, namely, (Red In), (Red Out), (Red Open) and (Red BC).

The first three rules express the reduction happening in multi-level blocking contexts. If all of the blocked sets of capability are empty, then these three rules are the same as the general reduction rules for in , out and open in mobile ambients.

In (Red In), if in $m \notin C \cup C'$, then ambient n can enter ambient m, as the result, ambient n leave block C, block C' will be moved along with n; block C'' does not control this movement.

In (Red Out), if out $m \notin C \cup C'$, then ambient n can exit ambient m, as the result, ambient n leave block C, block C' will be moved along with n.

In (Red Open), if open $m \notin C$, after ambient m is dissolved, block C' on process Q is reserved.

In (Red BC), blocking has no effect on internal event within the blocking scope.

4.3.2 The Comparison between MABN and MABC

In MABN, the effect of blocking on names is name hiding. If a process accesses some external resources, it must know the names of the resources it tries to visit. In MABC, blocking on capabilities is a mechanism of the process based mandatory access control. blocking on capabilities is used to apply restriction on processes. Blocking on names more relax than blocking on capabilities.

We will consider the following process which illustrates what is difference between blocking on names and blocking on capabilities.

$$(\overset{(1)}{a}[\mathsf{in}\ n] \mid \overset{(2)}{n}[\mathsf{in}\ a])\ \wr_{\mathsf{in}\ n} \mid \overset{(3)}{n}[\mathsf{in}\ a] \mid \overset{(4)}{a}[\mathsf{in}\ n]$$

We have four ambients which try to exercise in capability, and in n capability is blocked within the scope of $(a[\mathsf{in}\ n]|n[\mathsf{in}\ a])$. We will use notations (1), (2), (3), (4) instead of four ambients respectively.

In this example, only one different from the example in section 4.2.2, we use $\wr_{\mathsf{in}\ n}$ instead of $\backslash_n$. In the previous example, the process has the following feasibilities of movement:

$$(1) \rightarrow (2) \quad (1) \nrightarrow (3)$$
$$(2) \rightarrow (1) \quad (2) \rightarrow (4)$$
$$(3) \rightarrow (4) \quad (3) \rightarrow (1)$$
$$(4) \rightarrow (3) \quad (4) \nrightarrow (2)$$

In this case, we have the following behavior of processes can happen or can not happen:

$$(1) \rightarrow (2) \quad (1) \nrightarrow (3)$$
$$(2) \rightarrow (1) \quad (2) \rightarrow (4)$$
$$(3) \rightarrow (4) \quad (3) \rightarrow (1)$$
$$(4) \rightarrow (3) \quad (4) \rightarrow (2)$$

In both cases, we see the last reduction in the right column, in the first case, it blocks any movement in terms of n. In the second case, $\mathsf{in}\ n$ capability is blocked within its scope, it can not be exercised from scope to outside, on the contrary, it is unblocked from outside to its scope.

In MABN, the rule (Struct Block Par Symm),$P\backslash_N|Q \equiv P|Q\backslash_N$ expresses the blocking on names makes the same effect on bi-directions. In MABC, a remarkable is $P \wr_C |Q \not\equiv P|Q\wr_C$. The blocking on capabilities only make effect on one direction.

4.4 Blocking at Ambient level: MABA

Mobile Ambients with Blocking at Ambient level (MABA) is one of the three kinds of blockings available for the calculus of Mobile Ambients. The idea of blocking at the ambient level is similar to the blocking on capability but is applied directly on the ambient. That is ambients are assigned a set of blockings which must be respected by the ambient during its movement. This kind of blocking moves along with the ambient and can be dissolved in case the ambient is opened. The blocking at ambient level is denoted by the notation $n[P]^{[C]}$, where C represents the capabilities that are blocked within the ambient boundary. Consider the following example:

$$n[\mathsf{in}\ m.P]^{[\mathsf{in}\ a]}|m[\mathsf{in}\ c|Q]^{[\mathsf{in}\ c]}|c[R]^{[\mathsf{in}\ b]} \rightarrow m[\mathsf{in}\ c|Q|n[P]]^{[\mathsf{in}\ c]}|c[R]^{[\mathsf{in}\ b]}$$

In the above example ambient n can enter an ambient m, but the ambient m cannot enter ambient c. Therefore, we can say that blocking on ambient level effects all the processes contained within that ambient.

n	names
$Cap = \{\mathsf{in}\ , \mathsf{out}\ , \mathsf{open}\ \}$	
$C \subseteq Cap \times Amb$	
$P, Q ::=$	processes
$(\nu n)P$	restriction
0	inactivity
$P \mid Q$	composition
$!\,P$	replication
$n[P]^{[C]}$	blocking at ambient level
$M.P$	action
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n

Table 4.11: The syntax of MABA

4.4.1 Syntax and Semantics

Table 4.11 give the formal definition of the syntax of MABA. It is assumed that an infinite set of names $\mathscr{N}$ ranges over n, where n are the ambient name. C denotes a set of capabilities which are blocked within the boundary of an ambient. The other syntactic elements have the same meaning as the calculus of mobile ambients.

Table 4.12 shows the definition of free names in MABA.

$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(\mathsf{in}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(\mathsf{out}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(P \mid Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{open}\ \ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\,P)$	$\triangleq$	$\mathrm{fn}(P)$	$\mathrm{fn}(C)$	$\triangleq$	$\bigcup_{M \in C} \mathrm{fn}(\alpha)$
$\mathrm{fn}(n[P]^{[C]})$	$\triangleq$	$\{n\} \cup \mathrm{fn}(P) \cup \mathrm{fn}(C)$			
$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$			

Table 4.12: The free names of MABA

$(\nu n)P\|Q$	is read	$((\nu n)P)\|Q$
$M.P\|Q$	is read	$(M.P)\|Q$
$(\nu n_1 \ldots n_m)P$	$\triangleq$	$(\nu n_1)\ldots(\nu n_m)P$
$n[]^{[C]}$	$\triangleq$	$n[0]^{[C]}$
$n[P]^{[\varepsilon]}$	$\triangleq$	$n[P]$
$n[]^{[\varepsilon]}$	$\triangleq$	$n[0]$
M	$\triangleq$	$M.0$

Table 4.13: The syntactic conventions of MABA

Table 4.13 shows the syntactic conventions of MABA.

Table 4.14 gives the definition of structural congruence of MABA. Most of them are the same as that of mobile ambients. The introduction of blocking operation brings some new rules which are explained in the Table 4.14:

(Struct Amb): If two equivalent processes are enclosed within the same ambient having the same blocking then they will have the same behavioural properties.

(Struct Res Amb): If the restricted name is not the name of the ambient on which there is a blocking, and the name is not contained in the blocking set then, they do not make effect on each other(that is it does not matter if the restriction is outside or within the ambient).

The behavior of a processes is given by its reduction rule. Table 4.15 shows the reduction rules of MABA.

From (Red In) and (Red Out), the blocking is carried along with the ambient. This kind of blocking is effective on all the processes within that ambient only on the first level i.e, it prohibits the capabilities from being exercised at the process level directly enclosed within the ambient and not on its subambients. From (Red Open), we can see that, the blocking dissolves along with the ambient boundary in case an ambient is opened.

4.4.2 Discussion

The calculus of MABA is different from the calculus of MABC in the sense that the blocking is applied directly on the ambient as a result of which the blocking can be carried along with an ambient. Hence we can say that the blocking at ambient level is a self constraint placed on an ambient. Hence, it can prevent

$P \equiv P$	(Struct Refl)
$\dfrac{P \equiv Q}{Q \equiv P}$	(Struct Symm)
$\dfrac{P \equiv Q, Q \equiv R}{P \equiv R}$	(Struct Trans)
$\dfrac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$	(Struct Res)
$\dfrac{P \equiv Q}{P \mid R \equiv Q \mid R}$	(Struct Par)
$\dfrac{P \equiv Q}{!\,P \equiv !\,Q}$	(Struct Repl)
$\dfrac{P \equiv Q}{n[P]^{[C]} \equiv n[Q]^{[C]}}$	(Struct Amb)
$\dfrac{P \equiv Q}{M.P \equiv M.Q}$	(Struct Action)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$!\,P \equiv P \mid !\,P$	(Struct Repl Par)
$(\nu n)(\nu n^{'})P \equiv (\nu n^{'})(\nu n)P$	(Struct Res Res)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\nu n)(n^{'}[P]^{[C]}) \equiv n^{'}[(\nu n)P]^{[C]}$ if $n \neq n^{'}$ and $n \notin fn(C)$	(Struct Res Amb)
$P \mid 0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$!\,0 \equiv 0$	(Struct Zero Repl)

Table 4.14: The structural congruence of MABA

$n[\mathsf{in}\ m.P]^{[C]}\|m[Q]^{[C']} \rightarrow m[n[P]^{[C]}\|Q]^{[C']}$ ($\mathsf{in}\ m \notin C$)	(Red In)
$m[n[\mathsf{out}\ m.P]^{[C]}\|Q]^{[C']} \rightarrow m[Q]^{[C']}\|n[P]^{[C]}$ ($\mathsf{out}\ m \notin C$)	(Red Out)
$\mathsf{open}\ m.P\|m[Q]^{[C]} \rightarrow P\|Q$	(Red Open)
$\dfrac{P \rightarrow Q}{(\nu n)P \rightarrow (\nu n)Q}$	(Red Res)
$\dfrac{P \rightarrow Q}{n[P]^{[C]} \rightarrow n[Q]^{[C]}}$	(Red Amb)
$\dfrac{P \rightarrow Q}{P\|R \rightarrow Q\|R}$	(Red Par)
$\dfrac{P' \equiv P, P \rightarrow Q, Q \equiv Q'}{P' \rightarrow Q'}$	(Red $\equiv$)

Table 4.15: The reduction rules of MABA

some harmful code from being executed within an ambient even if it manages to enter it. There are also a few security concerns. One of the major security concerns of MABA is that it cannot prevent the harmful execution of processes effectively.

Example 4.4.1 *Consider the following reduction:*

$$d[a[\mathsf{open}\ b]^{[\mathsf{in}\ b]}\|b[\mathsf{in}\ c\|Q]^{[\mathsf{in}\ c]}]^{[\mathsf{open}\ m]}\|c[P]^{[\mathsf{open}\ a]} \xrightarrow{*} c[d[a[]^{[\mathsf{in}\ b]}\|Q]^{[\mathsf{open}\ m]}]^{[\mathsf{in}\ d]}$$

In Example 4.4.1, we can observe that the capability $\mathsf{in}\ c$ was blocked at ambient level of ambient b but, as ambient b was opened the blocking was dissolved along with the ambient boundary as a result the capability $\mathsf{in}\ c$ was executed and the process Q within ambient b was able to enter the ambient c. This is a major security concern of the calculus of MABA.

Chapter 5

Digitally Signed Ambients with Blocking on Capabilities

The calculus of DSA with Blocking on capability(DSABC) was developed by combining the calculi of DSA and Blocking on capabilities. The main motive behind this was to model the concept of digital signatures and access control on process migration. The two pre existing models based on similar lines that served as a motivation were the Sandbox model for Java2 and Umbrella.

5.1 Security Models

The two security models that were analyzed before creation of the calculus of DSABC were the Sandbox model for Java2 and Umbrella model for Security in Linux.

Sandbox Model for Java2

The architecture of the sandbox model[3] was further evolved in the Java2 Platform Security Architecture. In this model the local code is subjected to the same security control as applets from an untrusted source. The difference being that the policy on the local code is more liberal than the remote code, thus enabling such code to run effectively as totally trusted. The same principle applies to signed applets and any Java application. That is a signed applet is subjected to some security control based on its source which can be its key. It is shown in the Figure 5.1

The main features of this model are:

1. Fine-grained access control.
2. Easily configurable security policy.

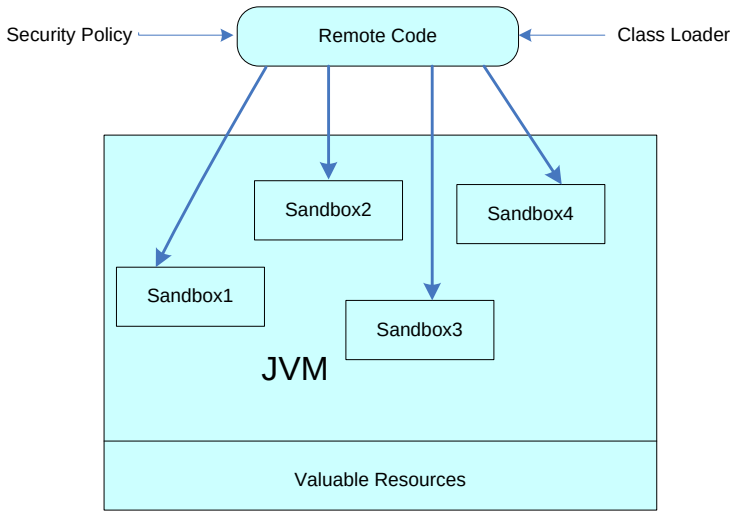

Figure 5.1: Java2 Platform Security Model

3. Easily extensible access control structure.

4. Extension of security checks to all Java programs.

Umbrella

Umbrella project was developed with an intention of providing more security to the Linux platform. It carries out a combination of process based mandatory access control (MAC) and authentication of files for Linux. In the MAC scheme restrictions are enforced on individual processes. The Umbrella security server first authenticates the signature of the file before allowing it to enter the system. If the signature is valid then it enforces the restrictions that are included along with the signature of that file. In the MAC scheme restrictions are enforced on individual processes. If the file cannot be authenticated then it is assigned a default set of restrictions. This eases the introduction of restrictions on the system. The working of the Umbrella model[6] is shown in the Figure 5.2:
The two improvement areas provided by Umbrella were:

1. Discretionary access control (DAC) mechanism was supplemented by introducing the concept of mandatory access control(MAC). In MAC structure as a tree, the children have at least all the restrictions of its parent. It avoids the need for manual setting of restrictions for all programs in the system.

2. Integrity of executables was implemented by introducing vendor-signing of executable files.

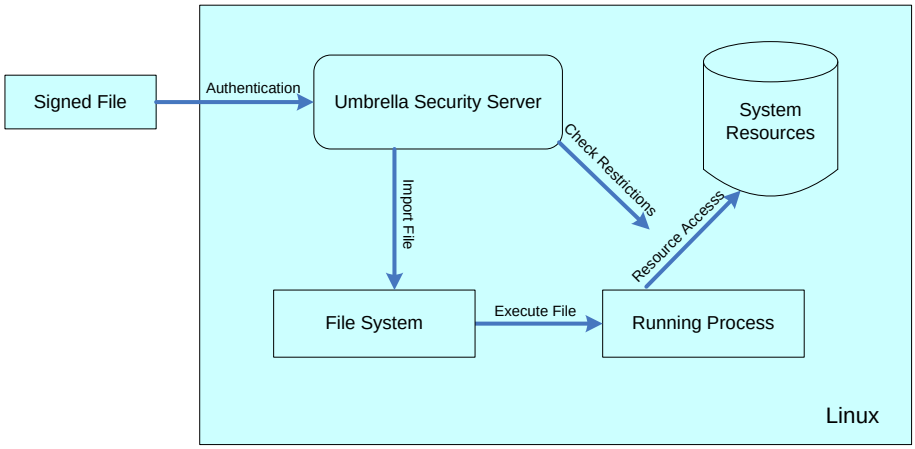

Figure 5.2: The Umbrella Model

DSA with Blocking on Capability(DSABC)

The calculus of DISA is different from the Sandbox model and Umbrella in the following ways:

- In the sandbox model for Java2 the system assigns the restriction based on the source of the code, in Umbrella the restrictions are included along with the key and if there are no restrictions included or the key cannot be authenticated then a default set of restrictions are applied on the processes where as in DSABC a set of minimal blocking are associated with the group of that ambient which must be respected during its movement.

- In sandbox model for Java2 and Umbrella a default set of blocking is applied in case the signature cannot be verified but in DSABC the code is not allowed to enter the system in case the authentication fails. That is signed ambients can enter other signed ambients only in case they can be authenticated.

5.2 Syntax and Semantics

Table 5.1 shows the syntax of DSABC. All the definitions have the same meaning as described in the calculi of DSA and MABC.

Table 5.2 defines the set of free names of DSABC.

Table 5.3 shows the syntactic conventions of DSABC.

n	names
k	key
$s = \varepsilon \mid k$	signature
$Cap = \{\mathsf{in}\ , \mathsf{out}\ , \mathsf{open}\ \}$	the set of three capabilities
Amb	the complete ambient name set
$C \subseteq Cap \times Amb$	the blocking capability set
$P, Q ::=$	processes
$(\nu n)P$	restriction
0	inactivity
$P \mid Q$	composition
$!\,P$	replication
$n[P]_s$	signed ambient
$M.P$	action
$P \wr_C$	blocking on capabilities
$M ::=$	capabilities
$\mathsf{in}\ n$	can enter n
$\mathsf{out}\ n$	can exit n
$\mathsf{open}\ n$	can open n
$\mathsf{auth}(m, k)$	can authenticate m with key k
$\mathsf{sign}(m, k)$	can sign m with key k

Table 5.1: The syntax of DSABC

$\mathrm{fn}((\nu n)P)$	$\triangleq$	$\mathrm{fn}(P) - \{n\}$	$\mathrm{fn}(\varepsilon)$	$\triangleq$	$\emptyset$
$\mathrm{fn}(0)$	$\triangleq$	$\emptyset$	$\mathrm{fn}(k)$	$\triangleq$	$\{k\}$
$\mathrm{fn}(P \mid Q)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(Q)$	$\mathrm{fn}(\mathsf{in}\ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(!\,P)$	$\triangleq$	$\mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{out}\ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(n[P]_s)$	$\triangleq$	$\mathrm{fn}(P) \cup \{n\} \cup \mathrm{fn}(s)$	$\mathrm{fn}(\mathsf{open}\ n)$	$\triangleq$	$\{n\}$
$\mathrm{fn}(M.P)$	$\triangleq$	$\mathrm{fn}(M) \cup \mathrm{fn}(P)$	$\mathrm{fn}(\mathsf{auth}(m, k))$	$\triangleq$	$\{m, k\}$
$\mathrm{fn}(P \wr_C)$	$\triangleq$	$\mathrm{fn}(P) \cup \mathrm{fn}(C)$	$\mathrm{fn}(\mathsf{sign}(m, k))$	$\triangleq$	$\{m, k\}$
$\mathrm{fn}(C)$	$\triangleq$	$\bigcup_{M \in C} \mathrm{fn}(M)$			

Table 5.2: The free names of DSABC

$P \wr_C \mid Q$	is read	$(P \wr_C) \mid Q$
$(\nu n)P \mid Q$	is read	$((\nu n)P) \mid Q$
$M.P \mid Q$	is read	$(M.P) \mid Q$
$(\nu n_1 \ldots n_m)P$	$\triangleq$	$(\nu n_1) \ldots (\nu n_m)P$
$n[P]_\epsilon$	$\triangleq$	$n[P]$
$n[]_s$	$\triangleq$	$n[0]_s$
$()\wr_C$	$\triangleq$	$(0)\wr_C$
M	$\triangleq$	$M.0$

Table 5.3: The syntactic conventions of DSABC

The structural congruence of DSABC inherits all the rules from the calculus of MABC. The only difference is in the rule (Struct Amb), where signed ambients are used instead of normal ambients. Table 5.4 describes all the structural congruence rules.

Similarly, we have the form of the evaluation context, $\mathscr{C}^C(\!|\ |\!) ::= (\!|\ |\!) \wr_C \mid (\mathscr{C}^{C'}(\!|\ |\!)|P)\wr_{C''}$, where $C = C' \cup C''$. The Reduction rules of DSABC inherits seven rules from the calculi of DSA and MABC, Table 5.5 shows all the reduction rules.

It is important to notice that the rules, (Red In), (Red Out), (Red Auth). If all of the blocked sets of capabilities are empty, then these three rules are the same as the reduction rules for in , out and auth in DSA.

In (Red In), if in $m \notin C \cup C'$, then signed ambient n can enter unsigned ambient m, ambient n leave block C, block C' will be moved along with n; block C'' has no effect on this movement. (Red In) expresses only the signed ambient can enter the unsigned ambient.

In (Red Out), if out $m \notin C \cup C'$, then signed ambient n can exit unsigned ambient m, signed ambient n leaves block C, block C' will be moved along with n. (Red Out) expresses only the signed ambient can exit the unsigned ambient.

In (Red Auth), For the authentication to occur, the authenticating ambient should be within the same blocking level as the signed ambient to be authenticated. After the authentication, the authenticated ambient becomes an unsigned ambient and is assigned the same blocking scope as the auth capability.

(Red Open), this rules is more strict than the in MABC. It expresses that

$P \equiv P$	(Struct Refl)
$\dfrac{P \equiv Q}{Q \equiv P}$	(Struct Symm)
$\dfrac{P \equiv Q, Q \equiv R}{P \equiv R}$	(Struct Trans)
$\dfrac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}$	(Struct Res)
$\dfrac{P \equiv Q}{P\|R \equiv Q\|R}$	(Struct Par)
$\dfrac{P \equiv Q}{!\,P \equiv !\,Q}$	(Struct Repl)
$\dfrac{P \equiv Q}{n[P]_s \equiv n[Q]_s}$	(Struct Amb)
$\dfrac{P \equiv Q}{M.P \equiv M.Q}$	(Struct Action)
$\dfrac{P \equiv Q}{P\wr_C \equiv Q\wr_C}$	(Struct BC)
$P\|Q \equiv Q\|P$	(Struct Par Comm)
$(P\|Q)\|R \equiv P\|(Q\|R)$	(Struct Par Assoc)
$!\,P \equiv P\|!\,P$	(Struct Repl Par)
$(\nu n)(\nu n')P \equiv (\nu n')(\nu n)P$	(Struct Res Res)
$(\nu n)(P\|Q) \equiv P\|(\nu n)Q$ if $n \notin \mathrm{fn}(P)$	(Struct Res Par)
$(\nu n)(n'[P]) \equiv n'[(\nu n)P]$ if $n \neq n'$	(Struct Res Amb)
$(\nu n)(P\wr_C) \equiv ((\nu n)P)\wr_C$ if $n \notin \mathrm{fn}(C)$	(Struct Res BC)
$P\|0 \equiv P$	(Struct Zero Par)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$!\,0 \equiv 0$	(Struct Zero Repl)
$P\wr_C \wr_{C'} \equiv P\wr_{C \cup C'}$	(Struct BC Comb)
$P\wr_\emptyset \equiv P$	(Struct BC Empty)
$0\wr_C \equiv 0$	(Struct Zero BC)

Table 5.4: The structural congruence of DSABC

$\mathscr{C}^C(\!\| n[\mathscr{C}^{C'}(\!\| \mathsf{in}\ m.P)\!\|]_s \mid Q)\!\| \mid \mathscr{C}^{C''}(\!\| m[R])\!\|$ $\quad \to \mathscr{C}^C(\!\| Q)\!\| \mid \mathscr{C}^{C''}(\!\| m[R \mid n[\mathscr{C}^{C'}(\!\| P)\!\|]_s])\!\|$ $\quad (\mathsf{in}\ m \notin C \cup C')$	(Red In)
$m[\mathscr{C}^C(\!\| n[\mathscr{C}^{C'}(\!\| \mathsf{out}\ m.P)\!\|]_s \mid Q)\!\|]$ $\quad \to m[\mathscr{C}^C(\!\| Q)\!\|] \mid n[\mathscr{C}^{C'}(\!\| P)\!\|]_s \quad (\mathsf{out}\ m \notin C \cup C')$	(Red Out)
$\mathsf{open}\ m.P \mid m[Q] \to P \mid Q$	(Red Open)
$m[P]_k \mid n[\mathscr{C}^C(\!\| \mathsf{auth}(m,k).Q)\!\| \mid R]_s \to n[\mathscr{C}^C(\!\| Q \mid m[P])\!\| \mid R]_s$	(Red Auth)
$\mathsf{sign}(m,k).P \mid m[Q]_\epsilon \to P \mid m[Q]_k$	(Red Sign)
$\dfrac{P \to Q}{(\nu n)P \to (\nu n)Q}$	(Red Res)
$\dfrac{P \to Q}{n[P]_s \to n[Q]_s}$	(Red Amb)
$\dfrac{P \to Q}{P \mid R \to Q \mid R}$	(Red Par)
$\dfrac{P \to Q}{P\wr_C \to Q\wr_C}$	(Red BC)
$\dfrac{P' \equiv P, P \to Q, Q \equiv Q'}{P' \to Q'}$	(Red $\equiv$)

Table 5.5: The reduction rules of DSABC

our system only allows open operations to occur at the same blocking level.

5.3 Advantages

The hybrid calculus of DSABC is advantageous as the property of *digital signature* prevents the malicious code from being downloaded and run on the client machine. The property of *blocking* ensures that the downloaded code is run with some restrictions which ensure that it does not harm the system. Hence the combination of the calculi of DSA and MABC we can ensure the properties of trust provided by digital signature and access control provided by blocking on capabilities. As a result of which the system can provide selective access control that is assigning different access rights to different processes depending on the key which makes the calculus more robust. Codes coming from trusted sources are also given some restrictions. This model is similar to the Java sandbox model in Java2 where even the trusted code are given some restrictions but which are more liberal than the ones laid on untrusted code. In the system that calculus of DSABC the untrusted code is not allowed to enter the system at all which ensures a very high level of security. Every code is assigned a different set of restrictions similar to the sandbox model where the different pieces of code coming from the various sources are assigned different sandboxes and are executed separately.

Example 5.3.1 *Modeling of the Safe System using DSABC:*

$$app_1[\mathsf{auth}(app_3, k_3)|P]_{k_1}|app_2[Q]_{k_2}|app_3[\mathsf{out}\ app_1.\mathsf{in}\ pub.\mathsf{out}\ pub.\mathsf{in}\ pri|R_1]_{k_3}|$$
$$app_4[S]_{k_4}|sys[(\mathsf{auth}(app_1, k_1))\ \wr_C\ |(\mathsf{auth}(app_4, k_4))\ \wr_{C'}\ |pri[X]|pub[Y]]_{k_s}$$

$$\rightarrow app_1[app_3[\mathsf{out}\ app_1.\mathsf{in}\ pub.\mathsf{out}\ pub.\mathsf{in}\ pri|R_1]|P]_{k_1}|app_2[Q]_{k_2}|app_4[S]_{k_4}|$$
$$sys[(\mathsf{auth}(app_1, k_1))\ \wr_C\ |(\mathsf{auth}(app_4, k_4))\ \wr_{C'}\ |pri[X]|pub[Y]]_{k_s}$$

$$\rightarrow app_2[Q]_{k_2}|sys[(app_1[app_3[\mathsf{out}\ app_1.\mathsf{in}\ pub.\mathsf{out}\ pub.\mathsf{in}\ pri|R_1]|P])\ \wr_C\ |$$
$$(app_4[S])\ \wr_{C'}\ |pri[X]|pub[Y]]_{k_s}$$

In this we further evolve the example and model the system using the concept of blocking on capabilities. The diagrammatic representation of the system is shown in Figure 5.3.

In the first reduction, $app3$ is authenticated by $app1$. In the second reduction, the authenticated applets app_1 and app_4 are assigned different capability blocking corresponding to their source. The source of an ambient is identified using the key with which it is signed. This resolves the issue of access control on processes. The following are the security issues which are not covered by the calculus of DSABC :

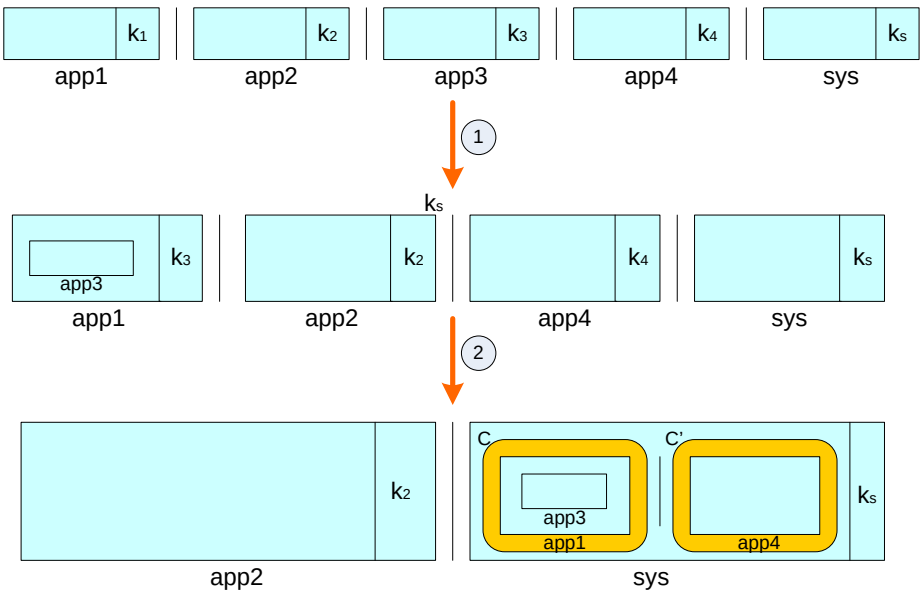

Figure 5.3: Modeling Safe System using DSABC

- The problem of app_3 using app_1 as a Trojan horse to enter the system still persists. A malicious process is still being executed within the system.
- The second problem is regarding key distribution. If ambient app_1 signs some processes and releases the public key, how does system know that app_1 is the right signatory and not an impostor?
- The third problem is regarding minimal blocking. When an ambient enters another ambient its minimal blocking policy should be respected. In the example above when the ambient app_1 enters into the Safe System, its minimum blocking policy

To resolve this kind of security issues that may arise, the type system of DSABC is proposed.

Chapter 6

A Type System for DSABC

In this chapter, we will introduce a type system for DSABC which captures three safety properties: who visit whom, key distribution, and the minimal blocking policy for ambients. In Section 6.1, the general characteristics of type systems is introduced; in Section 6.2, the three safety properties that our type system cares for are introduced; in Section 6.3, the formal definition of the type system and the explanation on typing rules are given; in Section 6.4, the semantic soundness and type safety are given.

6.1 General Overview of Type Systems

A type system[14] is used to classify values and variables into different types and describe their way in which they interact and are manipulated. The basic purpose of a type system is to indicate a set of values that have the same meaning or purpose. Thus it prevents the execution errors from occurring during the program execution. A program is supposed to run according to some constraints which are specified in the type system. The compiler of a programming language is constructed based on its type system which ensures that the program is type safe.

Type systems can be used to do type checking. Type checking is the process of verifying and enforcing the constraints predefined on types of values or processes. There are two kinds of type checking:

1. Static Typing: In this way, type checking is performed at compilation time. Static type checking is the primary task of the semantic analysis carried out by a compiler.

2. Dynamic Typing: In this way, type checking is performed during the runtime of a program. During the runtime, all type information must be col-

lected, and all changes on type information must be monitored and analyzed.

Type systems can provide the following advantages:

1. Safety: The compiler can detect invalid code using a type system.
2. Optimization: More information can be provided to the compiler using static type checking.
3. Documentation: Type systems which are more expressive can serve as a level of documentation.
4. Abstraction: It allows the programmers to think at a higher level and not consider the lower level implementation details.

6.2 The Ideas of Type System for DSABC

DSABC was developed with the aim of modeling the sandbox model in Java2. Umbrella was also analyzed with the ideas which are related to the sandbox model in Java2. In Umbrella an executable file is authenticated before being allowed to enter the system. If the authentication is successful, the file can be executed as a process, and it is given a set of restrictions predefined according to where the executable file comes from, which is identified through the authentication; otherwise a default set of restrictions is applied. The restrictions laid on the active processes are checked by the Umbrella security server running in the system, when the processes try to gain access to system resources.

The formulation of the type system for DSABC began with the investigating on the safety properties in terms of blocking. The calculus of DSABC can prevent a process to exercise the blocked capabilities to interact with its environment, when the process is managed within the blocking scope; however, the blocking cannot avoid the blocked capabilities to be prohibited forever. The blocking scope is always fixed relative to the ambient enclosing it directly, so that once an ambient moves out of the blocking scope which encloses the ambient directly, and enter the scope of the other blocking, some capabilities prohibited by the former blocking may be executed, because the latter blocking maybe prohibit less capabilities than the former. In order to avoid this situation, we introduced *minimal blocking policy* into our type system. Every ambient has its minimal blocking policy, which is a set of blocked capabilities and must be respected during the transitions, that means the ambient must be always managed by the blocking more restricted than its minimal blocking policy during the movements of the ambient.

Another important aspect of DSABC is *who can visit whom.* This is a very important safety property, because we need to ensure who are the trusted visitors and who are allowed to visit for the specified ambient. There exist following ways for an ambient to visit another ambient: (1)If an ambient resides within another ambient, then the former is regarded as the visitor of the later; (2) An ambient uses in capability to enter another ambient; (3)when an ambient exercises out capability, it leaves its parent ambient, and enters the ambient which encloses its parent; (4) an ambient is authenticated by another ambient by exercising auth capability, the authenticated ambient is the visitor of the authenticating ambient; (5) or an ambient can visit another ambient through entering an intermediate ambient which can visit the later. For capturing this safety property, we use the concept of *Ambient Groups*[15]. In our type system, every ambient belongs to one group. We prefer to put the ambients with the same security policy into the same ambient group, so that we can say ambients of which group can visit ambients of the specified group. Based on the concept of ambient groups, we also introduced abstract capabilities such as in A, which means "enter ambients of group A". Our type system takes care of the location of a process. A process or an ambient maybe reside within another ambient, but not every ambient or process resides within a specified ambient, so that we have to introduce the concept of *the universal ambient* @. If we do not specify a location for a process or an ambient, we say that it resides within the universal ambient @. Accordingly there exist *the universal group* Uni, which has only one member ambient @.

The third property of our type system for DSABC is *Key Distribution* which is captured by introducing the notion of *key groups*, which is similar to ambient groups. Every key belongs to one key group, keys of the same key group has the same key distribution policy, which check the following safety properties:

1. The group of ambients that can be authenticated by the ambients who holds an auth capability with the corresponding public key.
2. The group of ambients who are allowed to sign an unsigned ambient with the corresponding private key.

6.3 Typing Rules

Our type system is implicitly parameterized with respect to a set of security constraints. Security constraints ensure which process migrations are allowed between the ambients. We have introduced the notion of groups, with the intention that each ambient(a) belongs to an ambient group(A) and each key(k) belongs to a key group(K). Abstract capabilities are extracted from concrete capabilities,

which are explained in the definition $type(\beta)$.

Process Types

Process types describe the behaviors that process may exhibit. We use set notation for various components on process types. Process type of process P in terms of components can be defined as follows:

$$\mathrm{P} ::= (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$$

Where $\mathcal{B}$ is a set of abstract capabilities that can be exhibited by process P. $\mathcal{A}$ is a set of ambient groups which are found within the scope of current blocking level. $\mathcal{T}$ is a set of abstract auth capabilities that are found within the scope of current blocking level. $\mathcal{U}$ is a set of all ambient groups whose ambients are contained in P. This set also includes the ambient groups which are contained in $\mathcal{A}$. Therefore, $\mathcal{A} \subseteq \mathcal{U}$.

Security Policy and Environments

As [16] described, an ambient is a generalization of barriers, and security has to do with the ability or inability to cross barriers. Therefore, the security policy is defined at the ambient level. Suppose ambient a has the security policy $s \in S$, where S is the set of security policies. s defines the relationship between ambient groups, set of capabilities as follows:

$$s ::= (\quad \mathcal{P}(\mathcal{G}) \times \mathcal{P}(\mathcal{C}) \times \mathcal{P}(\mathcal{C}) \quad) \cup (\quad \mathcal{P}(\mathcal{G}) \times \mathcal{P}(\mathcal{G}) \quad)$$

Where $\mathcal{P}(\mathcal{G})$ is a set of ambient groups, and $\mathcal{P}(\mathcal{C})$ is a set of abstract capabilities.

We define two kinds of environments, namely *Group Environments*, denoted by Γ, and *Security Policy Environments*, denoted by Δ.

1. Γ Environment:

$$\begin{aligned} \Gamma &: \mathcal{N} \rightarrow \mathcal{G} \\ \Gamma &: a \mapsto A \\ \Gamma &: k \mapsto K \end{aligned}$$

   Where $\mathcal{N}$ is an infinite set of names ranged over ambients and keys, and $\mathcal{G}$ is a set of ambient groups or key groups. Γ is a function from names to groups. Here, two kinds of Γ environments can be possible. One is ambient group environments and the other is key group environments. $\Gamma : a \mapsto A$ means that ambient a belongs to ambient group A. $\Gamma : k \mapsto K$ means that key k belongs to key group K. Each ambient a holds a minimal blocking policy which is described in Δ environment.

2. Δ Environment:

$$\begin{aligned}\Delta &: \mathcal{G} \rightarrow S\\ \Delta &: A \mapsto (V_A, \kappa_A, M_A)\\ \Delta &: K \mapsto (A_{\Bbbk}, S_{\Bbbk})\end{aligned}$$

Where Δ is a function from ambient groups or key groups to security policy. Each ambient group or key group must follow the security policy. Here, two kinds of Δ environments can be possible. $\Delta : A \mapsto (V_A, \kappa_A, M_A)$ means that ambient group A follows the security policy (V_A, κ_A, M_A). Where V_A is the set of ambient groups who are allowed to visit any ambient of group A, κ_A is the set of abstract capabilities that can be contained by the ambients of group A, and M_A is the set of blocked capabilities, which is known as a minimal blocking policy that is defined on ambient of group A. The minimal blocking policy should be respected during the movement of the ambients, that means the ambients should be managed by blocking not weaker than the minimal blocking policy. $\Delta : K \mapsto (A_{\Bbbk}, S_{\Bbbk})$ means that key group K follows the security policy $(A_{\Bbbk}, S_{\Bbbk})$, where $A_{\Bbbk}$ is the set of ambient groups who can be authenticated by any ambient that holds a valid public key k of key group K, and $S_{\Bbbk}$ is the set of ambient groups who are allowed to sign an ambient or who holds a valid private key k of key group K.

$type(\beta)$

$type(\beta)$ is a function converting a set of concrete blocked capabilities(β) to a set of corresponding abstract capabilities as described in the following definition:

Definition 6.3.1 ($type(\beta)$) .

$$type(\beta) = \begin{cases} \emptyset & if \beta = \emptyset \\ \bigcup_{C \in \beta} \{type(C)\} & if \beta \neq \emptyset \end{cases}$$

$$\begin{aligned} type(\mathsf{in}\ a) &= \mathsf{in}\ A \ \ if\ \Gamma(a) = A \\ type(\mathsf{out}\ a) &= \mathsf{out}\ A \ \ if\ \Gamma(a) = A \\ type(\mathsf{open}\ a) &= \mathsf{open}\ A \ \ if\ \Gamma(a) = A \\ type(\mathsf{auth}(a,k)) &= \mathsf{auth}(A,K) \ \ if\ \Gamma(a) = A, \Gamma(k) = K \\ type(\mathsf{sign}(a,k)) &= \mathsf{sign}(A,K) \ \ if\ \Gamma(a) = A, \Gamma(k) = K \end{aligned}$$

Judgments

We base our type system on three judgments. Judgments usually have the form $\Delta, \Gamma \vdash \Im$, where $\Im$ is either $\diamond$ or $P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. The pair Γ, Δ are referred to as group environment and security policy environment. The judgments have the following standard meaning:

$\Gamma, \Delta \vdash \diamond$	well-formed environment
$\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$	P has type $(\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ within b
$\Delta, \Gamma \vdash^{\textcircled{S}}_{@} P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$	P is well-typed within @

b can also be @. @ is the universal ambient, which may contain any processes and any ambients. In this report, we follow the convention, that if we specify an arbitrary process without its location, then it is supposed to reside in the universal ambient.

Every ambient has its own minimal blocking policy. When one ambient enters another ambient, the minimal blocking policy of the entering ambient must be respected during transit. i.e, a migrating process is not supposed to enter an area with weaker blocking constraints than its minimal blocking policy. An environment can be considered as well-formed, if it satisfies the following predicate:

$$\begin{aligned}\Delta, \Gamma \vdash \diamond \iff \forall a \in Dom(\Gamma)(\quad & \Gamma(a) = A \quad \wedge \quad \Delta(A) = (V_A, \kappa_A, M_A) \quad \wedge \\ & (\forall b \in Dom(\Gamma)(\Gamma(b) = B \wedge \Delta(B) = (V_B, \kappa_B, M_B) \\ & \wedge \forall(\mathsf{in}\ B \in \kappa_A \vee \mathsf{out}\ B \in \kappa_A))) \implies M_A \subseteq M_B \quad)\end{aligned}$$

Based on the discussion that we have had so far, we can formalize the typing rules as described in table 6.1.

Explanation

(Nil) : Process 0 is well-typed within a well-formed environment.

(Res) & (Repl) : Restriction and replication that are applied on process P will not change its type.

(Par) : The process type of two parallel processes is the union of their respective components from each process.

(In) : If process P which is enclosed in an ambient b is well-typed, then $\mathsf{in}\ a.P$ is also well-typed in case that it is within a well-formed environment, if it also satisfies the following conditions: (1) Since ambient b tries to enter ambient a, the ambients who can visit ambient b must also be the visitors of ambient a. (2) Ambient b must be the visitor of ambient a, as ambient b itself performs subjective move $\mathsf{in}\ a$. If these constraints are not satisfied, then the process violates the security policy of "who visit whom".

(Out) : According to Out rule, an ambient a is trying to move out from an ambient b. Therefore, before exercising $\mathsf{out}\ a$ capability, the ambient b must be

$$\frac{\Delta, \Gamma \vdash \diamond}{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)} \quad \text{(Nil)}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b (\nu n)P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \quad \text{(Res)}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b !\, P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \quad \text{(Repl)}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad \Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b P | P' : (\mathcal{B} \cup \mathcal{B}', \mathcal{A} \cup \mathcal{A}', \mathcal{T} \cup \mathcal{T}', \mathcal{U} \cup \mathcal{U}')} \quad \text{(Par)}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{in}\ a.P : (\mathcal{B} \cup \{\mathsf{in}\ A\}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \quad \text{(In)}$$

$$\begin{array}{l} \Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \\ V_B \subseteq V_A \quad B \in V_A \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{out}\ a.P : (\mathcal{B} \cup \{\mathsf{out}\ A\}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \quad \text{(Out)}$$

$$\begin{array}{l} \Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \\ V_B \subseteq V_A \quad B \in V_A \\ \forall r \in Dom(\Gamma) \quad (\quad \Gamma(r) = R \wedge \Delta(R) = (V_R, \kappa_R, M_R) \\ \qquad\qquad\qquad\qquad \wedge A \in V_R \implies B \in V_R \quad) \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{open}\ a.P : (\mathcal{B} \cup \{\mathsf{open}\ A\} \cup \kappa_A, \mathcal{A}, \mathcal{T}, \mathcal{U})} \quad \text{(Open)}$$

$$\begin{array}{l} \Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \\ V_A \subseteq V_B \quad A \in V_B \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b P\wr_\beta : (\mathcal{B} - type(\beta), \emptyset, \emptyset, \mathcal{U})} \quad \text{(Block)}$$

$$\begin{array}{l} \beta \neq \emptyset \\ \forall A \in \mathcal{A} \quad (\quad \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(\beta) \quad) \\ \forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(\beta)) \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_r b[P] : (\emptyset, \mathcal{A} \cup \{B\}, \emptyset, \mathcal{U} \cup \{B\})} \quad \text{(Amb)}$$

$$\begin{array}{l} \forall A \in \mathcal{A}(\Gamma(a) = A \wedge \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq M_B) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \quad \mathcal{B} \subseteq \kappa_B \quad \mathcal{U} \subseteq V_B \\ \forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset) \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_r b[P]_k : (\emptyset, \mathcal{A} \cup \{B\}, \emptyset, \mathcal{U} \cup \{B\})} \quad \text{(Sign Amb)}$$

$$\begin{array}{l} \forall A \in \mathcal{A}(\Gamma(a) = A \wedge \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq M_B) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \\ \Gamma(k) = K \quad \Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}}) \\ \mathcal{B} \subseteq \kappa_B \quad B \in S_{\mathbb{k}} \quad \mathcal{U} \subseteq V_B \\ \forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset) \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{auth}(a, k).P : (\mathcal{B} \cup \{\mathsf{auth}(A, K)\}, \mathcal{A}, \mathcal{T} \cup \{\mathsf{auth}(A, K)\}, \mathcal{U})} \quad \text{(Auth)}$$

$$\begin{array}{l} \Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \\ \Gamma(k) = K \quad \Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}}) \\ A \in A_{\mathbb{k}} \quad V_A \subseteq V_B \quad A \in V_B \end{array}$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{sign}(a, k).P : (\mathcal{B} \cup \{\mathsf{sign}(A, K)\}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \quad \text{(Sign)}$$

$$\begin{array}{l} \Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \\ \Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B) \\ \Gamma(k) = K \quad \Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}}) \\ B \in S_{\mathbb{k}} \quad V_A \subseteq V_B \quad A \in V_B \end{array}$$

$$\frac{\begin{array}{l}\mathcal{B}_1 \subseteq \mathcal{B}_2 \quad \mathcal{U}_1 \subseteq \mathcal{U}_2 \\ \forall A \in \mathcal{A}_1(\Delta(A) = (V_A, \kappa_A, M_A) \wedge M_A \neq \emptyset \implies A \in \mathcal{A}_2) \\ \forall \mathsf{auth}(D,K) \in \mathcal{T}_1(\quad \Delta(D) = (V_D, \kappa_D, M_D) \wedge M_D \neq \emptyset \\ \qquad\qquad \implies \mathsf{auth}(D,K) \in \mathcal{T}_2 \quad)\end{array}}{(\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \leq (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)} \qquad (\textsc{Sub})$$

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b P : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')} \qquad (\textsc{SubSumption})$$

$$\frac{\begin{array}{l}\Delta, \Gamma \vdash_@ P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \\ \forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A = \emptyset) \\ \forall \mathsf{auth}(D,K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)\end{array}}{\Delta, \Gamma \vdash_@^{\textcircled{S}} P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \qquad (\textsc{Uni})$$

Table 6.1: Typing Rules

inside ambient a, and the environment must also satisfy the following conditions: (1) Since ambient b is inside ambient a, ambients who can visit the ambient b must also be the visitors of ambient a. (2) And ambient b must be the visitor of the ambient a. (3) After the $\mathsf{out}\ a$ capability is exercised, ambient b enters some arbitrary ambient r. Therefore, we have to make sure that ambient b must be the visitor of ambient r.

(Open) : When a process dissolves the boundary of an ambient, the capabilities that are enclosed by that ambient are unleashed and the process acquires those capabilities. As a result those capabilities get exercised within the system. Hence, this eventual action indirectly affects the behavior of the whole system. Therefore, these capabilities must be reflected in the component $\mathcal{B}$. And the environment must also satisfy the following conditions: (1) The ambient that is being opened must be the visitor of an ambient which contains the open capability, otherwise the open capability can never be exercised. (2) The ambients who are the visitors of the ambient which is being opened, must be the visitors of the opening ambient.

(Block) : The crux of our type system is (Block), in which every component of the process type has some special feature. Each component of the process type can be clearly explained with the following process:

$$P ::= a[0] | \mathsf{auth}(m,k).0 | \mathsf{in}\ b.\mathsf{in}\ n$$

The components of process type for above process can be defined as follows:

- $\mathcal{B}$ is the set of abstract capabilities that are exhibited by process P. i.e, $\mathcal{B} = \{\mathsf{in}\ B, \mathsf{in}\ N, \mathsf{auth}(M, K)\}$.
- $\mathcal{A}$ is the set of ambient which are found within the scope of current blocking level or the set of ambients that are waiting to be blocked by some blocking i.e, $\mathcal{A} = \{A\}$.
- $\mathcal{T}$ is the set of abstract auth capabilities that are found within the scope of current blocking level or the set of abstract auth capabilities that are waiting to be blocked by some blocking. i.e, $\mathcal{T} = \{\mathsf{auth}(M, K)\}$.
- $\mathcal{U}$ is the set of all ambient groups whose ambients are contained in P. i.e, $\mathcal{U} = \{A\}$.

When blocking $\beta = \{\mathsf{in}\ b\}$ is applied on the above process as follows:

$$(a[0]|\mathsf{auth}(m, k).0|\mathsf{in}\ b.\mathsf{in}\ n)\wr_{\mathsf{in}\ b}$$

then the type of the resulting process changes as described below:

- β is a set of blocked capabilities that manages process P. Hence, those capabilities are prohibited to get exercised to make interaction between process P and its environment. Therefore, the resulting $\mathcal{B}$ must not include those set of capabilities. In the example, the resulting $\mathcal{B} = \{\mathsf{in}\ N, \mathsf{auth}(M, K)\}$.
- The condition, $M_A \subseteq type(\beta)$ ensures that the minimal blocking policy of ambients in $\mathcal{A}$ must be respected within the scope of β. If this condition is satisfied, then the component $\mathcal{A}$ becomes empty set after the process is enclosed by β.
- Similarly, $M_D \subseteq type(\beta)$ ensures that the minimal blocking policy of ambients that are being authenticated, must also be respected within the scope of β. When this condition is satisfied, then the component $\mathcal{T}$ becomes empty set after the process is enclosed by β.
- Component $\mathcal{U}$ collects all ambients that are contained in process P. Therefore $\mathcal{U}$ remains the same.

(Amb) : In Amb rule, to ensure the environment to be well-formed when process P is enclosed in an ambient b, process P must follow the security policy of b. The environment must also satisfies the following conditions: (1) According to $M_A \subseteq M_B$, the minimal blocking policy of all the ambients that are present within the blocking scope of process P are respected when they are being migrated to ambient b, that means the parent ambient should have more stronger minimal blocking policy than its subambients if they are within the same blocking scope. (2)According to $\mathcal{U} \subseteq V_B$, the set of ambients $\mathcal{U}$ contained in P must be the

visitors of ambient b, since they are subambients of ambient b. (3) According to $\mathcal{B} \subseteq \kappa_B$, the set of capabilities that are exhibited by P can be held by ambient b. (4)According to $M_D = \emptyset$, when a process enclosed in an ambient has auth capabilities which are not managed by any blocking within the ambient, then the ambients which are to be authenticated by these auth capabilities have no requirement with respect to blocking. Consider the following example:

$$P ::= m[T]_k | a[b[Q] | \mathsf{auth}(m,k).R | c[\mathsf{in}\ b.S]]$$

Here we think process P is $b[Q]|\mathsf{auth}(m,k).R|c[\mathsf{in}\ b.S]$, which is enclosed by ambient a, and we will check this with our typing rules. When ambient m is authenticated, it is placed in parallel to process R. Since there is no blocking that manages the auth capability, and if the minimal blocking policy of ambient m is not empty, it may enter an area without blocking. Therefore, to avoid such type of security inconsistency, the minimal blocking policy of m must be empty.

The components of process type can be changed as follows:

- The reason for the $\mathcal{B} = \emptyset$ is that when process P is enclosed in an ambient, the behavior of the ambient becomes nothing with respect to the capabilities that are exhibited by process P. An ambient can be regarded as process 0.
- As a new ambient a is checked, it should be collected in $\mathcal{A}$.
- When process P is enclosed in an ambient, the behavior of the ambient becomes nothing with respect to the auth capabilities that are exhibited by process P, since the authenticated ambients have no requirements with respect to blocking.
- Ambient a must be added to the set $\mathcal{U}$, since it is checked here.

(Sign Amb): In addition to the conditions that are specified in Amb rule, Sing Amb rule must also follow the following one condition: Ambient b can sign its own contents. Therefore, it must belong to the group of ambients who are allowed to sign an ambient with the corresponding key k.

(Auth) : In this typing rule, we try to check the ambients who can be authenticated by an ambient that holds a valid public key k. A process $\mathsf{auth}(a,k).P$ is well-formed if it satisfies the following conditions: (1) Since ambient a gets authenticated by $\mathsf{auth}(a,k)$, it must belong to the set of ambients who are allowed to be authenticated with key k. (2) Since this objective move reflects ambient a to enter ambient b, the ambients who can visit ambient a must also be the visitors of the ambient b. (3) And ambient a must be the visitor of ambient b.

(SIGN) : In this typing rule, we try to check the ambients who are allowed to sign an ambient. A process $\mathsf{sign}(a, k).P$ is well-formed if it satisfies the following conditions: (1) Since ambient b is the one who is going to sign the ambient a, the ambient b must hold a valid private key. If it holds a valid private key, it can belong to the set of ambients who are allowed to sign an ambient with key k.

(SUB) : As mentioned in[14], *Subtyping* captures the intuitive notion of inclusion between types. An element of a type can be considered also as an element of any of its super types. Therefore, if the type of a process is a subtype of another process, it follows the following conditions: (1) All behaviors exhibited by the former are also exhibited by the later as in $\mathcal{B}_1 \subseteq \mathcal{B}_2$. (2) All ambients that are present in the set $\mathcal{U}_1$ are also part of the set $\mathcal{U}_2$. (3) The ambients which have no requirement on blocking can not distinguish the behavioral properties of process in terms of blocking.

(UNI) : A process can be well-typed within the universal ambient(@) through type checking, if it can guarantee the safety properties that are mentioned in Section 6.2; and the process satisfies the following two conditions: (1) $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A = \emptyset$, (2) $\forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset$. These two conditions ensure that there are no safety critical information or subprocesses to be exposed to the universal ambient, then the process is safe(marked by Ⓢ) in an open public environment.

The security issues discussed in Example **??** of Chapter 5 are resolved by the type systems as follows:

- The issue of app_3 using app_1 as a Trojan horse to enter the system still persists. A malicious process is still being executed within the system.
  Solution: For this to be type safe, according to the *auth* rule $V_A \subseteq V_B$, from the reductions it is evident that for this to be possible $V_{app3} \subseteq V_{app1}$ and $V_{app1} \subseteq V_{sys}$ hence we can ensure that if the applet $app3$ entered into the system using a carrier, the system is still safe.

- The second issue was regarding key distribution. If ambient app_1 signs some processes and releases the public key, how does system know that app_1 is the right signatory and not an impostor?
  Solution: Using the (SIGN) rule, the condition $A \in A_{\mathbb{k}}$ ensure that only valid signatories are allowed to sign. In addition to that the sideline condition $A \in V_B$ that ensures that the ambient that will be signed will be in its visitors list V_B

- The third issue was regarding minimal blocking. When an ambient enters

another ambient its minimal blocking policy should be respected. In the example above when the ambient app_1 enters into the Safe System, its minimum blocking policy.
Solution: It is achieved in the (BLOCK),(AMB) and (SIGN AMB) rules. When ever we come across an auth capability it is stored in the $\mathcal{T}$ component, the ambients names are collected in the $\mathcal{A}$ component and the *block* rules is used when ever we come across any blocking. This ensures that minimum blocking is respected.

6.4 Type Safety

In this section, we show our type system is semantically sound. The semantic soundness is formulated as subject reduction. Subject reduction is a property stating that a well typed process always reduce to a well typed process, thus it can guarantee that a well typed process cannot generate "type errors" during its reductions, and it behaves according to the safety properties. In Section 6.2, it is pointed that the three safety properties are captured by our type system.

The soundness of our type system is expressed by the following theorem:

Theorem 1 (Subject Reduction) *If* $\Delta, \Gamma \vdash_r P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $P \rightarrow Q$, *then* $\Delta, \Gamma \vdash_r Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

PROOF: By induction on the derivation of $P \rightarrow Q$. The basic idea is: For one kind of reduction as $P \rightarrow Q$, suppose $\Delta, \Gamma \vdash_r P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, then calculate the process type of every term in process P according to typing rules. Q is reduced from P, hence the process Q is composed of the terms in process P. Based on the process types of those terms which have already been known, we can calculate the process type of Q according to typing rules.

In order to give proof for (RED $\equiv$), Proposition 1 is given.

And in order to overcome the difficulties brought by multi-level blocking, the six propositions 2 3 4 5 6 7 are given.

All the proofs are found in Appendix A and Appendix B.

□

Proposition 1 (Subject Congruence) *(1) If* $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $P \equiv Q$, *then* $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(2) If $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $Q \equiv P$, *then* $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

We use the following function to catch the most interior blocking in an evaluation context:

Definition 6.4.1 (The Inner Most Blocking)

$$InnerMostBlk(\mathscr{C}^C(\!|\ |\!)) =$$

$$\begin{cases} \emptyset & \text{if } C = \emptyset \\ type(C) & \text{if } C \neq \emptyset \text{ and } \mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C \\ type(C) & \text{if } C \neq \emptyset \text{ and } C' = 0 \text{ and} \\ & \mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P)\wr_{C''} (C = C' \cup C'') \\ InnerMostBlk(\mathscr{C}^{C'}(\!|\ |\!)) & \text{if } C \neq \emptyset \text{ and } C' \neq 0 \text{ and} \\ & \mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P)\wr_{C''} (C = C' \cup C'') \end{cases}$$

Proposition 2 is used to calculate the process type when an in capability is consumed within one evaluation context.

Proposition 2 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ \mathsf{in}\ a.P|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *, then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ P|\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ *, where* $\Gamma(a) = A$*,* $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{in}\ A\}$ *when* $\mathsf{in}\ a \notin C$*,* $\mathcal{B} = \mathcal{B}'$ *when* $\mathsf{in}\ a \in C$*.*

Proposition 3 is used to calculate the process type when an out capability is consumed within one evaluation context.

Proposition 3 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ \mathsf{out}\ a.P|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *, then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ P|\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ *where* $\Gamma(a) = A$*,* $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$ *when* $\mathsf{out}\ a \notin C$*,* $\mathcal{B} = \mathcal{B}'$ *when* $\mathsf{out}\ a \in C$*.*

Proposition 4 is used to calculate the process type when an ambient leaves out from an evaluation context.

Proposition 4 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ a[P]_s|Q|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$*, then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ Q|\!) : (\mathcal{B}, \mathcal{A}'', \mathcal{T}, \mathcal{U}'')$ *when* $C = \emptyset$*,* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ Q|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}'')$ *when* $C \neq \emptyset$*, where* $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$*, and* $\mathcal{U} = \mathcal{U}' \cup \mathcal{U}''$*.*

Proposition 5 is used to calculate the process type when an ambient enters into another ambient which resides in an evaluation context.

Proposition 5 *If* $\Delta, \Gamma \vdash_n \mathscr{C}^C(\!|\ b[R]|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$*,* $\Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$*,* $\mathcal{U}' \subseteq V_B$ *and* $M_A \subseteq M_B$*, then* $\Delta, \Gamma \vdash_n \mathscr{C}^C(\!|\ b[R|a[Q]_s]|\!) : (\mathcal{B}, \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{U} \cup \mathcal{U}')$ *when* $C = \emptyset$*,* $\Delta, \Gamma \vdash_n \mathscr{C}^C(\!|\ b[R|a[Q]_s]|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U}')$ *when* $C \neq \emptyset$*, where* $\Gamma(a) = A$*,* $\Delta(A) = (V_A, \kappa_A, M_A)$*,* $\Gamma(b) = B$*,* $\Delta(B) = (V_B, \kappa_B, M_B)$*.*

Proposition 6 is used to calculate the process type when an auth capability is consumed within an evaluation context.

Proposition 6 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ \mathsf{auth}(a,k).P \ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, *then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ P \ |\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}', \mathcal{U})$ *when* $C = \emptyset$, $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ P \ |\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ *when* $C \neq \emptyset$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(k) = K$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{T} = \mathcal{T}' \cup \{\mathsf{auth}(A, K)\}$.

Proposition 7 is used to calculate the process type when an ambient enters into a blocking scope by being authenticated.

Proposition 7 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ Q \ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$ *and* $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!| \ |\!))$, *then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ Q | a[P]_s \ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U}')$ *when* $C \neq \emptyset$, $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ Q | a[P]_s \ |\!) : (\mathcal{B}, \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{U} \cup \mathcal{U}')$ *when* $C = \emptyset$, *where* $\Gamma(a) = A$ *and* $\Delta(A) = (V_A, \kappa_A, M_A)$.

The contribution of our type system is that, it can detect and prohibit the error transition defined in Definition 6.4.2.

Definition 6.4.2 (Error Transition) $\rightarrow_{err}$, *denotes error transition. Suppose there exists predefined environments* Γ *and* Δ, *which satisfy* $\Gamma, \Delta \vdash \diamond$. *The error transitions are defined in terms of* Γ *and* Δ *as follows:*

- (Error In)*:*

$$\mathscr{C}^C(\!| \ b[\mathscr{C}^{C'}(\!| \ \mathsf{in}\ a.P \ |\!)]_s \mid Q \ |\!) \mid \mathscr{C}^{C''}(\!| \ a[R] \ |\!) \ \rightarrow_{err} \mathscr{C}^C(\!| \ Q \ |\!) \mid \mathscr{C}^{C''}(\!| \ a[R \mid b[\mathscr{C}^{C'}(\!| \ P \ |\!)]_s] \ |\!) \ \ (\mathsf{in}\ a \notin C \cup C')$$

  when $\mathsf{in}\ A \notin \kappa_B$, *or* $V_B \not\subseteq V_A$, *or* $B \notin V_A$, *or* $B \notin S_{\Bbbk}$ *if* $s = k$, *or* $M_b \not\subseteq InnerMostBlk(\mathscr{C}^C(\!| \ |\!))$, *or* $M_A \not\subseteq InnerMostBlk(\mathscr{C}^{C''}(\!| \ |\!))$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, *and* $\Delta(K) = (A_{\Bbbk}, S_{\Bbbk})$.

- (Error Out)*:*

$$a[\mathscr{C}^C(\!| \ b[\mathscr{C}^{C'}(\!| \ \mathsf{out}\ a.P \ |\!)]_s \mid Q \ |\!)] \ \rightarrow_{err} a[\mathscr{C}^C(\!| \ Q \ |\!)] \mid b[\mathscr{C}^{C'}(\!| \ P \ |\!)]_s \ (\mathsf{out}\ a \notin C \cup C')$$

  when $\mathsf{out}\ A \notin \kappa_B$, *or* $V_B \not\subseteq V_A$, *or* $B \notin V_A$, *or* $M_B \not\subseteq InnerMostBlk(\mathscr{C}^{C'}(\!| \ |\!))$, *or* $B \notin S_{\Bbbk}$ *if* $s = k$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, *and* $\Delta(K) = (A_{\Bbbk}, S_{\Bbbk})$.

- (Error Open)*:*

$$b[\mathscr{C}^C(\!| \ \mathsf{open}\ a.P \mid a[Q] \ |\!)]_s \mid \ \rightarrow_{err} b[\mathscr{C}^C(\!| \ P \mid Q \ |\!)]_s$$

when $\mathsf{open}\ A \notin \kappa_B$, *or* $\kappa_A \not\subseteq \kappa_B$, *or* $V_A \not\subseteq V_B$, *or* $A \notin V_B$, *or* $M_A \not\subseteq InnerMostBlk(\mathscr{C}^C(\!|\ |\!))$, *or* $B \notin S_{\Bbbk}$ *if* $s = k$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, *and* $\Delta(K) = (A_{\Bbbk}, S_{\Bbbk})$.

- (Error Auth)*:*

$$a[P]_k | b[\mathscr{C}^C(\!|\ \mathsf{auth}(a,k).Q\ |\!)|R]_s \rightarrow_{err} b[\mathscr{C}^C(\!|\ Q|a[P]\ |\!)\ |\ R]_s$$

when $\mathsf{auth}(A,K) \notin \kappa_B$, *or* $M_A \not\subseteq InnerMostBlk(\mathscr{C}^C(\!|\ |\!))$, *or* $A \notin S_{\Bbbk}$, *or* $B \notin A_{\Bbbk}$, *or* $B \notin S'_{\Bbbk}$ *if* $s = k'$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\Bbbk}, S_{\Bbbk})$, $\Gamma(k') = K'$, *and* $\Delta(K') = (A'_{\Bbbk}, S'_{\Bbbk})$.

- (Error Sign)*:*

$$b[\mathsf{sign}(a,k).P|a[Q]_{\epsilon}]_s \rightarrow_{err} b[P|a[Q]_k]_s$$

when $B \notin S_{\Bbbk}$, *or* $V_A \not\subseteq V_B$, *or* $A \notin V_B$, *or* $B \notin S'_{\Bbbk}$ *if* $s = k'$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\Bbbk}, S_{\Bbbk})$, $\Gamma(k') = K'$, *and* $\Delta(K') = (A'_{\Bbbk}, S'_{\Bbbk})$.

- (Error Res)*:*

$$\frac{P \rightarrow_{err} Q}{(\nu n)P \rightarrow_{err} (\nu n)Q}$$

- (Error Amb)*:*

$$\frac{P \rightarrow_{err} Q}{a[P]_s \rightarrow_{err} a[Q]_s}$$

- (Error Par)*:*

$$\frac{P \rightarrow_{err} Q}{P|R \rightarrow_{err} Q|R}$$

- (Error BC)*:*

$$\frac{P \rightarrow_{err} Q}{P\wr_C \rightarrow_{err} Q\wr_C}$$

- (Error $\equiv$)*:*

$$\frac{P \equiv P' \quad P' \rightarrow_{err} Q' \quad Q' \equiv Q}{P \rightarrow_{err} Q}$$

(Error In) (Error Out) (Error Open) (Error Auth) (Error Sign) define the basic transitions which violate the security policy with respect to our type system; (Error Res) (Error Amb) (Error Par) (Error BC) are defined based on the basic error transitions, that means if an internal event of a process is an error transition, then the reduction of the whole process caused by the internal transition is also an error transition; (Error $\equiv$) shows if a process has error transitions, its semantic equivalent processes also have the same error transitions.

A safe process should have no error transition in a predefined well formed environment. Therefore, in order to make judgments if a process is safe, we can define one predicate $safe(\Gamma, \Delta, P)$ as following:

Definition 6.4.3 (Safety) *Given the environments* Γ*,* Δ *and process* P*, we say that* $\text{safe}(\Gamma, \Delta, P)$ *if* $\Gamma, \Delta \vdash \diamond$ *and* $P \nrightarrow_{err}$ *in the environments* Γ *and* Δ*, where* $P \nrightarrow_{err}$ *denotes there exists no error transition from* P*.*

If a process is well typed in the universal ambient, it is supposed to satisfy the safety defined in Definition 6.4.3. This can be expressed by the following theorem:

Theorem 2 (Type Safety) *If* $\Delta, \Gamma \vdash^{\mathbb{S}}_{@} P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$*, then* $\text{safe}(\Gamma, \Delta, P)$*.*

Proof: By inductions on the derivations of $\Delta, \Gamma \vdash_{@} P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. □

Chapter 7

Conclusion

An extension of Mobile Ambients with digital signature and blocking is introduced in this report. We introduced the concept of digital signatures and access control in the form of blocking into Mobile Ambients for the first time. Section 7.1 describes the main objectives that this project accomplishes, and in Section 7.2 we describe a few of the possible continuations of this project.

7.1 Achievements

Based on the application of digital signature and the sandbox model in Java, we have developed the calculus of DISA , which describes a traditional view of the Internet: a piece of code is signed in the sender site, and the code is authenticated by the receiver site, if the authentication is successful, the code can be activated as a process running within the receiver site. We introduce code as a new notation to express this. code is one kind of static structure, processes within code are passive and digitally signed to prevent any tampering. An ambient uses the sign capability to enter a code in order to sign itself, and the auth capability to authenticate an incoming code and activate it in the case of a successful authentication.

Inspired from *mobile agents*, we think over a mechanism for autonomous migrating program to protect itself to be tampered. Ambients can be used to model this kind of programs. Ambients are allowed to sign itself in the calculus of DSA. In DSA, the concept of digitally signed ambients is introduced. A digitally signed ambient can recalculate its own digital signature as it is moving, but prohibits the other ambients without its private key to change its contents and digital signature.

Inspired from Java sandbox model and Umbrella, we introduced the concept of blocking into mobile ambients to model restrictions on processes. We develop three kinds of calculi of mobile ambients with blocking: MABN introduces block-

ing on names into mobile ambients, which is a mechanism to divide a process into different name spaces, and prohibit actions with respect to the blocked names through name hiding; MABC introduces blocking on capabilities into mobile ambients, which models process based mandatory access control; MABA introduces blocking at ambient level into Mobile Ambients, which combines ambient boundary with blocking together, in which blocking is applied along with the boundary of ambients.

For the motivation of modeling Umbrella in which the foreign code gets the corresponding blocking according to where it comes from(identified by its public key), and for the motivation of proposing new solutions of mobile code, we combine DSA and MABC together, and form the calculus of DSABC, which models a predefined blocking assigned to an incoming code according to the key used to calculate its digital signature.

Based on DSABC, we developed a type system which captures the three safety properties to guarantee there is no error transition violating the security policy predefined: who visit whom, the minimal blocking policy and key distribution.

7.2 Future Work

There are various directions for the future work of this project. One of the possible directions is to introduce communication primitives into our calculi with blocking. This would enrich the semantics of blocking when we propose the blocking on communication in mobile ambients.

To capture the property of hierarchical blocking at all levels that is at any given level of an ambient for it to be well typed the child should have more restrictions than its parent.

Another possible direction is to investigate the critical problems of security in the applications of mobile computing and mobile computation. To figure out ways to use our calculi of blocking to solve those problems and improve the security of the applications, at the same time improve our calculi.

Finally, based on the improvements on the calculi, we can develop a type checker and a series of tools.

Appendix A

Subject Congruence

Proposition 1 (Subject Congruence) *(1) If* $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $P \equiv Q$*, then* $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(2) If $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $Q \equiv P$*, then* $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

PROOF: By mutual induction on the derivations of $P \equiv Q$ and $Q \equiv P$.

(1) If $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $P \equiv Q$, then $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(STRUCT REFL) Trival.

(STRUCT SYMM) If $P \equiv Q$, then $Q \equiv P$. According to hypothesis (2), $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(STRUCT TRANS) For $P \equiv Q$, there exists R to make $P \equiv R$, and $R \equiv Q$. We have $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, according to hypothesis (1), we can get $\Delta, \Gamma \vdash_b R : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. According to the same hypothesis, we can get $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(STRUCT RES) $P = (\nu n)P'$, $Q = (\nu n)Q'$, and $P' \equiv Q'$. Using (RES), we have

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b (\nu n)P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.1}$$

from (A.1), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (RES), we have

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b (\nu n)Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.2}$$

(Struct Par) $P = P'|R$, $Q = Q'|R$, $P' \equiv Q'$. Using (Par)

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}') \quad \Delta, \Gamma \vdash_b R : (\mathcal{B}'', \mathcal{A}'', \mathcal{T}'', \mathcal{U}'')}{\Delta, \Gamma \vdash_b P'|R : (\mathcal{B}' \cup \mathcal{B}'', \mathcal{A}' \cup \mathcal{A}'', \mathcal{T}' \cup \mathcal{T}'', \mathcal{U}' \cup \mathcal{U}'')} \tag{A.3}$$

From (A.3),according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')$. Using (Par)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}') \quad \Delta, \Gamma \vdash_b R : (\mathcal{B}'', \mathcal{A}'', \mathcal{T}'', \mathcal{U}'')}{\Delta, \Gamma \vdash_b Q'|R : (\mathcal{B}' \cup \mathcal{B}'', \mathcal{A}' \cup \mathcal{A}'', \mathcal{T}' \cup \mathcal{T}'', \mathcal{U}' \cup \mathcal{U}'')} \tag{A.4}$$

(Struct Repl) $P =! P'$,$Q =! Q'$,$P' \equiv Q'$. Using (Repl)

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b ! P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.5}$$

From (A.5), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (Repl)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b ! Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.6}$$

(Struct Amb) $P = n[P']_s$, $Q = n[Q']_s$, $P' \equiv Q'$. Using (Amb) or (Sign Amb)

$$\frac{\Delta, \Gamma \vdash_n P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b n[P']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{A.7}$$

Where $\Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B} \subseteq \kappa_B$, $\mathcal{U} \subseteq V_B$, $\forall A \in \mathcal{A}(\Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq M_B)$, $\forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$, $\mathcal{A} = \mathcal{A}' \cup \{N\}, \mathcal{U} = \mathcal{U}' \cup \{N\}$

From (A.7), according to hypothesis (1), we have $\Delta, \Gamma \vdash_n Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')$. Using (Amb)

$$\frac{\Delta, \Gamma \vdash_n Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b n[Q']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{A.8}$$

Where $\Gamma(b) = B \quad \Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B} \subseteq \kappa_B$, $\mathcal{U} \subseteq V_B$, $\forall A \in \mathcal{A}(\Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq M_B)$, $\forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$, $\mathcal{A} = \mathcal{A}' \cup \{N\}, \mathcal{U} = \mathcal{U}' \cup \{N\}$

(Struct Action) $P = M.P'$, $Q = M.Q'$, $P' \equiv Q'$.

If $M = \mathsf{in}\ a$: Using (In),

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{in}\ a.P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.9}$$

Where $\Gamma(a) = A, \Delta(A) = (V_A, \kappa_A, M_A), \Gamma(b) = B, \Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{in}\ A\}$

From (A.9), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (In)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{in}\ a.Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.10}$$

Where $\Gamma(a) = A, \Delta(A) = (V_A, \kappa_A, M_A), \Gamma(b) = B, \Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{in}\ A\}$

If $M = \mathsf{out}\ a$: Using (Out),

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{out}\ a.P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.11}$$

Where $\Gamma(a) = A, \Delta(A) = (V_A, \kappa_A, M_A), \Gamma(b) = B, \Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$, $\forall r \in Dom(\Gamma)(\Gamma(r) = R \wedge \Delta(R) = (V_R, \kappa_R, M_R) \wedge A \in V_R \implies B \in V_R)$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$

From (A.11), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (Out)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{out}\ a.Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.12}$$

Where $\Gamma(a) = A, \Delta(A) = (V_A, \kappa_A, M_A), \Gamma(b) = B, \Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$, $\forall r \in Dom(\Gamma)(\Gamma(r) = R \wedge \Delta(R) = (V_R, \kappa_R, M_R) \wedge A \in V_R \implies B \in V_R)$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$

If $M = \mathsf{open}\ a$: Using (Open),

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{open}\ a.P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.13}$$

Where $\Gamma(a) = A, \Delta(A) = (V_A, \kappa_A, M_A), \Gamma(b) = B, \Delta(B) = (V_B, \kappa_B, M_B)$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{open}\ A\} \cup \kappa_A$

From (A.13), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (Open)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{open}\ a.Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.14}$$

Where $\Gamma(a) = A, \Delta(A) = (V_A, \kappa_A, M_A), \Gamma(b) = B, \Delta(B) = (V_B, \kappa_B, M_B)$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{open}\ A\} \cup \kappa_A$

If $M = \mathsf{auth}(a, k)$: Using (AUTH)

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}, \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{auth}(a, k).P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.15}$$

Where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}})$, $A \in A_{\mathbb{k}}$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{T} = \mathcal{T}' \cup \{\mathsf{auth}(A, K)\}$

From (A.15), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (AUTH)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{auth}(a, k).Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.16}$$

Where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}})$, $A \in A_{\mathbb{k}}$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{T} = \mathcal{T}' \cup \{\mathsf{auth}(A, K)\}$

If $M = \mathsf{sign}(a, k)$: Using (SIGN)

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{sign}(a, k).P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.17}$$

Where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}})$, $B \in S_{\mathbb{k}}$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{sign}(A, K)\}$

From (A.17), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (SIGN)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{sign}ak.Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.18}$$

Where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}})$, $B \in S_{\mathbb{k}}$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{sign}(A, K)\}$

(STRUCT BC) $P = P'\wr_C$, $Q = Q'\wr_C$, $P \equiv Q'$, $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, Using (BLOCK)

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_b P'\wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.19}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{B} = \mathcal{B}' - type(C)$,

$\mathcal{A} = \emptyset$, $\mathcal{T} = \emptyset$

From (A.19), according to hypothesis (1), we have $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (Block)

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_b Q' \wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.20}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{B} = \mathcal{B}' - type(C)$, $\mathcal{A} = \emptyset$, $\mathcal{T} = \emptyset$

(Struct Par Comm) $P = P'|Q', Q = Q'|P'$. Using (Par),

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b Q' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b P'|Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.21}$$

Where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$

From (A.21), using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2) \quad \Delta, \Gamma \vdash_b P' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)}{\Delta, \Gamma \vdash_b P'|Q' : (\mathcal{B}_2 \cup \mathcal{B}_1, \mathcal{A}_2 \cup \mathcal{A}_1, \mathcal{T}_2 \cup \mathcal{T}_1, \mathcal{U}_2 \cup \mathcal{U}_1)} \tag{A.22}$$

Where $\mathcal{B}_2 \cup \mathcal{B}_1 = \mathcal{B}$, $\mathcal{A}_2 \cup \mathcal{A}_1 = \mathcal{A}$, $\mathcal{T}_2 \cup \mathcal{T}_1 = \mathcal{T}$, $\mathcal{U}_2 \cup \mathcal{U}_2 = \mathcal{U}$

(Struct Par Assoc) $P = (P'|Q')|R', Q = P'|(Q'|R')$.Using (Par), we have the following:

$$\frac{\Delta, \Gamma \vdash_b P'|Q' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b R' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b (P'|Q')|R' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.23}$$

Where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1) \quad \Delta, \Gamma \vdash_b Q' : (\mathcal{B}''_1, \mathcal{A}''_1, \mathcal{T}''_1, \mathcal{U}''_1)}{\Delta, \Gamma \vdash_b P'|Q' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)} \tag{A.24}$$

Where $\mathcal{B}_1 = \mathcal{B}'_1 \cup \mathcal{B}''_1$, $\mathcal{A}_1 = \mathcal{A}'_1 \cup \mathcal{A}''_1$, $\mathcal{T}_1 = \mathcal{T}'_1 \cup \mathcal{T}''_1$, $\mathcal{U}_1 = \mathcal{U}'_1 \cup \mathcal{U}''_1$

From (A.23) and (A.24), using (Par), we have the following:

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}''_1, \mathcal{A}''_1, \mathcal{T}''_1, \mathcal{U}''_1) \quad \Delta, \Gamma \vdash_b R' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b Q'|R' : (\mathcal{B}_2 \cup \mathcal{B}''_1, \mathcal{A}_2 \cup \mathcal{A}''_1, \mathcal{T}_2 \cup \mathcal{T}''_1, \mathcal{U}_2 \cup \mathcal{U}''_1)} \tag{A.25}$$

$$\frac{\begin{array}{c}\Delta,\Gamma \vdash_b P' : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1) \\ \Delta,\Gamma \vdash_b Q'|R' : (\mathcal{B}_2 \cup \mathcal{B}''_1, \mathcal{A}_2 \cup \mathcal{A}''_1, \mathcal{T}_2 \cup \mathcal{T}''_1, \mathcal{U}_2 \cup \mathcal{U}''_1)\end{array}}{\Delta,\Gamma \vdash_b P'|(Q'|R') : (\mathcal{B}'_1 \cup \mathcal{B}_2 \cup \mathcal{B}''_1, \mathcal{A}'_1 \cup \mathcal{A}_2 \cup \mathcal{A}''_1, \mathcal{T}'_1 \cup \mathcal{T}_2 \cup \mathcal{T}''_1, \mathcal{U}'_1 \cup \mathcal{U}_2 \cup \mathcal{U}''_1)} \tag{A.26}$$

Where $\mathcal{B}'_1 \cup \mathcal{B}_2 \cup \mathcal{B}''_1 = \mathcal{B}$, $\mathcal{A}'_1 \cup \mathcal{A}_2 \cup \mathcal{A}''_1 = \mathcal{A}$, $\mathcal{T}'_1 \cup \mathcal{T}_2 \cup \mathcal{T}''_1 = \mathcal{T}$, $\mathcal{U}'_1 \cup \mathcal{U}_2 \cup \mathcal{U}''_1 = \mathcal{B}$

(Struct Repl Par) $P =! P', Q = P'|! P'$. Using (Repl), we have

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b ! P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.27}$$

From (A.27), using (Par), we have

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad \Delta,\Gamma \vdash_b ! P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b P'|! P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.28}$$

(Struct Res Res) $P = (\nu n)(\nu n')P'$, $Q = (\nu n')(\nu n)P'$. Using (Res), we have the following:

$$\frac{\Delta,\Gamma \vdash_b (\nu n')P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n)(\nu n')P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.29}$$

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n')P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.30}$$

From (A.30), Using (Res), we have:

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n)P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.31}$$

$$\frac{\Delta,\Gamma \vdash_b (\nu n)P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n')(\nu n)P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.32}$$

(Struct Res Par) $P = (\nu n)(P'|Q')$, $Q = P'|(\nu n)Q', n \notin \text{fn}(P')$. Using (Res) and (Par), we have the following:

$$\frac{\Delta,\Gamma \vdash_b P'|Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n)P'|Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.33}$$

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1) \quad \Delta,\Gamma \vdash_b Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_b P'|Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.34}$$

Where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$

From (A.34), using (Res),we have

$$\frac{\Delta,\Gamma \vdash_b Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_b (\nu n)Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)} \tag{A.35}$$

From (A.34), (A.35), Using (Par), we have

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1) \quad \Delta,\Gamma \vdash_b (\nu n)Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_b P'|(\nu n)Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.36}$$

(Struct Res Amb) $P = (\nu n)n'[P']$, $Q = n'[(\nu n)P']$, $n \neq n'$. Using (Res), we have

$$\frac{\Delta,\Gamma \vdash_b n'[P] : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n)n'[P] : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.37}$$

Using (Amb), we have

$$\frac{\Delta,\Gamma \vdash_n P' : (\mathcal{B}',\mathcal{A}',\mathcal{T}',\mathcal{U}')}{\Delta,\Gamma \vdash_b n'[P'] : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.38}$$

Where $\Gamma(b) = B \quad \Delta(B) = (V_B,\kappa_B,M_B)$, $\mathcal{B} \subseteq \kappa_B$, $\mathcal{U} \subseteq V_B$, $\forall A \in \mathcal{A}(\Gamma(a) = A \quad \Delta(A) = (V_A,\kappa_A,M_A) \implies M_A \subseteq M_B)$, $\forall \mathsf{auth}(D,K) \in \mathcal{T} \quad (\Delta(D) = (V_D,\kappa_D,M_D) \implies M_D = \emptyset)$, $\mathcal{A} = \mathcal{A}' \cup \{N'\}$, $\mathcal{U} = \mathcal{U}' \cup \{N'\}$, $\mathcal{B} = \emptyset$, $\mathcal{T} = \emptyset$

From (A.38), using (Res), we have

$$\frac{\Delta,\Gamma \vdash_b P : (\mathcal{B}',\mathcal{A}',\mathcal{T}',\mathcal{U}')}{\Delta,\Gamma \vdash_b (\nu n)P : (\mathcal{B}',\mathcal{A}',\mathcal{T}',\mathcal{U}')} \tag{A.39}$$

From (A.39) and (A.38), using (Amb), we have

$$\frac{\Delta,\Gamma \vdash_n (\nu n)P' : (\mathcal{B}',\mathcal{A}',\mathcal{T}',\mathcal{U}')}{\Delta,\Gamma \vdash_b n'[(\nu n)P'] : (\emptyset,\mathcal{A},\emptyset,\mathcal{U})} \tag{A.40}$$

Where $\Gamma(b) = B \quad \Delta(B) = (V_B,\kappa_B,M_B)$, $\mathcal{B} \subseteq \kappa_B$, $\mathcal{U} \subseteq V_B$, $\forall A \in \mathcal{A}(\Gamma(a) = A \quad \Delta(A) = (V_A,\kappa_A,M_A) \implies M_A \subseteq M_B)$, $\forall \mathsf{auth}(D,K) \in \mathcal{T} \quad (\Delta(D) = (V_D,\kappa_D,M_D) \implies M_D = \emptyset)$, $\mathcal{A} = \mathcal{A}' \cup \{N'\}$, $\mathcal{U} = \mathcal{U}' \cup \{N'\}$

(Struct Res BC) $P = (\nu n)(P'\wr_C)$, $Q = ((\nu n)P')\wr_C$, $n \notin \text{fn}(C)$, Using (Res), we have

$$\frac{\Delta, \Gamma \vdash_b P'\wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b (\nu n)(P)\wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.41}$$

From (A.41), and using (Block), we have:

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_b P'\wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.42}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{B} = \mathcal{B}' - type(C)$, $\mathcal{A} = \emptyset$, $\mathcal{T} = \emptyset$
From (A.42), using (Res), we have:

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_b (\nu n)P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')} \tag{A.43}$$

From (A.43), using (Block), we have:

$$\frac{\Delta, \Gamma \vdash_b (\nu n)P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b ((\nu n)P')\wr_C : (\mathcal{B}'', \mathcal{A}'', \mathcal{T}'', \mathcal{U}'')} \tag{A.44}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{B}'' = \mathcal{B}' - type(C)$, $\mathcal{A}'' = \emptyset$, $\mathcal{T}'' = \emptyset$, $\mathcal{U}'' = \mathcal{U}'$

(Struct Zero Par) $P = P'|0$,$P = P'$. Using (Par) and (Nil), we have

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad \Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b P'|0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.45}$$

Hence, we get $\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(Struct Zero Res) $P = (\nu n)0$, $Q = 0$. Using (Res),we have

$$\frac{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b (\nu n)0 : (\emptyset, \emptyset, \emptyset, \emptyset)} \tag{A.46}$$

From (A.46), we have $\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)$.

(Struct Zero Repl) $P =!\,0$, $Q = 0$.Using (Repl), we have

$$\frac{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b !\,0 : (\emptyset, \emptyset, \emptyset, \emptyset)} \tag{A.47}$$

From (A.47), we have $\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)$.

(Struct Par BC) $P = P' \wr_C |Q' \wr_C$, $Q = (P'|Q') \wr_C$, Using (Par), we have:

$$\frac{\Delta, \Gamma \vdash_b P' \wr_C : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b Q' \wr_C : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b P' \wr_C |Q' \wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.48}$$

Where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$

From (A.48), using (Block), we have:

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1)}{\Delta, \Gamma \vdash_b P' \wr_C : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)} \tag{A.49}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{B}_1 = \mathcal{B}'_1 - type(C)$, $\mathcal{A}_1 = \emptyset$, $\mathcal{T}_1 = \emptyset$, $\mathcal{U}_1 = \mathcal{U}'_1$

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2)}{\Delta, \Gamma \vdash_b Q' \wr_C : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)} \tag{A.50}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{B}_2 = \mathcal{B}'_2 - type(C)$, $\mathcal{A}_2 = \emptyset$, $\mathcal{T}_2 = \emptyset$, $\mathcal{U}_2 = \mathcal{U}'_2$

From (A.49), (A.50), Using (Par), we have:

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1) \quad \Delta, \Gamma \vdash_b Q' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2)}{\Delta, \Gamma \vdash_b P'|Q' : (\mathcal{B}_3, \mathcal{A}_3, \mathcal{T}_3, \mathcal{U}_3)} \tag{A.51}$$

Where $\mathcal{B}_3 = \mathcal{B}'_1 \cup \mathcal{B}'_2$, $\mathcal{A}_3 = \mathcal{A}'_1 \cup \mathcal{A}'_2$, $\mathcal{T}_3 = \mathcal{T}'_1 \cup \mathcal{T}'_2$, $\mathcal{U}_3 = \mathcal{U}'_1 \cup \mathcal{U}'_2$

From (A.51), Using (Block), we have:

$$\frac{\Delta, \Gamma \vdash_b P'|Q' : (\mathcal{B}_3, \mathcal{A}_3, \mathcal{T}_3, \mathcal{U}_3)}{\Delta, \Gamma \vdash_b (P'|Q') \wr_C : (\mathcal{B}_4, \mathcal{A}_4, \mathcal{T}_4, \mathcal{U}_4)} \tag{A.52}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$, $\mathcal{A}_4 = \emptyset = \mathcal{A}$, $\mathcal{T}_4 = \emptyset = \mathcal{T}$, $\mathcal{U}_4 = \mathcal{U}_3 = \mathcal{A}$, , $\mathcal{B}_4 = \mathcal{B}_3 - type(C) \implies \mathcal{B}_4 = (\mathcal{B}'_1 \cup \mathcal{B}'_2) - type(C) \implies \mathcal{B}_4 = (\mathcal{B}'_1 - type(C)) \cup (\mathcal{B}'_2 - type(C)) \implies \mathcal{B}_4 = \mathcal{B}_1 \cup \mathcal{B}_2 \implies \mathcal{B}$

(Struct BC Comb) $P = P' \wr_C \wr_{C'}$, $Q = (P') \wr_{C \cup C'}$, Using (block), we have:

$$\frac{\Delta, \Gamma \vdash_b P' \wr_C : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)}{\Delta, \Gamma \vdash_b P' \wr_C \wr_{C'} : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.53}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C)), \mathcal{A} = \emptyset, \mathcal{T} = \emptyset, \mathcal{U} = \mathcal{U}_1,$, $\mathcal{B} = \mathcal{B}_1 - type(C')$

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b P' \wr_C : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)} \tag{A.54}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C)), \mathcal{A}_1 = \emptyset, \mathcal{T}_1 = \emptyset, \mathcal{U}_1 = \mathcal{U}_2,$, $\mathcal{B}_1 = \mathcal{B}_2 - type(C)$

From (A.54), Using (Block), we have:

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b P' \wr_{C \cup C'} : (\mathcal{B}_3, \mathcal{A}_3, \mathcal{T}_3, \mathcal{U}_3)} \tag{A.55}$$

Where $\forall A \in \mathcal{A}(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C)), \forall \mathsf{auth}(D, K) \in \mathcal{T}(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C)), \mathcal{A}_3 = \emptyset, \mathcal{T}_3 = \emptyset, \mathcal{U}_3 = \mathcal{U}_2,$, $\mathcal{B}_3 = \mathcal{B}_2 - type(C \cup C') \implies \mathcal{B}_3 = \mathcal{B}_2 - type(C) - type(C') \implies \mathcal{B}_3 = \mathcal{B}_1 - type(C') \implies \mathcal{B}_3 = \mathcal{B}$

(Struct BC Empty) Trivial as in $P \equiv P$.

(Struct Zero BC) $P = 0\wr_C$, $Q = 0$. Using (Block), we have:

$$\frac{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b 0\wr_C : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.56}$$

Where $\mathcal{B} = \emptyset - type(C) \implies \mathcal{B} = \emptyset$. Hence, we get $\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)$

(2) If $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $Q \equiv P$, then $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(Struct Refl) Trival.

(Struct Symm) We have $P \equiv Q$.According to hypothesis (1), $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(Struct Trans) For $Q \equiv P$, there exists R to make $Q \equiv R$,$R \equiv P$. Suppose $\Delta, \Gamma \vdash_b P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, according to hypothesis (2), we have $\Delta, \Gamma \vdash_b R : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. And according to the same hypothesis, we get $\Delta, \Gamma \vdash_b Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

(Struct Res) (Struct Par) (Struct Repl) (Struct Amb) (Struct Action) (Struct BC) (Struct Par Comm) (Struct Par Assoc) (Struct Res Res) symmertical to case (1).

(Struct Repl Par) $P = P'|!\,P', Q =!\,P'$. Using (Par), We have

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U}) \quad \Delta,\Gamma \vdash_b !\,P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_b P'|!\,P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.57}$$

(Struct Res Par) $P = P'|(\nu n)Q', Q = (\nu n)(P'|Q'), n \notin \text{fn}(P')$. Using (Par), we have

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1) \quad \Delta,\Gamma \vdash_b (\nu n)Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_b P'|(\nu n)Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.58}$$

Where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$
Using (Res), we have

$$\frac{\Delta,\Gamma \vdash_b Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_b (\nu n)Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)} \tag{A.59}$$

From (A.58), we get $\Delta,\Gamma \vdash_b P' : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1)$. Using (Par), we have

$$\frac{\Delta,\Gamma \vdash_b P' : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1) \quad \Delta,\Gamma \vdash_b Q' : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_b P'|Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.60}$$

Where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$
Using (Res), we get

$$\frac{\Delta,\Gamma \vdash_b P'|Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_b (\nu n)P'|Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.61}$$

(Struct Res Amb) $P = s[(\nu n)P'], Q = (\nu n)s[P'], n \neq m$. Using (Amb), we have

$$\frac{\Delta,\Gamma \vdash_s (\nu n)P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_r s[(\nu n)P'] : (\emptyset,\mathcal{A} \cup \{S\},\emptyset,\mathcal{U} \cup \{S\})} \tag{A.62}$$

Where $\forall A \in \mathcal{A}(\Gamma(a) = A \quad \Delta(A) = (V_A,\kappa_A,M_A) \implies M_A \subseteq M_S), \Gamma(s) = S \quad \Delta(S) = (V_S,\kappa_S,M_S) \quad \mathcal{B} \subseteq \kappa_S \quad \mathcal{U} \subseteq V_S, \forall \mathsf{auth}(D,K) \in \mathcal{T} \quad (\Delta(D) = (V_D,\kappa_D,M_D) \implies M_D = \emptyset)$
Using (Res),we have

$$\frac{\Delta,\Gamma \vdash_s P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_s (\nu n)P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{A.63}$$

From (A.62) and (A.63),using (Amb), we have

$$\frac{\Delta,\Gamma \vdash_s P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_r s[P'] : (\emptyset,\mathcal{A} \cup \{S\},\emptyset,\mathcal{U} \cup \{S\})} \tag{A.64}$$

Where $\forall A \in \mathcal{A}(\Gamma(a) = A \quad \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq M_S), \Gamma(s) = S \quad \Delta(S) = (V_S, \kappa_S, M_S) \quad \mathcal{B} \subseteq \kappa_S \quad \mathcal{U} \subseteq V_S, \forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$
Using (RES), we get

$$\frac{\Delta, \Gamma \vdash_r s[P'] : (\emptyset, \mathcal{A} \cup \{S\}, \emptyset, \mathcal{U} \cup \{S\})}{\Delta, \Gamma \vdash_r (\nu n)s[P'] : (\emptyset, \mathcal{A} \cup \{S\}, \emptyset, \mathcal{U} \cup \{S\})} \tag{A.65}$$

(STRUCT ZERO PAR) $P = P'$, $Q = P'|0$. Using (PAR) and (NIL), we have

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad \Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b P'|0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{A.66}$$

(STRUCT ZERO RES) $P = 0$, $Q = (\nu n)0$. Using (NIL), we get $\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)$. Using (RES), we have

$$\frac{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b (\nu n)0 : (\emptyset, \emptyset, \emptyset, \emptyset)} \tag{A.67}$$

(STRUCT ZERO REPL) $P = 0$, $P = !\,0$. Using (NIL), we get $\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)$. Using (REPL), we have

$$\frac{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b !\,0 : (\emptyset, \emptyset, \emptyset, \emptyset)} \tag{A.68}$$

(STRUCT BC COMB) $P = P' \wr_{C \cup C'}$, $Q = P' \wr_C \wr_{C'}$. Using (BLOCK), we have

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b P' \wr_{C \cup C'} : (\mathcal{B} - type(C \cup C'), \emptyset, \emptyset, \mathcal{U})} \tag{A.69}$$

Where $C \cup C' \neq \emptyset, \forall A \in \mathcal{A} \quad (\quad \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C \cup C') \quad), \forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C \cup C')$
Using (BLOCK), we get

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b P' \wr_C : (\mathcal{B} - type(C), \emptyset, \emptyset, \mathcal{U})} \tag{A.70}$$

Where $C \neq \emptyset, \forall A \in \mathcal{A} \quad (\quad \Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C) \quad), \forall \mathsf{auth}(D, K) \in \mathcal{T} \quad (\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C)$
Using (BLOCK), we get

$$\frac{\Delta, \Gamma \vdash_b P' \wr_C : (\mathcal{B} - type(C), \emptyset, \emptyset, \mathcal{U})}{\Delta, \Gamma \vdash_b P' \wr_C \wr_{C'} : (\mathcal{B} - type(C \cup C'), \emptyset, \emptyset, \mathcal{U})} \tag{A.71}$$

Where $C \cup C' \neq \emptyset$

(Struct BC Empty) Trival.

(Struct Zero BC) $P = 0$, $Q = 0\wr_C$. Using (Nil), we get $\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)$. Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_b 0 : (\emptyset, \emptyset, \emptyset, \emptyset)}{\Delta, \Gamma \vdash_b 0\wr_C : (\emptyset, \emptyset, \emptyset, \emptyset)} \tag{A.72}$$

□

Appendix B

Subject Reduction

Proposition 2 *If* $\Delta,\Gamma \vdash_b \mathscr{C}^C(\!|\ \mathit{in}\ a.P|\!) : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$ *, then* $\Delta,\Gamma \vdash_b \mathscr{C}^C(\!|\ P|\!) : (\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U})$ *, where* $\Gamma(a) = A$*,* $\mathcal{B} = \mathcal{B}' \cup \{\mathit{in}\ A\}$ *when* $\mathit{in}\ a \notin C$*,* $\mathcal{B} = \mathcal{B}'$ *when* $\mathit{in}\ a \in C$*.*

Proof: By induction on the structure of $\mathscr{C}^C(\!|\ |\!)$.

(1) In the case of $\mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C$:

- If $C = \emptyset$:
  We have $\Delta,\Gamma \vdash_b \mathsf{in}\ a.P : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$ and $\mathsf{in}\ a \notin C$, try to prove $\Delta,\Gamma \vdash_b P : (\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{in}\ A\}$. Using (In), we get

$$\frac{\Delta,\Gamma \vdash_b P : (\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_b \mathsf{in}\ a.P : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.1}$$

  where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{in}\ A\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A,\kappa_A,M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B,\kappa_B,M_B)$, $V_B \subseteq V_A$, $B \in V_A$.

- If $C \neq \emptyset$:
  We have $\Delta,\Gamma \vdash_b (\mathsf{in}\ a.P)\wr_C : (\mathcal{B},\emptyset,\emptyset,\mathcal{U})$, try to prove $\Delta,\Gamma \vdash_b (P)\wr_C : (\mathcal{B}',\emptyset,\emptyset,\mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{in}\ A\}$ when $\mathsf{in}\ a \notin C$, $\mathcal{B} = \mathcal{B}'$ when $\mathsf{in}\ a \in C$. Using (Block), we get

$$\frac{\Delta,\Gamma \vdash_b \mathsf{in}\ a.P : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U})}{\Delta,\Gamma \vdash_b (\mathsf{in}\ a.P)\wr_C : (\mathcal{B},\emptyset,\emptyset,\mathcal{U})} \tag{B.2}$$

  where $\mathcal{B} = \mathcal{B}_1 - \mathit{type}(C)$, $\forall R \in \mathcal{A}_1(\Delta(R) = (V_R,\kappa_R,M_R) \implies M_R \subseteq \mathit{type}(C))$, $\forall \mathsf{auth}(D,K) \in \mathcal{T}_1(\Delta(D) = (V_D,\kappa_D,M_D) \implies M_D \subseteq \mathit{type}(C))$. From (B.2),using (In), we get

$$\frac{\Delta,\Gamma \vdash_b P : (\mathcal{B}'_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U})}{\Delta,\Gamma \vdash_b \mathsf{in}\ a.P : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U})} \tag{B.3}$$

where $\mathcal{B}_1 = \mathcal{B'}_1 \cup \{\mathsf{in}\ A\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$.

So that, we have the following:
From (B.3), according to (B.2), using (BLOCK), we get

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B'}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U})}{\Delta, \Gamma \vdash_b (P)\wr_C : (\mathcal{B'}, \emptyset, \emptyset, \mathcal{U})} \tag{B.4}$$

where $\mathcal{B'} = \mathcal{B'}_1 - type(C)$

We have known $\mathcal{B} = \mathcal{B}_1 - type(C)$ and $\mathcal{B}_1 = \mathcal{B'}_1 \cup \{\mathsf{in}\ A\}$. From these, we can get $\mathcal{B} = \mathcal{B'}_1 \cup \{\mathsf{in}\ A\} - type(C)$. If $\mathsf{in}\ a \in C$, then $\mathsf{in}\ A \in type(C)$, then $\mathcal{B} = (\mathcal{B'}_1 - type(C)) \cup (\{\mathsf{in}\ A\} - type(C)) = \mathcal{B'}_1 - type(C) = \mathcal{B'}$; If $\mathsf{in}\ a \notin C$, then $\mathsf{in}\ A \notin type(C)$, then $\mathcal{B} = (\mathcal{B'}_1 - type(C)) \cup \{\mathsf{in}\ A\} = \mathcal{B'} \cup \{\mathsf{in}\ A\}$

(2) In the case of $\mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P_0)\wr_{C''}\ (C = C' \cup C'')$:
Suppose this proposition is hold in $\mathscr{C}^{C'}(\!|\ |\!)$, we must prove this proposition is hold in $\mathscr{C}^C(\!|\ |\!)$.

- If $C'' = \emptyset$:
We have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, we must prove $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!)|P_0 : (\mathcal{B'}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, where $\mathcal{B} = \mathcal{B'} \cup \{\mathsf{in}\ A\}$ when $\mathsf{in}\ a \notin C$, $\mathcal{B} = \mathcal{B'}$ when $\mathsf{in}\ a \in C$.
Using (PAR), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.5}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
According to the supposition, from (B.5), we can have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!) : (\mathcal{B'}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$ where $\mathcal{B}_1 = \mathcal{B'}_1 \cup \{\mathsf{in}\ A\}$ when $\mathsf{in}\ a \notin C'$, or $\mathcal{B}_1 = \mathcal{B'}_1$ when $\mathsf{in}\ a \in C'$.
So that, we have the following:
From (B.5), using (PAR), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!) : (\mathcal{B'}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!)|P_0 : (\mathcal{B'}_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.6}$$

Because $\mathcal{B'}_1 \cup \mathcal{B}_2 \subseteq \mathcal{B}$,using (SUB), we have $(\mathcal{B'}_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

Using (SubSumption), we have

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!)|P_0 : (\mathcal{B'}_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad (\mathcal{B'}_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.7}$$

- If $C'' \neq \emptyset$:
  We have $\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, we must prove $\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$. Using (Block), we get

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.8}$$

  where $\mathcal{B} = \mathcal{B}_t - type(C'')$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C''))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C''))$.
  From (B.8), using (Par), we get

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta,\Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{in}\ a.P|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.9}$$

  where $\mathcal{B}_t = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A}_t = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T}_t = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
  According to the supposition, from (B.9), we get $\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!) : (\mathcal{B'}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$ where $\mathcal{B}_1 = \mathcal{B'}_1 \cup \{\mathsf{in}\ A\}$ when $\mathsf{in}\ a \notin C'$, $\mathcal{B}_1 = \mathcal{B'}_1$ when $\mathsf{in}\ a \in C'$.
  So that, we have the following:
  From (B.9), using (Par), we get

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!) : (\mathcal{B'}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta,\Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!)|P_0 : (\mathcal{B'}_1 \cup \mathcal{B}_2, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.10}$$

  From (B.8), using (Block), we get

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P|\!)|P_0 : (\mathcal{B'}_1 \cup \mathcal{B}_2, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P|\!)|P_0)\wr_{C''} : (\mathcal{B'}_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U})} \tag{B.11}$$

  Because $\mathcal{B'}_1 \subseteq \mathcal{B}_1$, $\mathcal{B'}_1 \cup \mathcal{B}_2 \subseteq \mathcal{B}_t$. So that $\mathcal{B'}_1 \cup \mathcal{B}_2 - type(C'') \subseteq \mathcal{B}$.
  Using (Sub), we have $(\mathcal{B'}_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U}) \leq (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$
  Using (SubSumption), we get

$$\frac{\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P|\!)|P_0)\wr_{C''} : (\mathcal{B'}_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U}) \quad (\mathcal{B'}_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U}) \leq (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})}{\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.12}$$

□

Proposition 3 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ \mathsf{out}\ a.P\ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *, then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ P\ |\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ *where* $\Gamma(a) = A$*,* $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$ *when* $\mathsf{out}\ a \notin C$*,* $\mathcal{B} = \mathcal{B}'$ *when* $\mathsf{out}\ a \in C$*.*

PROOF: By induction on the structure of $\mathscr{C}^C(\!|\ |\!)$.

(1) In the case of $\mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C$:

- If $C = \emptyset$:
  We have $\Delta, \Gamma \vdash_b \mathsf{out}\ a.P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $\mathsf{out}\ a \notin C$, try to prove $\Delta, \Gamma \vdash_b P : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$. Using (OUT), we get

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{out}\ a.P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.13}$$

  where $B = B' \cup \{\mathsf{out}\ A\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$, $\forall r \in Dom(\Gamma)(\Gamma(r) = R \wedge \Delta(R) = (V_R, \kappa_R, M_R) \wedge A \in V_R \implies B \in V_R)$.

- If $C \neq \emptyset$:
  We have $\Delta, \Gamma \vdash_b (\mathsf{out}\ a.P)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, try to prove $\Delta, \Gamma \vdash_b (P)\wr_C : (\mathcal{B}', \emptyset, \emptyset, \mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$ when $\mathsf{out}\ a \notin C$, $\mathcal{B} = \mathcal{B}'$ when $\mathsf{out}\ a \in C$. Using (BLOCK), we get

$$\frac{\Delta, \Gamma \vdash_b \mathsf{out}\ a.P : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathsf{out}\ a.P)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.14}$$

  where $\mathcal{B} = \mathcal{B}_1 - type(C)$, $\forall R \in \mathcal{A}_1(\Delta(R) = (V_R, \kappa_R, M_R) \implies M_R \subseteq type(C))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_1(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$.
  From (B.14),using (OUT), we get

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{out}\ a.P : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U})} \tag{B.15}$$

  where $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{out}\ A\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $V_B \subseteq V_A$, $B \in V_A$, $\forall r \in Dom(\Gamma)(\Gamma(r) = R \wedge \Delta(R) = (V_R, \kappa_R, M_R) \wedge A \in V_R \implies B \in V_R)$.

  So that, we have the following:
  From (B.15), according to (B.14), using (BLOCK), we get

$$\frac{\Delta, \Gamma \vdash_b P : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U})}{\Delta, \Gamma \vdash_b (P)\wr_C : (\mathcal{B}', \emptyset, \emptyset, \mathcal{U})} \tag{B.16}$$

  where $\mathcal{B}' = \mathcal{B}'_1 - type(C)$

We have known $\mathcal{B} = \mathcal{B}_1 - type(C)$ and $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{out}\ A\}$. From these, we can get $\mathcal{B} = \mathcal{B}'_1 \cup \{\mathsf{out}\ A\} - type(C)$. If $\mathsf{out}\ a \in C$, then $\mathsf{out}\ A \in type(C)$, then $\mathcal{B} = (\mathcal{B}'_1 - type(C)) \cup (\{\mathsf{out}\ A\} - type(C)) = \mathcal{B}'_1 - type(C) = \mathcal{B}'$; If $\mathsf{out}\ a \notin C$, then $\mathsf{out}\ A \notin type(C)$, then $\mathcal{B} = (\mathcal{B}'_1 - type(C)) \cup \{\mathsf{out}\ A\} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$

(2) In the case of $\mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P_0)\wr_{C''}\ (C = C' \cup C'')$:
Suppose this proposition is hold in $\mathscr{C}^{C'}(\!|\ |\!)$, we must prove this proposition is hold in $\mathscr{C}^C(\!|\ |\!)$.

- If $C'' = \emptyset$:
We have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, we must prove $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!)|P_0 : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$, where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{out}\ A\}$ when $\mathsf{out}\ a \notin C$, $\mathcal{B} = \mathcal{B}'$ when $\mathsf{out}\ a \in C$.
Using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.17}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
According to the supposition, from (B.17), we can have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$ where $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{out}\ A\}$ when $\mathsf{out}\ a \notin C'$, or $\mathcal{B}_1 = \mathcal{B}'_1$ when $\mathsf{out}\ a \in C'$.
So that, we have the following:
From (B.17), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!)|P_0 : (\mathcal{B}'_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.18}$$

Because $\mathcal{B}'_1 \cup \mathcal{B}_2 \subseteq \mathcal{B}$,using (Sub), we have $(\mathcal{B}'_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.
Using (SubSumption), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!)|P_0 : (\mathcal{B}'_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U}) \quad (\mathcal{B}'_1 \cup \mathcal{B}_2, \mathcal{A}, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.19}$$

- If $C'' \neq \emptyset$:
We have $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P\ |\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, we must prove $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P\ |\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$. Using (Block), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P\ |\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P\ |\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.20}$$

where $\mathcal{B} = \mathcal{B}_t - type(C'')$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C''))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C''))$.
From (B.20), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \ \mathsf{out}\ a.P |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \ \mathsf{out}\ a.P |\!) | P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.21}$$

where $\mathcal{B}_t = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A}_t = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T}_t = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
According to the supposition, from (B.21), we get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \ P |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$ where $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{out}\ A\}$ when $\mathsf{out}\ a \notin C'$, $\mathcal{B}_1 = \mathcal{B}'_1$ when $\mathsf{out}\ a \in C'$.
So that, we have the following:
From (B.21), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \ P |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \ P |\!) | P_0 : (\mathcal{B}'_1 \cup \mathcal{B}_2, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.22}$$

From (B.20), using (Block), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \ P |\!) | P_0 : (\mathcal{B}'_1 \cup \mathcal{B}_2, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!| \ P |\!) | P_0) \wr_{C''} : (\mathcal{B}'_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U})} \tag{B.23}$$

Because $\mathcal{B}'_1 \subseteq \mathcal{B}_1$, $\mathcal{B}'_1 \cup \mathcal{B}_2 \subseteq \mathcal{B}_t$. So that $\mathcal{B}'_1 \cup \mathcal{B}_2 - type(C'') \subseteq \mathcal{B}$.
Using (Sub), we have $(\mathcal{B}'_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U}) \leq (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$
Using (SubSumption), we get

$$\frac{\begin{array}{c}\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!| \ P |\!) | P_0) \wr_{C''} : (\mathcal{B}'_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U}) \\ (\mathcal{B}'_1 \cup \mathcal{B}_2 - type(C''), \emptyset, \emptyset, \mathcal{U}) \leq (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})\end{array}}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!| \ P |\!) | P_0) \wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.24}$$

□

Proposition 4 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^{C}(\!| \ a[P]_s | Q |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ *and* $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, *then* $\Delta, \Gamma \vdash_b \mathscr{C}^{C}(\!| \ Q |\!) : (\mathcal{B}, \mathcal{A}'', \mathcal{T}, \mathcal{U}'')$ *when* $C = \emptyset$, $\Delta, \Gamma \vdash_b \mathscr{C}^{C}(\!| \ Q |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}'')$ *when* $C \neq \emptyset$, *where* $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$, *and* $\mathcal{U} = \mathcal{U}' \cup \mathcal{U}''$.

Proof:

- When $C = \emptyset$:
  It can be supposed that $\mathscr{C}^{\emptyset}(\!| \ a[P]_s | Q |\!) = a[P]_s | Q | P_t$, $\mathscr{C}^{\emptyset}(\!| \ Q |\!) = Q | P_t$.

  We have $\Delta, \Gamma \vdash_b a[P]_s | Q | P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, try to prove $\Delta, \Gamma \vdash_b Q | P_t : (\mathcal{B}, \mathcal{A}'', \mathcal{T}, \mathcal{U}'')$, where $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$, and

$\mathcal{U} = \mathcal{U}' \cup \mathcal{U}''$.

Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}') \quad \Delta, \Gamma \vdash_b Q|P_t : (\mathcal{B}, \mathcal{A}'', \mathcal{T}, \mathcal{U}'')}{\Delta, \Gamma \vdash_b a[P]_s|Q|P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.25}$$

where $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$, and $\mathcal{U} = \mathcal{U}' \cup \mathcal{U}''$.

- When $C \neq \emptyset$:
  By induction on the structure of $\mathscr{C}^C(\!|\ |\!)$.

  (1) In the case of $\mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C$:
  We have $\Delta, \Gamma \vdash_b (a[P]_s|Q)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$ and $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, try to prove $\Delta, \Gamma \vdash_b (Q)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U}'')$ where $\mathcal{U} = \mathcal{U}' \cup \mathcal{U}''$.

  Using (Block), we have

  $$\frac{\Delta, \Gamma \vdash_b a[P]_s|Q : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (a[P]_s|Q)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.26}$$

  where $\mathcal{B} = \mathcal{B}_t - type(C)$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$.

  From (B.26), using (Par), we have

  $$\frac{\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}') \quad \Delta, \Gamma \vdash_b Q : (\mathcal{B}_t, \mathcal{A}_1, \mathcal{T}_t, \mathcal{U}'')}{\Delta, \Gamma \vdash_b a[P]_s|Q : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.27}$$

  where $\mathcal{A}_t = \mathcal{A}_1 \cup \mathcal{A}'$, $\mathcal{U} = \mathcal{U}'' \cup \mathcal{U}'$.

  So that we have the following:
  From (B.27), using (Block), we get

  $$\frac{\Delta, \Gamma \vdash_b Q : (\mathcal{B}_t, \mathcal{A}_1, \mathcal{T}_t, \mathcal{U}'')}{\Delta, \Gamma \vdash_b (Q)\wr_\cdot (\mathcal{B}, \emptyset, \emptyset, \mathcal{U}'')} \tag{B.28}$$

  since $\mathcal{A}_1 \subseteq \mathcal{A}_t$.

  (2) In the case of $\mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P_0)\wr_{C''}$ $(C = C' \cup C'')$:
  Suppose this proposition is held on $\mathscr{C}^{C'}(\!|\ |\!)$, we must prove it is held on $\mathscr{C}^C(\!|\ |\!)$.

– if $C'' = \emptyset$:
We have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, try to prove $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.29}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

We know $C' \neq \emptyset$. From (B.29), according to the supposition, we get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}'_1)$ where $\mathcal{U}_1 = \mathcal{U}'_1 \cup \mathcal{U}'$.

So that we have the following:
Using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}'_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}'_1 \cup \mathcal{U}_2)} \tag{B.30}$$

Because $\mathcal{U}'_1 \cup \mathcal{U}_2 \subseteq \mathcal{U}$, $(\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}'_1 \cup \mathcal{U}_2) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, U)$. Using (SubSumption), we get

$$\frac{\begin{array}{c}\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}'_1 \cup \mathcal{U}_2) \\ (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U}'_1 \cup \mathcal{U}_2) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, U)\end{array}}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.31}$$

– if $C'' \neq \emptyset$:
We have $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$ and $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, try to prove $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ Q|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$.

Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.32}$$

where $\mathcal{B} = \mathcal{B}_t - type(C'')$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C''))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C''))$.

From (B.32), using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ a[P]_s|Q|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.33}$$

where $\mathcal{B}_t = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A}_t = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T}_t = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From (B.33), according to the supposition, we get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\, Q \,|\!) : (\mathcal{B}_1, \mathcal{A'}_1, \mathcal{T}_1, \mathcal{U'}_1)$ where $\mathcal{U}_1 = \mathcal{U'}_1 \cup \mathcal{U'}$, $\mathcal{A'}_1 = \mathcal{A}_1$ when $C' \neq \emptyset$, $\mathcal{A}_1 = \mathcal{A'}_1 \cup \mathcal{A'}$ when $C' = \emptyset$.

So that we have the following:
Using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\, Q \,|\!) : (\mathcal{B}_1, \mathcal{A'}_1, \mathcal{T}_1, \mathcal{U'}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\, Q \,|\!) | P_0 : (\mathcal{B}_t, \mathcal{A'}_1 \cup \mathcal{A}_2, \mathcal{T}_t, \mathcal{U'}_1 \cup \mathcal{U}_2)} \tag{B.34}$$

We know $\mathcal{A'}_1 \cup \mathcal{A}_2 \subseteq \mathcal{A}_t$ and $\mathcal{U'}_1 \cup \mathcal{U}_2 \subseteq \mathcal{U}$. From (B.34) and (B.32), using (Block), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\, Q \,|\!) | P_0 : (\mathcal{B}_t, \mathcal{A'}_1 \cup \mathcal{A}_2, \mathcal{T}_t, \mathcal{U'}_1 \cup \mathcal{U}_2)}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\, Q \,|\!) | P_0) \wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U'}_1 \cup \mathcal{U}_2)} \tag{B.35}$$

Because $\mathcal{U'}_1 \cup \mathcal{U}_2 \subseteq \mathcal{U}$, according to (Sub) we have $(\mathcal{B}, \emptyset, \emptyset, \mathcal{U'}_1 \cup \mathcal{U}_2) \leq (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$. Using (SubSumption), we have

$$\frac{\begin{array}{c}\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\, Q \,|\!) | P_0) \wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U'}_1 \cup \mathcal{U}_2) \\ (\mathcal{B}, \emptyset, \emptyset, \mathcal{U'}_1 \cup \mathcal{U}_2) \leq (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})\end{array}}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\, Q \,|\!) | P_0) \wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.36}$$

□

Proposition 5 *If* $\Delta, \Gamma \vdash_n \mathscr{C}^{C}(\!|\, b[R] \,|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A'}, \emptyset, \mathcal{U'})$, $\mathcal{U'} \subseteq V_B$ *and* $M_A \subseteq M_B$, *then* $\Delta, \Gamma \vdash_n \mathscr{C}^{C}(\!|\, b[R|a[Q]_s] \,|\!) : (\mathcal{B}, \mathcal{A} \cup \mathcal{A'}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})$ *when* $C = \emptyset$, $\Delta, \Gamma \vdash_n \mathscr{C}^{C}(\!|\, b[R|a[Q]_s] \,|\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})$ *when* $C \neq \emptyset$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$.

Proof:

- When $C = \emptyset$:
  It can be supposed that $\mathscr{C}^{\emptyset}(\!|\, b[R] \,|\!) = b[R] | P_t$, $\mathscr{C}^{\emptyset}(\!|\, b[R|a[Q]_s] \,|\!) = b[R|a[Q]_s] | P_t$.
  We have $\Delta, \Gamma \vdash_n b[R] | P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A'}, \emptyset, \mathcal{U'})$,$\mathcal{U'} \subseteq V_B$ and $M_A \subseteq M_B$, try to prove $\Delta, \Gamma \vdash_n b[R|a[Q]_s] | P_t : (\mathcal{B}, \mathcal{A} \cup \mathcal{A'}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})$, $\forall A \in \mathcal{A'}_1 (\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subset M_B)$.
  Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_n b[R] : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_n P_t : (\mathcal{B}, \mathcal{A}_2, \mathcal{T}, \mathcal{U}_2)}{\Delta, \Gamma \vdash_n b[R] | P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.37}$$

  where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From (B.37), using (Amb), we have

$$\frac{\Delta, \Gamma \vdash_b R : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1)}{\Delta, \Gamma \vdash_n b[R] : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)} \tag{B.38}$$

where $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B}'_1 \subseteq \kappa_B$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}'_1(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$, $\mathcal{U}_1 = \mathcal{U}'_1 \cup \{B\}$, $\mathcal{U}'_1 \subseteq V_B$, $\mathcal{A}_1 = \mathcal{A}'_1 \cup \{B\}$.

So that, we have the following:
Using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b R : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1) \quad \Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')}{\Delta, \Gamma \vdash_b R|a[Q]_s : (\mathcal{B}'_1, \mathcal{A}'_1 \cup \mathcal{A}', \mathcal{T}'_1, \mathcal{U}'_1 \cup \mathcal{U}')} \tag{B.39}$$

From (B.39), according to (B.38), we have know $\mathcal{U}' \subseteq V_B$, using (Amb), we get

$$\frac{\Delta, \Gamma \vdash_b R|a[Q]_s : (\mathcal{B}'_1, \mathcal{A}'_1 \cup \mathcal{A}', \mathcal{T}'_1, \mathcal{U}'_1 \cup \mathcal{U}')}{\Delta, \Gamma \vdash_n b[R|a[Q]_s] : (\emptyset, \mathcal{A}_1 \cup \mathcal{A}', \emptyset, \mathcal{U}_1 \cup \mathcal{U}')} \tag{B.40}$$

From (B.40) and (B.37), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_n b[R|a[Q]_s] : (\emptyset, \mathcal{A}_1 \cup \mathcal{A}', \emptyset, \mathcal{U}_1 \cup \mathcal{U}') \quad \Delta, \Gamma \vdash_n P_t : (\mathcal{B}, \mathcal{A}_2, \mathcal{T}, \mathcal{U}_2)}{\Delta, \Gamma \vdash_n b[R|a[Q]_s]|P_t : (\mathcal{B}, \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{U} \cup \mathcal{U}')} \tag{B.41}$$

- When $C \neq \emptyset$:
  By induction on the structure of $\mathscr{C}^C(\!|\ |\!)$.

  (1) In the case of $\mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C$:
  We have $\Delta, \Gamma \vdash_n (b[R])\wr_C : (\emptyset, \emptyset, \emptyset, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, $\mathcal{U}' \subseteq V_B$ and $M_A \subseteq M_B$, try to prove $\Delta, \Gamma \vdash_n (b[R|a[Q]_s])\wr_C : (\emptyset, \emptyset, \emptyset, \mathcal{U} \cup \mathcal{U}')$.

  Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_n b[R] : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U})}{\Delta, \Gamma \vdash_n (b[R])\wr_C : (\emptyset, \emptyset, \emptyset, \mathcal{U})} \tag{B.42}$$

  where $\forall A \in \mathcal{A}_1(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C))$.

  From (B.42), using (Amb), we have

$$\frac{\Delta, \Gamma \vdash_b R : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}_1, \mathcal{U}'_1)}{\Delta, \Gamma \vdash_n b[R] : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U})} \tag{B.43}$$

where $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B}'_1 \subseteq \kappa_B$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_1(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset), \forall A \in \mathcal{A}'_1(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq M_B)$, $\mathcal{A}_1 = \mathcal{A}'_1 \cup \{B\}$, $\mathcal{U} = \mathcal{U}'_1 \cup \{B\}$, $\mathcal{U}'_1 \subseteq V_B$.

So that we have the following:

From (B.43), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b R : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}_1, \mathcal{U}'_1) \quad \Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')}{\Delta, \Gamma \vdash_b R|a[Q]_s : (\mathcal{B}'_1, \mathcal{A}'_1 \cup \mathcal{A}', \mathcal{T}_1, \mathcal{U}'_1 \cup \mathcal{U}')} \tag{B.44}$$

From (B.44) and (B.43), using (Amb), we get

$$\frac{\Delta, \Gamma \vdash_b R|a[Q]_s : (\mathcal{B}'_1, \mathcal{A}'_1 \cup \mathcal{A}', \mathcal{T}_1, \mathcal{U}'_1 \cup \mathcal{U}')}{\Delta, \Gamma \vdash_n b[R|a[Q]_s] : (\emptyset, \mathcal{A}_1 \cup \mathcal{A}', \emptyset, \mathcal{U} \cup \mathcal{U}')} \tag{B.45}$$

Because $B \in \mathcal{A}_1$, $M_B \subseteq type(C)$. And because we have $M_A \subseteq M_B$, $M_A \subseteq type(C)$. And according to (Sign Amb), $\forall G \in \mathcal{A}'(\Delta G = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$, so that $\forall G \in \mathcal{A}'(\Delta G = (V_G, \kappa_G, M_G) \implies M_G \subseteq type(C))$. From (B.45), using (Block), we get

$$\frac{\Delta, \Gamma \vdash_n b[R|a[Q]_s] : (\emptyset, \mathcal{A}_1 \cup \mathcal{A}', \emptyset, \mathcal{U} \cup \mathcal{U}')}{\Delta, \Gamma \vdash_n (b[R|a[Q]_s])\wr_C : (\emptyset, \emptyset, \emptyset, \mathcal{U} \cup \mathcal{U}')} \tag{B.46}$$

(2) In the case of $\mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P_0)\wr_{C''}$ $(C = C' \cup C'')$:
Suppose this proposition is held on $\mathscr{C}^{C'}(\!|\ |\!)$, we must prove it is held on $\mathscr{C}^C(\!|\ |\!)$.

– If $C'' = \emptyset$:
We have $\Delta, \Gamma \vdash_n \mathscr{C}^{C'}(\!|\ b[R]\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[Q]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$, $\mathcal{U}' \subseteq V_B$ and $M_A \subseteq M_B$, try to prove $\Delta, \Gamma \vdash_n \mathscr{C}^b(\!|\ R|a[Q]_s\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U}')$.

Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_n \mathscr{C}^{C'}(\!|\ b[R]\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_n P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_n \mathscr{C}^{C'}(\!|\ b[R]\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.47}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

We know $C' \neq \emptyset$. From (B.47), according to this supposition, we get $\Delta, \Gamma \vdash_n \mathscr{C}^{C'}(\!|\ b[R|a[Q]_s]\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U}')$.

Using (Par), we can get

$$\frac{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!) : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1\cup\mathcal{U}') \quad \Delta,\Gamma\vdash_n P_0 : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!)|P_0 : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U}\cup\mathcal{U}')} \tag{B.48}$$

– If $C''\neq\emptyset$:
We have $\Delta,\Gamma\vdash_n (\mathscr{C}^{C'}(\!|\, b[R]\,|\!)|P_0)\wr_{C''} : (\mathcal{B},\emptyset,\emptyset,\mathcal{U})$, $\Delta,\Gamma\vdash_b a[Q]_s : (\emptyset,\mathcal{A}',\emptyset,\mathcal{U}')$, $\mathcal{U}'\subseteq V_B$ and $M_A\subseteq M_B$,, try to prove $\Delta,\Gamma\vdash_n (\mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!)|P_0)\wr_{C''} : (\mathcal{B},\emptyset,\emptyset,\mathcal{U}\cup\mathcal{U}')$.

Using (Block), we have

$$\frac{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R]\,|\!)|P_0 : (\mathcal{B}_t,\mathcal{A}_t,\mathcal{T}_t,\mathcal{U})}{\Delta,\Gamma\vdash_n (\mathscr{C}^{C'}(\!|\, b[R]\,|\!)|P_0)\wr_{C''} : (\mathcal{B},\emptyset,\emptyset,\mathcal{U})} \tag{B.49}$$

where $\mathcal{B}=\mathcal{B}_t - type(C'')$, $\forall A\in\mathcal{A}_t(\Delta(A)=(V_A,\kappa_A,M_A) \implies M_A\subseteq type(C''))$, $\forall \mathsf{auth}(D,K)\in\mathcal{T}_t(\Delta(D)=(V_D,\kappa_D,M_D) \implies M_D\subseteq type(C''))$.

From (B.49), using (Par), we have

$$\frac{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R]\,|\!) : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1) \quad \Delta,\Gamma\vdash_n P_0 : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R]\,|\!)|P_0 : (\mathcal{B}_t,\mathcal{A}_t,\mathcal{T}_t,\mathcal{U})} \tag{B.50}$$

where $\mathcal{B}_t=\mathcal{B}_1\cup\mathcal{B}_2$, $\mathcal{A}_t=\mathcal{A}_1\cup\mathcal{A}_2$, $\mathcal{T}_t=\mathcal{T}_1\cup\mathcal{T}_2$ and $\mathcal{U}=\mathcal{U}_1\cup\mathcal{U}_2$.

From (B.50), according to this supposition, we can get $\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!) : (\mathcal{B}_1,\mathcal{A}'_1,\mathcal{T}_1,\mathcal{U}_1\cup\mathcal{U}')$, where $\mathcal{A}'_1=\mathcal{A}_1$ when $C'\neq\emptyset$, $\mathcal{A}'_1=\mathcal{A}_1\cup\mathcal{A}'$ when $C'=\emptyset$.

So that we have the following:
Using (Par), we get

$$\frac{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!) : (\mathcal{B}_1,\mathcal{A}'_1,\mathcal{T}_1,\mathcal{U}_1\cup\mathcal{U}') \quad \Delta,\Gamma\vdash_n P_0 : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!)|P_0 : (\mathcal{B}_t,\mathcal{A}'_1\cup\mathcal{A}_2,\mathcal{T}_t,\mathcal{U}\cup\mathcal{U}')} \tag{B.51}$$

where $\mathcal{A}'_1\cup\mathcal{A}_2=\mathcal{A}_t$ when $C'\neq\emptyset$, $\mathcal{A}'_1\cup\mathcal{A}_2=\mathcal{A}_t\cup\mathcal{A}'$ when $C'=\emptyset$.

We have $M_A\subseteq M_B$, and $B\in\mathcal{A}_t$ when $C'=\emptyset$, so that we have $\forall G\in\mathcal{A}'(\Delta(G)=(V_G,\kappa_G,M_G) \implies M_G\subseteq type(C''))$. Using (Block), we can get

$$\frac{\Delta,\Gamma\vdash_n \mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!)|P_0 : (\mathcal{B}_t,\mathcal{A}'_1\cup\mathcal{A}_2,\mathcal{T}_t,\mathcal{U}\cup\mathcal{U}')}{\Delta,\Gamma\vdash_n (\mathscr{C}^{C'}(\!|\, b[R|a[Q]_s]\,|\!)|P_0)\wr_{C''} : (\mathcal{B},\emptyset,\emptyset,\mathcal{U}\cup\mathcal{U}')} \tag{B.52}$$

□

Proposition 6 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ \mathsf{auth}(a,k).P\ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, *then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ P\ |\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}', \mathcal{U})$ *when* $C = \emptyset$, $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ P\ |\!) : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ *when* $C \neq \emptyset$, *where* $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(k) = K$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{T} = \mathcal{T}' \cup \{\mathsf{auth}(A, K)\}$.

PROOF:

- When $C = \emptyset$:
  It can supposed that $\mathscr{C}^{\emptyset}(\!|\ \mathsf{auth}(a,k).P\ |\!) = \mathsf{auth}(a,k).P|P_t$, $\mathscr{C}^{\emptyset}(\!|\ P\ |\!) = P|P_t$.

  We have $\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).P|P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, try to prove $\Delta, \Gamma \vdash_b P|P_t : (\mathcal{B}', \mathcal{A}, \mathcal{T}', \mathcal{U})$, where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{T} = \mathcal{T}' \cup \{\mathsf{auth}(A, K)\}$.

  Using (PAR), we have

$$\frac{\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).P : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_t : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).P|P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.53}$$

  where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

  From (B.53), using (AUTH), we have

$$\frac{\Delta, \Gamma \vdash_b P : ({\mathcal{B}'}_1, \mathcal{A}_1, {\mathcal{T}'}_1, \mathcal{U}_1)}{\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).P : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)} \tag{B.54}$$

  where ${\mathcal{B}}_1 = {\mathcal{B}'}_1 \cup \{\mathsf{auth}(A, K)\}$ and ${\mathcal{T}}_1 = {\mathcal{T}'}_1 \cup \{\mathsf{auth}(A, K)\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}})$, $A \in A_{\mathbb{k}}$, $V_A \subseteq V_B$, $A \in V_B$.

  So that , we have the following:
  Using (PAR), we get

$$\frac{\Delta, \Gamma \vdash_b P : ({\mathcal{B}'}_1, \mathcal{A}_1, {\mathcal{T}'}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_t : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b P|P_t : (\mathcal{B}', \mathcal{A}, \mathcal{T}', \mathcal{U})} \tag{B.55}$$

  where $\mathcal{B}' = {\mathcal{B}'}_1 \cup \mathcal{B}_2$, $\mathcal{T}' = {\mathcal{T}'}_1 \cup \mathcal{T}_2$. From (B.54), we can get $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{T} = \mathcal{T}' \cup \{\mathsf{auth}(A, K)\}$.

- When $C \neq \emptyset$:
  By induction on the structure of $\mathscr{C}^C(\!|\ |\!)$.

(1) In the case of $\mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C$:
We have $\Delta, \Gamma \vdash_b (\mathsf{auth}(a,k).P)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, try to prove $\Delta, \Gamma \vdash_b (P)\wr_C : (\mathcal{B}', \emptyset, \emptyset, \mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(D,K)\}$.

Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).P : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathsf{auth}(a,k).P)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.56}$$

where $\mathcal{B} = \mathcal{B}_t - type(C)$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C))$, $\forall \mathsf{auth}(D,K) \in T_A(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$.

From (B.56), using (Auth), we have

$$\frac{\Delta, \Gamma \vdash_b P : ({\mathcal{B}'}_t, \mathcal{A}_t, {\mathcal{T}'}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).P : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.57}$$

where $\mathcal{B}_t = {\mathcal{B}'}_t \cup \{\mathsf{auth}(A,K)\}$, $\mathcal{T}_t = {\mathcal{T}'}_t \cup \{\mathsf{auth}(A,K)\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\Gamma(k) = K$, $\Delta(K) = (A_{\mathbb{k}}, S_{\mathbb{k}})$, $A \in A_{\mathbb{k}}$, $V_A \subseteq V_B$, $A \in V_B$.

So that , we have the following:
Using (Block), we get

$$\frac{\Delta, \Gamma \vdash_b P : ({\mathcal{B}'}_t, \mathcal{A}_t, {\mathcal{T}'}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (P)\wr_C : (\mathcal{B}', \emptyset, \emptyset, \mathcal{U})} \tag{B.58}$$

where $\mathcal{B}' = {\mathcal{B}'}_t - type(C)$. Since $\mathcal{B}_t = {\mathcal{B}'}_t \cup \{\mathsf{auth}(A,K)\}$, $\mathcal{B} = \mathcal{B}_t - type(C) = \mathcal{B}' \cup \{\mathsf{auth}(A,K)\}$.

(2) In the case of $\mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P_0)\wr_{C''} (C = C' \cup C'')$:
Suppose this proposition is held on $\mathscr{C}^{C'}(\!|\ |\!)$, we must prove it is held on $\mathscr{C}^C(\!|\ |\!)$.

- If $C'' = \emptyset$:
  We have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a,k).P\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, try to prove $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!)|P_0 : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A,K)\}$.

  Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a,k).P\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a,k).P\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.59}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From (B.59), according to this supposition, we can get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$ where $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{auth}(A, K)\}$, since $C' \neq \emptyset$.
So that we have the following:
Using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) | P_0 : (\mathcal{B}', \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.60}$$

where $\mathcal{B}' = \mathcal{B}'_1 \cup \mathcal{B}_2$. Since $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{auth}(A, K)\}$, $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$.

– If $C'' \neq \emptyset$:
We have $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ \mathsf{auth}(a, k).P\ |\!) | P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, try to prove $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P\ |\!) | P_0)\wr_{C''} : (\mathcal{B}', \emptyset, \emptyset, \mathcal{U})$ where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{auth}(A, K)\}$.

Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a, k).P\ |\!) | P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a, k).P\ |\!) | P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.61}$$

where $\mathcal{B} = \mathcal{B}_t - type(C)$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C''))$, $\forall \mathsf{auth}(D, K) \in T_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C''))$.

From (B.61), using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a, k).P\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ \mathsf{auth}(a, k).P\ |\!) | P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.62}$$

where $\mathcal{B}_t = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A}_t = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T}_t = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From (B.63), according to this supposition, we can get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}'_1, \mathcal{U}_1)$ where $\mathcal{B}_1 = \mathcal{B}'_1 \cup \{\mathsf{auth}(D, K)\}$, $\mathcal{T}'_1 = \mathcal{T}_1$ when $C' \neq \emptyset$, $\mathcal{T}_1 = \mathcal{T}'_1 \cup \{\mathsf{auth}(A, K)\}$ when $C' = \emptyset$.

So that we have the following:
Using (Par), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}'_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!) | P_0 : (\mathcal{B}'_t, \mathcal{A}_t, \mathcal{T}'_1 \cup \mathcal{T}_2, \mathcal{U})} \tag{B.63}$$

where $\mathcal{B}_t = \mathcal{B'}_t \cup \{\mathsf{auth}(D, K)\}$.

From (B.64), using (BLOCK), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ P\ |\!)|P_0 : (\mathcal{B'}_t, \mathcal{A}_t, \mathcal{T'}_1 \cup \mathcal{T}_2, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ P\ |\!)|P_0)\wr_{C''} : (\mathcal{B'}, \emptyset, \emptyset, \mathcal{U})} \tag{B.64}$$

where $\mathcal{B'} = \mathcal{B'}_t - type(C'')$. Since $\mathcal{B}_t = \mathcal{B'}_t \cup \{\mathsf{auth}(D, K)\}$ and $\mathcal{B} = \mathcal{B}_t - type(C'')$, $\mathcal{B} = \mathcal{B'} \cup \{\mathsf{auth}(D, K)\}$.

□

Proposition 7 *If* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ Q\ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A'}, \emptyset, \mathcal{U'})$ *and* $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!|\ |\!))$, *then* $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ Q|a[P]_s\ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})$ *when* $C \neq \emptyset$, $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!|\ Q|a[P]_s\ |\!) : (\mathcal{B}, \mathcal{A} \cup \mathcal{A'}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})$ *when* $C = \emptyset$, *where* $\Gamma(a) = A$ *and* $\Delta(A) = (V_A, \kappa_A, M_A)$.

PROOF:

- When $C = \emptyset$:
  It can be supposed that $\mathscr{C}^{\emptyset}(\!|\ Q\ |\!) = Q|P_t$ and $\mathscr{C}^{\emptyset}(\!|\ Q|a[P]_s\ |\!) = Q|a[P]_s|P_t$. And $InnerMostBlk(\mathscr{C}^{\emptyset}(\!|\ |\!)) = \emptyset$, so that $M_A = \emptyset$.

  We have $\Delta, \Gamma \vdash_b Q|P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A'}, \emptyset, \mathcal{U'})$ and $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!|\ |\!))$, try to prove $\Delta, \Gamma \vdash_b Q|a[P]_s|P_t : (\mathcal{B}, \mathcal{A} \cup \mathcal{A'}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})$.

  Using (PAR), we have

$$\frac{\Delta, \Gamma \vdash_b Q : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_t : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b Q|P_t : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.65}$$

  where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

  So that , we have the following:
  Using (PAR), we get

$$\frac{\Delta, \Gamma \vdash_b Q : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A'}, \emptyset, \mathcal{U'})}{\Delta, \Gamma \vdash_b Q|a[P]_s : (\mathcal{B}_1, \mathcal{A}_1 \cup \mathcal{A'}, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U'})} \tag{B.66}$$

  From (B.66) and (B.65), using (PAR), we get

$$\frac{\Delta, \Gamma \vdash_b Q|a[P]_s : (\mathcal{B}_1, \mathcal{A}_1 \cup \mathcal{A'}, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U'}) \quad \Delta, \Gamma \vdash_b P_t : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b Q|a[P]_s|P_t : (\mathcal{B}, \mathcal{A} \cup \mathcal{A'}, \mathcal{T}, \mathcal{U} \cup \mathcal{U'})} \tag{B.67}$$

- When $C \neq \emptyset$:
  By induction on the structure of $\mathscr{C}^C(\!|\ |\!)$.

  (1) In the case of $\mathscr{C}^C(\!|\ |\!) = (\!|\ |\!)\wr_C$
  We have $\Delta, \Gamma \vdash_b (Q)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$. $InnerMostBlk((\!|\ |\!)\wr_C) = type(C)$, so that $M_A \subseteq type(C)$. Try to prove $\Delta, \Gamma \vdash_b (Q|a[P]_s)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U} \cup \mathcal{U}')$.

  Using (Block), we have

  $$\frac{\Delta, \Gamma \vdash_b Q : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (Q)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.68}$$

  where $\mathcal{B} = \mathcal{B}_t - type(C)$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$.

  From (B.68), using (Par), we have

  $$\frac{\Delta, \Gamma \vdash_b Q : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U}) \quad \Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')}{\Delta, \Gamma \vdash_b Q|a[P]_s : (\mathcal{B}_t, \mathcal{A}_t \cup \mathcal{A}', \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')} \tag{B.69}$$

  From (B.69), using (Block), we can get

  $$\frac{\Delta, \Gamma \vdash_b Q|a[P]_s : (\mathcal{B}_t, \mathcal{A}_t \cup \mathcal{A}', \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')}{\Delta, \Gamma \vdash_b (Q|a[P]_s)\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U} \cup \mathcal{U}')} \tag{B.70}$$

  since $M_A \subseteq type(C)$, and those ambients of group sets $\mathcal{A}' - \{A\}$ are within ambient a, so that $\forall G \in (\mathcal{A}' - \{A\})(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$, hence $\forall G \in A'(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq type(C))$.

  (2) In the case of $\mathscr{C}^C(\!|\ |\!) = (\mathscr{C}^{C'}(\!|\ |\!)|P_0)\wr_{C''}$ $(C = C' \cup C'')$:
  Suppose this proposition is held on $\mathscr{C}^{C'}(\!|\ |\!)$, we must prove it is held on $\mathscr{C}^C(\!|\ |\!)$.

  – If $C'' = \emptyset$:
    We have $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$ and $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!|\ |\!))$, try to prove $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U}')$.

    Using (Par), we have

    $$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.71}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
From (B.71), according to the supposition, we can get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U}')$, since $C' \neq \emptyset$.

So that we have the following:
Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U}') \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!)|P_0 : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U} \cup \mathcal{U}')} \tag{B.72}$$

– If $C'' \neq \emptyset$:
We have $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ Q\ |\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, $\Delta, \Gamma \vdash_b a[P]_s : (\emptyset, \mathcal{A}', \emptyset, \mathcal{U}')$ and $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!|\ \ |\!))$, try to prove $\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$.

Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q\ |\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})}{\Delta, \Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ Q\ |\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.73}$$

where $\mathcal{B} = \mathcal{B}_t - type(C'')$, $\forall A \in \mathcal{A}_t(\Delta(A) = (V_A, \kappa_A, M_A) \implies M_A \subseteq type(C''))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}_t(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C''))$.

From (B.73), using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q\ |\!) : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q\ |\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U})} \tag{B.74}$$

where $\mathcal{B}_t = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A}_t = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T}_t = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From (B.74), according to this supposition, we can get $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!) : (\mathcal{B}_1, \mathcal{A}'_1, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U}')$ where $\mathcal{A}'_1 = \mathcal{A}_1 \cup \mathcal{A}'$ when $C' = \emptyset$, $\mathcal{A}'_1 = \mathcal{A}_1$ when $C' \neq \emptyset$.
So that, we have the following:
Using (Par), we can get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!) : (\mathcal{B}_1, \mathcal{A}'_1, \mathcal{T}_1, \mathcal{U}_1 \cup \mathcal{U}') \quad \Delta, \Gamma \vdash_b P_0 : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s\ |\!)|P_0 : (\mathcal{B}_t, \mathcal{A}'_1 \cup \mathcal{A}_2, \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')} \tag{B.75}$$

If $C' = \emptyset$, $InnerMostBlk(\mathscr{C}^C(\!|\ \ |\!)) = type(C'')$. So that $M_A \subseteq type(C'')$. Because all ambients of $\mathcal{A}' - \{A\}$ are in ambient a, $\forall G \in (\mathcal{A}' - \{A\})(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$. Therefore $\forall G \in \mathcal{A}'(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq type(C''))$. If $C' =$

$\emptyset,\ \Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t \cup \mathcal{A}', \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')$. Using (Block), we get

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t \cup \mathcal{A}', \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')}{\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ Q|a[P]_s|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.76}$$

If $C' \neq \emptyset$, $\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')$. Using (Block), we get

$$\frac{\Delta,\Gamma \vdash_b \mathscr{C}^{C'}(\!|\ Q|a[P]_s|\!)|P_0 : (\mathcal{B}_t, \mathcal{A}_t, \mathcal{T}_t, \mathcal{U} \cup \mathcal{U}')}{\Delta,\Gamma \vdash_b (\mathscr{C}^{C'}(\!|\ Q|a[P]_s|\!)|P_0)\wr_{C''} : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.77}$$

□

Lemma 1 *If* $\Delta,\Gamma \vdash_a P : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$, *then* $\Delta,\Gamma \vdash_b P : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$.

Lemma 2 *If* $\Delta,\Gamma \vdash_r a[P]_k : (\emptyset,\mathcal{A},\emptyset,\mathcal{U})$, *then* $\Delta,\Gamma \vdash_r a[P] : (\emptyset,\mathcal{A},\emptyset,\mathcal{U})$.

Lemma 3 *If* $\Delta,\Gamma \vdash_r a[P] : (\emptyset,\mathcal{A},\emptyset,\mathcal{U})$, *then* $\Delta,\Gamma \vdash_r a[P]_k : (\emptyset,\mathcal{A},\emptyset,\mathcal{U})$.

Theorem 1 (Subject Reduction) *If* $\Delta,\Gamma \vdash_r P : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$ *and* $P \to Q$, *then* $\Delta,\Gamma \vdash_r Q : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$.

Proof: By induction on the derivation of $P \to Q$.

(Red In) $P = \mathscr{C}^C(\!|\ a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'|\!)]_s|Q'|\!)|\mathscr{C}^{C''}(\!|\ b[R']|\!)$, $Q = \mathscr{C}^C(\!|\ Q'|\!)|\mathscr{C}^{C''}(\!|\ b[R'|a[\mathscr{C}^{C'}(\!|\ P'|\!)]_s]|\!)$, and $\mathsf{in}\ b \notin C \cup C'$. Suppose $\Delta,\Gamma \vdash_r \mathscr{C}^C(\!|\ a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'|\!)]_s|Q'|\!)|\mathscr{C}^{C''}(\!|\ b[R']|\!) : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$, prove $\Delta,\Gamma \vdash_r \mathscr{C}^C(\!|\ Q'|\!)|\mathscr{C}^{C''}(\!|\ b[R'|a[\mathscr{C}^{C'}(\!|\ P'|\!)]_s]|\!) : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$.

Using (Par), we have

$$\frac{\Delta,\Gamma \vdash_r \mathscr{C}^C(\!|\ a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'|\!)]_s|Q'|\!) : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1) \quad \Delta,\Gamma \vdash_r \mathscr{C}^{C''}(\!|\ b[R']|\!) : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)}{\Delta,\Gamma \vdash_r \mathscr{C}^C(\!|\ a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'|\!)]_s|Q'|\!)|\mathscr{C}^{C''}(\!|\ b[R']|\!) : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.78}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

Suppose

$$\Delta,\Gamma \vdash_r a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'|\!)]_s : (\emptyset,\mathcal{A}_3,\emptyset,\mathcal{U}_3) \tag{B.79}$$

From $\Delta,\Gamma \vdash_r \mathscr{C}^C(\!|\ a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'|\!)]_s|Q'|\!) : (\mathcal{B}_1,\mathcal{A}_1,\mathcal{T}_1,\mathcal{U}_1)$ in (B.78), according to Proposition 4, we can get

$$\Delta,\Gamma \vdash^C_{\mathscr{C}} (\!|\ Q'|\!) : (\mathcal{B}_1,\mathcal{A}'_1,\mathcal{T}_1,\mathcal{U}'_1) \tag{B.80}$$

where $\mathcal{U}_1 = \mathcal{U}'_1 \cup \mathcal{U}_3$, $\mathcal{A}_1 = \mathcal{A}'_1 \cup \mathcal{A}_3$ when $C = \emptyset$, $\mathcal{A}'_1 = \mathcal{A}_1$ when $C \neq \emptyset$.

From (B.79), using (Sign Amb) or (Amb), we have

$$\frac{\Delta,\Gamma \vdash_a \mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'\ |\!) : (\mathcal{B}'_3,\mathcal{A}'_3,\mathcal{T}'_3,\mathcal{U}'_3)}{\Delta,\Gamma \vdash_r a[\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'\ |\!)]_s : (\emptyset,\mathcal{A}_3,\emptyset,\mathcal{U}_3)} \tag{B.81}$$

where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\mathcal{B}'_3 \subseteq \kappa_A$, $\forall \mathsf{auth}(D,K) \in \mathcal{T}'_3(\Delta(D) = (V_D,\kappa_D,M_D) \implies M_D = \emptyset)$, $\forall G \in \mathcal{A}'_3(\Delta(G) = (V_G,\kappa_G,M_G) \implies M_G \subseteq M_A)$, $\mathcal{U}'_3 \subseteq V_A$, $\mathcal{A}_3 = \mathcal{A}'_3 \cup \{A\}$, $\mathcal{U}_3 = \mathcal{U}'_3 \cup \{A\}$.

From $\Delta,\Gamma \vdash_a \mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'\ |\!) : (\mathcal{B}'_3,\mathcal{A}'_3,\mathcal{T}'_3,\mathcal{U}'_3)$ in (B.81), we have known $\mathsf{in}\ b \notin C'$, according to Proposition 2, we can get

$$\Delta,\Gamma \vdash_a \mathscr{C}^{C'}(\!|\ P'\ |\!) : (\mathcal{B}''_3,\mathcal{A}'_3,\mathcal{T}'_3,\mathcal{U}'_3) \tag{B.82}$$

where $\mathcal{B}'_3 = \mathcal{B}''_3 \cup \{\mathsf{in}\ B\}$, $\Gamma(b) = B$, $\Delta(B) = (V_B,\kappa_B,M_B)$.

From (B.82) and (B.81), using (Sign Amb) or (Amb), we can get

$$\frac{\Delta,\Gamma \vdash_a \mathscr{C}^{C'}(\!|\ P'\ |\!) : (\mathcal{B}''_3,\mathcal{A}'_3,\mathcal{T}'_3,\mathcal{U}'_3)}{\Delta,\Gamma \vdash_r a[\mathscr{C}^{C'}(\!|\ P'\ |\!)]_s : (\emptyset,\mathcal{A}_3,\emptyset,\mathcal{U}_3)} \tag{B.83}$$

From $\Delta,\Gamma \vdash_r a[\mathscr{C}^{C'}(\!|\ P'\ |\!)]_s : (\emptyset,\mathcal{A}_3,\emptyset,\mathcal{U}_3)$ in (B.83), according to Lemma 1, we get

$$\Delta,\Gamma \vdash_b a[\mathscr{C}^{C'}(\!|\ P'\ |\!)]_s : (\emptyset,\mathcal{A}_3,\emptyset,\mathcal{U}_3) \tag{B.84}$$

From (B.81), because $\mathsf{in}\ b \notin C'$, we can get $\mathsf{in}\ B \in \mathcal{B}'_3$, and because $\mathcal{B}'_3 \subseteq \kappa_A$, so that $\mathsf{in}\ B \in \kappa_A$. According to the definition of $\Delta,\Gamma \vdash \diamond$, we can get $M_A \subseteq M_B$.

From (B.81), we can see that $\mathscr{C}^{C'}(\!|\ \mathsf{in}\ b.P'\ |\!)$ is well typed, so that $\mathsf{in}\ b.P'$ is also well typed, Suppose $\Delta,\Gamma \vdash_a \mathsf{in}\ b.P : (\mathcal{B}',\mathcal{A}',\mathcal{T}',\mathcal{U}')$. Using (In), we can have

$$\frac{\Delta,\Gamma \vdash_a P' : (\mathcal{B}'',\mathcal{A}',\mathcal{T}',\mathcal{U}')}{\Delta,\Gamma \vdash_a \mathsf{in}\ b.P' : (\mathcal{B}',\mathcal{A}',\mathcal{T}',\mathcal{U}')} \tag{B.85}$$

where $\Gamma(a) = A$, $\Delta(A) = (V_A,\kappa_A,M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B,\kappa_B,M_B)$, $V_A \subseteq V_B$, $A \in V_B$, $\mathcal{B}' = \mathcal{B}'' \cup \{\mathsf{in}\ B\}$.

From (B.85) and (B.81), we can get $\mathcal{U}_3 \subseteq V_B$.

From (B.84) and $\Delta,\Gamma \vdash_r \mathscr{C}^{C''}(\!|\ b[R']\ |\!) : (\mathcal{B}_2,\mathcal{A}_2,\mathcal{T}_2,\mathcal{U}_2)$ in (B.78), we have known $\mathcal{U}_3 \subseteq V_B$ and $M_A \subseteq M_B$, according to Proposition 5, we can get

$$\Delta,\Gamma \vdash_r \mathscr{C}^{C''}(\!|\ b[R'|a[\mathscr{C}^{C'}(\!|\ P'\ |\!)]_s]\ |\!) : (\mathcal{B}_2,\mathcal{A}'_2,\mathcal{T}_2,\mathcal{U}_2 \cup \mathcal{U}_3) \tag{B.86}$$

where $\mathcal{A}'_2 = \mathcal{A}_2 \cup \mathcal{A}_3$ when $C'' = \emptyset$, $\mathcal{A}'_2 = \mathcal{A}_2$ when $C'' \neq \emptyset$.

From (B.86) and (B.80), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_r \mathscr{C}^C(\!| \ Q' \ |\!) : (\mathcal{B}_1, \mathcal{A}'_1, \mathcal{T}_1, \mathcal{U}'_1) \quad \Delta, \Gamma \vdash_r \mathscr{C}^{C''}(\!| \ b[R'|a[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s] \ |\!) : (\mathcal{B}_2, \mathcal{A}'_2, \mathcal{T}_2, \mathcal{U}_2 \cup \mathcal{U}_3)}{\Delta, \Gamma \vdash_r \mathscr{C}^C(\!| \ Q' \ |\!) | \mathscr{C}^{C''}(\!| \ b[R'|a[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s] \ |\!) : (\mathcal{U}, \mathcal{A}'_1 \cup \mathcal{A}'_2, \mathcal{T}, \mathcal{U})} \tag{B.87}$$

Analyze $\mathcal{A}'_1 \cup \mathcal{A}'_2$. If $C = \emptyset$, then $\mathcal{A}_1 = \mathcal{A}'_1 \cup \mathcal{A}_3$; If $C \neq \emptyset$, then $\mathcal{A}'_1 = \mathcal{A}_1$. If $C'' = \emptyset$, then $\mathcal{A}'_2 = \mathcal{A}_2 \cup \mathcal{A}_3$; if $C'' \neq \emptyset$, then $\mathcal{A}_2$. So that there exist 4 cases for calculating $\mathcal{A}'_1 \cup \mathcal{A}'_2$. When $C \neq \emptyset$ and $C'' = \emptyset$, $\mathcal{A}'_1 \cup \mathcal{A}'_2 = \mathcal{A} \cup \mathcal{A}_3$; in the other three cases, $\mathcal{A}'_1 \cup \mathcal{A}'_2 = \mathcal{A}$, which satisfy $\Delta, \Gamma \vdash_r \mathscr{C}^C(\!| \ Q' \ |\!) | \mathscr{C}^{C''}(\!| \ b[R'|a[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s] \ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

Let us prove the case of $C \neq \emptyset$ and $C'' = \emptyset$:
Because $\mathscr{C}^{C''}(\!| \ \ b[R'] \ |\!)$ is within the universal ambient, so that we have $\Delta, \Gamma \vdash_@ \mathscr{C}^{C''}(\!| \ b[R'] \ |\!) : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)$ from (B.78). According to the definition of the well typed process within the universal ambient, we can get $\forall G \in \mathcal{A}_2(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G = \emptyset)$. Because $B \in \mathcal{A}_2$, $M_B = \emptyset$. We have $M_A \subseteq M_B$, $\forall G \in \mathcal{A}'_3(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$, and $\mathcal{A}_3 = \mathcal{A}'_3 \cup \{A\}$, so that $\forall G \in \mathcal{A}_3(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G = \emptyset)$. According (Sub), $(\mathcal{B}, \mathcal{A} \cup \mathcal{A}_3, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$. Using (SubSumption), we have

$$\frac{\begin{array}{c}\Delta, \Gamma \vdash_r \mathscr{C}^C(\!| \ Q' \ |\!) | \mathscr{C}^{C''}(\!| \ b[R'|a[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s] \ |\!) : (\mathcal{B}, \mathcal{A} \cup \mathcal{A}_3, \mathcal{T}, \mathcal{U}) \\ (\mathcal{B}, \mathcal{A} \cup \mathcal{A}_3, \mathcal{T}, \mathcal{U}) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})\end{array}}{\Delta, \Gamma \vdash_r \mathscr{C}^C(\!| \ Q' \ |\!) | \mathscr{C}^{C''}(\!| \ b[R'|a[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s] \ |\!) : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.88}$$

(Red Out) $P = a[\mathscr{C}^C(\!| \ b[\mathscr{C}^{C'}(\!| \ \mathsf{out} \ a.P' \ |\!)]_s | Q \ |\!)]$, $Q = a[\mathscr{C}^C(\!| \ Q' \ |\!)] | b[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s$, $\mathsf{out} \ a \notin C \cup C'$. Suppose $\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!| \ b[\mathscr{C}^{C'}(\!| \ \mathsf{out} \ a.P' \ |\!)]_s | Q \ |\!)] : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!| \ Q' \ |\!)] | b[\mathscr{C}^{C'}(\!| \ P' \ |\!)]_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$.

Using (Amb), we have

$$\frac{\Delta, \Gamma \vdash_a \mathscr{C}^C(\!| \ b[\mathscr{C}^{C'}(\!| \ \mathsf{out} \ a.P' \ |\!)]_s | Q \ |\!) : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!| \ b[\mathscr{C}^{C'}(\!| \ \mathsf{out} \ a.P' \ |\!)]_s | Q \ |\!)] : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{B.89}$$

where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\mathcal{B}' \subseteq \kappa_A$, $\mathcal{A} = \mathcal{A}' \cup \{A\}$, $\mathcal{U} = \mathcal{U}' \cup \{A\}$, $\mathcal{U}' \subseteq V_A$, $\forall G \in \mathcal{A}'(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}'(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$.

Suppose

$$\Delta, \Gamma \vdash_a b[\mathscr{C}^{C'}(\!| \ \mathsf{out} \ a.P' \ |\!)]_s : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1) \tag{B.90}$$

From (B.89) and (B.90), according to Proposition 4, we can get

$$\Delta, \Gamma \vdash_a \mathscr{C}^C(\!| Q |\!) : (\mathcal{B}', \mathcal{A}'', \mathcal{T}', \mathcal{U}'') \tag{B.91}$$

where $\mathcal{U}' = \mathcal{U}_1 \cup \mathcal{U}''$, $\mathcal{A}' = \mathcal{A}_1 \cup \mathcal{A}''$ when $C = \emptyset$, $\mathcal{A}'' = \mathcal{A}'$ when $C \neq \emptyset$.

From (B.91) and (B.89), using (Amb), we get

$$\frac{\Delta, \Gamma \vdash_a \mathscr{C}^C(\!| Q |\!) : (\mathcal{B}', \mathcal{A}'', \mathcal{T}', \mathcal{U}'')}{\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!| Q |\!)] : (\emptyset, \mathcal{A}'' \cup \{A\}, \emptyset, \mathcal{U}'' \cup \{A\})} \tag{B.92}$$

From (B.90), using (Amb) or (Sign Amb), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \mathsf{out}\ a.P' |\!) : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1)}{\Delta, \Gamma \vdash_a b[\mathscr{C}^{C'}(\!| \mathsf{out}\ a.P' |\!)]_s : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)} \tag{B.93}$$

where $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B}'_1 \subseteq \kappa_B$, $\mathcal{A}_1 = \mathcal{A}'_1 \cup \{B\}$, $\mathcal{U}_1 = \mathcal{U}'_1 \cup \{B\}$, $\mathcal{U}' \subseteq V_B$, $\forall G \in \mathcal{A}'_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_B)$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}'_1(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$.

From $\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| \mathsf{out}\ a.P' |\!) : (\mathcal{B}'_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1)$ in (B.93), and $\mathsf{out}\ a \notin C'$, according to Proposition 3, we get

$$\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| P' |\!) : (\mathcal{B}''_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1) \tag{B.94}$$

where $\mathcal{B}'_1 = \mathcal{B}''_1 \cup \{\mathsf{out}\ A\}$.

From (B.94) and (B.93), using (Amb) or (Sign Amb), we get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^{C'}(\!| P' |\!) : (\mathcal{B}''_1, \mathcal{A}'_1, \mathcal{T}'_1, \mathcal{U}'_1)}{\Delta, \Gamma \vdash_a b[\mathscr{C}^{C'}(\!| P' |\!)]_s : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)} \tag{B.95}$$

From (B.95), according to Lemma 1, we get

$$\Delta, \Gamma \vdash_r b[\mathscr{C}^{C'}(\!| P' |\!)]_s : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1) \tag{B.96}$$

From (B.92) and (B.96), using (Par), we get

$$\frac{\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!| Q |\!)] : (\emptyset, \mathcal{A}'' \cup \{A\}, \emptyset, \mathcal{U}'' \cup \{A\}) \quad \Delta, \Gamma \vdash_r b[\mathscr{C}^{C'}(\!| P' |\!)]_s : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)}{\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!| Q |\!)] | b[\mathscr{C}^{C'}(\!| P' |\!)]_s : (\emptyset, \mathcal{A}'' \cup \{A\} \cup \mathcal{A}_1, \emptyset, \mathcal{U}'' \cup \{A\} \cup \mathcal{U}_1)} \tag{B.97}$$

Analyze $\mathcal{U}'' \cup \{A\} \cup \mathcal{U}_1$, we can get $\mathcal{U}'' \cup \{A\} \cup \mathcal{U}_1 = \mathcal{U}$.

Analyze $\mathcal{A}'' \cup \{A\} \cup \mathcal{A}_1$:
When $C = \emptyset$, we can get $\mathcal{A}'' \cup \{A\} \cup \mathcal{A}_1 = \mathcal{A}$. When $C \neq \emptyset$, $\mathcal{A}'' \cup \{A\} \cup \mathcal{A}_1 = \mathcal{A} \cup \mathcal{A}_1$.

Let us analyze $\mathcal{A} \cup \mathcal{A}_1$:
From $\forall G \in \mathcal{A}'_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_B)$ and $\mathcal{A}_1 = \mathcal{A}'_1 \cup \{B\}$, we get $\forall G \in \mathcal{A}_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_B)$.

From (B.93), because $\mathsf{out}\ a \notin C'$, $\mathsf{out}\ A \in \mathcal{B}'_1$. Because $\mathcal{B}'_1 \subseteq \kappa_B$, $\mathsf{out}\ A \in \kappa_B$. According to the definition of $\Delta, \Gamma \vdash \diamond$, we get $M_B \subseteq M_A$.

From $\forall G \in \mathcal{A}_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_B)$ and $M_B \subseteq M_A$, we can get $\forall G \in \mathcal{A}_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$. However, ambient a is within the universal ambient, that means $\Delta, \Gamma \vdash_{@} a[\mathscr{C}^C(\!|\ b[\mathscr{C}^{C'}(\!|\ \mathsf{out}\ a.P'|\!)]_s|Q|\!)] : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$, we can get $A \in \mathcal{A}$ and $\forall G \in \mathcal{A}(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G = \emptyset)$, so that $M_A = \emptyset$. Therefore, $\forall G \in \mathcal{A}_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G = \emptyset)$. According to (Sub), we have $(\emptyset, \mathcal{A} \cup \mathcal{A}_1, \emptyset, \mathcal{U}) \leq (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$. Using (SubSumption), we have

$$\frac{\begin{array}{c}\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!|\ Q'|\!)]|b[\mathscr{C}^{C'}(\!|\ P'|\!)]_s : (\emptyset, \mathcal{A} \cup \mathcal{A}_1, \emptyset, \mathcal{U}) \\ (\emptyset, \mathcal{A} \cup \mathcal{A}_1, \emptyset, \mathcal{U}) \leq (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})\end{array}}{\Delta, \Gamma \vdash_r a[\mathscr{C}^C(\!|\ Q'|\!)]|b[\mathscr{C}^{C'}(\!|\ P'|\!)]_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{B.98}$$

(Red Open) $P = \mathsf{open}\ a.P'|a[Q'], Q = P'|Q'$. Suppose $\Delta, \Gamma \vdash_r \mathsf{open}\ a.P'|a[Q'] : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r P'|Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_r \mathsf{open}\ a.P' : (\mathcal{B}, \mathcal{A}_1, \mathcal{T}, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_r a[Q'] : (\emptyset, \mathcal{A}_2, \emptyset, \mathcal{U}_2)}{\Delta, \Gamma \vdash_r \mathsf{open}\ a.P'|a[Q'] : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.99}$$

where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From $\Delta, \Gamma \vdash_r \mathsf{open}\ a.P' : (\mathcal{B}, \mathcal{A}_1, \mathcal{T}, \mathcal{U}_1)$ in (B.99), using (Open), we have

$$\frac{\Delta, \Gamma \vdash_r P' : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}, \mathcal{U}_1)}{\Delta, \Gamma \vdash_r \mathsf{open}\ a.P' : (\mathcal{B}, \mathcal{A}_1, \mathcal{T}, \mathcal{U}_1)} \tag{B.100}$$

where $\mathcal{B} = \mathcal{B}'_1 \cup \kappa_A \cup \{\mathsf{open}\ A\}$, $\Gamma(r) = R$, $\Delta(R) = (V_R, \kappa_R, M_R)$, $\Gamma(a0 = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $V_A \subseteq V_R$, $A \in V_R$.

From $\Delta, \Gamma \vdash_r a[Q'] : (\emptyset, \mathcal{A}_2, \emptyset, \mathcal{U}_2)$ in (B.99), using (AMB), we have

$$\frac{\Delta, \Gamma \vdash_a Q' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2)}{\Delta, \Gamma \vdash_r a[Q'] : (\emptyset, \mathcal{A}_2, \emptyset, \mathcal{U}_2)} \tag{B.101}$$

where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\mathcal{B}'_2 \subseteq \kappa_A$, $\mathcal{A}_2 = \mathcal{A}'_2 \cup \{A\}$, $\mathcal{U}_2 = \mathcal{U}'_2 \cup \{A\}$, $\mathcal{U}'_2 \subseteq V_A$, $\forall G \in \mathcal{A}'_2(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}'_2(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq \emptyset)$.
From (B.101), according to Lemma 1, we get

$$\Delta, \Gamma \vdash_r Q' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2) \tag{B.102}$$

From (B.102) and (B.100), using (PAR), we have

$$\frac{\Delta, \Gamma \vdash_r P' : (\mathcal{B}'_1, \mathcal{A}_1, \mathcal{T}, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_r Q' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2)}{\Delta, \Gamma \vdash_r P'|Q' : (\mathcal{B}'_1 \cup \mathcal{B}'_2, \mathcal{A}_1 \cup \mathcal{A}'_2, \mathcal{T} \cup \mathcal{T}'_2, \mathcal{U}_1 \cup \mathcal{U}'_2)} \tag{B.103}$$

From $\mathcal{B} = \mathcal{B}'_1 \cup \kappa_A \cup \{\mathsf{open}\ A\}$ and $\mathcal{B}'_2 \subseteq \kappa_A$, we can get $\mathcal{B}'_1 \cup \mathcal{B}'_2 \subseteq \mathcal{B}$.
From $\mathcal{A}_2 = \mathcal{A}'_2 \cup \{A\}$ and $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, we can get $\mathcal{A}_1 \cup \mathcal{A}'_2 \subseteq \mathcal{A}$.
From $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and $\mathcal{U}_2 = \mathcal{U}'_2 \cup \{A\}$, we can get $\mathcal{U}_1 \cup \mathcal{U}'_2 \subseteq \mathcal{U}$.
Analyze $\mathcal{T} \cup \mathcal{T}'_2$: we have already have $\forall \mathsf{auth}(D, K) \in \mathcal{T}'_2(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq \emptyset)$.
Therefore, according to (SUB), we can have $(\mathcal{B}'_1 \cup \mathcal{B}'_2, \mathcal{A}_1 \cup \mathcal{A}'_2, \mathcal{T} \cup \mathcal{T}'_2, \mathcal{U}_1 \cup \mathcal{U}'_2) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.
Using (SUBSUMPTION), we have

$$\frac{\begin{array}{c}\Delta, \Gamma \vdash_r P'|Q' : (\mathcal{B}'_1 \cup \mathcal{B}'_2, \mathcal{A}_1 \cup \mathcal{A}'_2, \mathcal{T} \cup \mathcal{T}'_2, \mathcal{U}_1 \cup \mathcal{U}'_2) \\ (\mathcal{B}'_1 \cup \mathcal{B}'_2, \mathcal{A}_1 \cup \mathcal{A}'_2, \mathcal{T} \cup \mathcal{T}'_2, \mathcal{U}_1 \cup \mathcal{U}'_2) \leq (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})\end{array}}{\Delta, \Gamma \vdash_r P'|Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.104}$$

(RED AUTH) $P = a[P']_k | b[\mathscr{C}^C(\!| \ \mathsf{auth}(a, k).Q' |\!)|R']_s$, $Q = b[\mathscr{C}^C(\!| \ Q'|a[P'] |\!)|R']_s$.
Suppose $\Delta, \Gamma \vdash_r a[P']_k | b[\mathscr{C}^C(\!| \ \mathsf{auth}(a, k).Q' |\!)|R']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \ Q'|a[P'] |\!)|R']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$.

Using (PAR), we have

$$\frac{\Delta, \Gamma \vdash_r a[P']_k : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \ \mathsf{auth}(a, k).Q' |\!)|R']_s : (\emptyset, \mathcal{A}_2, \emptyset, \mathcal{U}_2)}{\Delta, \Gamma \vdash_r a[P']_k | b[\mathscr{C}^C(\!| \ \mathsf{auth}(a, k).Q' |\!)|R']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{B.105}$$

where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From $\Delta, \Gamma \vdash_r a[P']_k : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)$ in (B.105), according to Lemma 2, we can get

$$\Delta, \Gamma \vdash_r a[P'] : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1) \tag{B.106}$$

From (B.106), according to Lemma 1, we can get

$$\Delta, \Gamma \vdash_b a[P'] : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1) \tag{B.107}$$

From $\Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!)|R']_s : (\emptyset, \mathcal{A}_2, \emptyset, \mathcal{U}_2)$ in (B.105), using (Sign Amb) or (Amb), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!)|R' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2)}{\Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!)|R']_s : (\emptyset, \mathcal{A}_2, \emptyset, \mathcal{U}_2)} \tag{B.108}$$

where $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B}'_2 \subseteq \kappa_B$, $\mathcal{A}_2 = \mathcal{A}'_2 \cup \{B\}$, $\mathcal{U}_2 = \mathcal{U}'_2 \cup \{B\}$, $\mathcal{U}'_2 \subseteq V_B$, $\forall G \in \mathcal{A}'_2(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_B)$, $\forall \mathsf{auth}(D,K) \in \mathcal{T}'_2(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$.

From (B.108), using (Par), we have

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!) : (\mathcal{B}_3, \mathcal{A}_3, \mathcal{T}_3, \mathcal{U}_3) \quad \Delta, \Gamma \vdash_b R' : (\mathcal{B}_4, \mathcal{A}_4, \mathcal{T}_4, \mathcal{U}_4)}{\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!)|R' : (\mathcal{B}'_2, \mathcal{A}'_2, \mathcal{T}'_2, \mathcal{U}'_2)} \tag{B.109}$$

where $\mathcal{B}'_2 = \mathcal{B}_3 \cup \mathcal{B}_4$, $\mathcal{A}'_2 = \mathcal{A}_3 \cup \mathcal{A}_4$, $\mathcal{T}'_2 = \mathcal{T}_3 \cup \mathcal{T}_4$, $\mathcal{U}'_2 = \mathcal{U}_3 \cup \mathcal{U}_4$.

From $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!) : (\mathcal{B}_3, \mathcal{A}_3, \mathcal{T}_3, \mathcal{U}_3)$ in (B.109), according to Proposition 6, we can get

$$\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ Q' |\!) : (\mathcal{B}'_3, \mathcal{A}_3, \mathcal{T}'_3, \mathcal{U}_3) \tag{B.110}$$

where $\mathcal{B}_3 = \mathcal{B}'_3 \cup \{\mathsf{auth}(A,K)\}$, $\mathcal{T}_3 = \mathcal{T}'_3 \cup \{\mathsf{auth}(A,K)\}$ when $C = \emptyset$, $\mathcal{T}'_3 = \mathcal{T}_3$ when $C \neq \emptyset$.

From $\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!) : (\mathcal{B}_3, \mathcal{A}_3, \mathcal{T}_3, \mathcal{U}_3)$ in (B.109), we can see that $\mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!)$ is well typed, so that $\mathsf{auth}(a,k).Q'$ is also well typed. Suppose $\Delta, \Gamma \vdash_b \mathsf{auth}(a,k).Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')$, according to (Auth), $\mathsf{auth}(A,K) \in \mathcal{T}'$, where $\Gamma(a) = A$, $\Gamma(k) = K$. Because $\mathscr{C}^C(\!| \ \mathsf{auth}(a,k).Q' |\!)$ is well typed, according to (Block), we have $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!| \ |\!))$ where $\Delta(A) = (V_A, \kappa_A, M_A)$.

From (B.110), (B.107) and $M_A \subseteq InnerMostBlk(\mathscr{C}^C(\!| \ |\!))$, according to Proposition 7, we can get

$$\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \ Q'|a[P'] |\!) : (\mathcal{B}'_3, \mathcal{A}''_3, \mathcal{T}'_3, \mathcal{U}_1 \cup \mathcal{U}_3) \tag{B.111}$$

where $\mathcal{A}''_3 = \mathcal{A}_3$ when $C \neq \emptyset$, $\mathcal{A}''_3 = \mathcal{A}_3 \cup \mathcal{A}_1$ when $C = \emptyset$.

From (B.111) and $\Delta, \Gamma \vdash_b R' : (\mathcal{B}_4, \mathcal{A}_4, \mathcal{T}_4, \mathcal{U}_4)$ in (B.109), using (Par), we can get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \, Q'|a[P']\,|\!) : (\mathcal{B}'_3, \mathcal{A}''_3, \mathcal{T}'_3, \mathcal{U}_1 \cup \mathcal{U}_3) \quad \Delta, \Gamma \vdash_b R' : (\mathcal{B}_4, \mathcal{A}_4, \mathcal{T}_4, \mathcal{U}_4)}{\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \, Q'|a[P']\,|\!)|R' : (\mathcal{B}'_3 \cup \mathcal{B}_4, \mathcal{A}''_3 \cup \mathcal{A}_4, \mathcal{T}'_3 \cup \mathcal{T}_4, \mathcal{U}_1 \cup \mathcal{U}_3 \cup \mathcal{U}_4)} \tag{B.112}$$

Before we use (Sign Amb) or (Amb), let us analyze those conditions which can be satisfied:

Analyze $\mathcal{B}'_3 \cup \mathcal{B}_4$: we can get $\mathcal{B}'_3 \cup \mathcal{B}_4 \subseteq \mathcal{B}'_2$, from $\mathcal{B}'_2 \subseteq \kappa_B$, we get $\mathcal{B}'_3 \cup \mathcal{B}_4 \subseteq \kappa_B$.

Analyze $\mathcal{A}''_3 \cup \mathcal{A}_4$: when $C \neq \emptyset$, $\mathcal{A}''_3 \cup \mathcal{A}_4 = \mathcal{A}'_2$; when $C = \emptyset$, $\mathcal{A}''_3 \cup \mathcal{A}_4 = \mathcal{A}'_2 \cup \mathcal{A}_1$. In $\Delta, \Gamma \vdash_r a[P']_k : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)$, ambient a is within the universal ambient, that means $\Delta, \Gamma \vdash_@ a[P']_k : (\emptyset, \mathcal{A}_1, \emptyset, \mathcal{U}_1)$. So that we have $\forall G \in \mathcal{A}_1(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G = \emptyset)$. Therefore, we have $\forall G \in (\mathcal{A}''_3 \cup \mathcal{A}_4)(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_B)$.

Analyze $\mathcal{T}'_3 \cup \mathcal{T}_4$: we can get $\mathcal{T}'_3 \cup \mathcal{T}_4 \subseteq \mathcal{T}'_2$.

Analyze $\mathcal{U}_1 \cup \mathcal{U}_3 \cup \mathcal{U}_4$: we can get $\mathcal{U}_1 \cup \mathcal{U}_3 \cup \mathcal{U}_4 = \mathcal{U}'_2 \cup \mathcal{U}_1$. From (B.107), we can get $\mathcal{U}_1 \subseteq V_A$, where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$. Because $\mathsf{auth}(a, k).Q'$ is well typed within ambient b, according to (Auth), we can get $V_A \subseteq V_B$. So that we can get $\mathcal{U}_1 \subseteq V_B$.

So that, from (B.112), using (Amb), we can get

$$\frac{\Delta, \Gamma \vdash_b \mathscr{C}^C(\!| \, Q'|a[P']\,|\!)|R' : (\mathcal{B}'_3 \cup \mathcal{B}_4, \mathcal{A}''_3 \cup \mathcal{A}_4, \mathcal{T}'_3 \cup \mathcal{T}_4, \mathcal{U}_1 \cup \mathcal{U}_3 \cup \mathcal{U}_4)}{\Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \, Q'|a[P']\,|\!)|R']_s : (\emptyset, \mathcal{A}''_3 \cup \mathcal{A}_4 \cup \{B\}, \emptyset, \mathcal{U})} \tag{B.113}$$

where $\mathcal{A}''_3 \cup \mathcal{A}_4 \cup \{B\} = \mathcal{A}_2$ when $C \neq \emptyset$, $\mathcal{A}_2 \subseteq \mathcal{A}$; $\mathcal{A}''_3 \cup \mathcal{A}_4 \cup \{B\} = \mathcal{A}_2 \cup \mathcal{A}_1 = \mathcal{A}$ when $C = \emptyset$. So that, according to (Sub), we have $(\emptyset, \mathcal{A}''_3 \cup \mathcal{A}_4 \cup \{B\}, \emptyset, \mathcal{U}) \leq (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$. Using (SubSumption), we have

$$\frac{\Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \, Q'|a[P']\,|\!)|R']_s : (\emptyset, \mathcal{A}''_3 \cup \mathcal{A}_4 \cup \{B\}, \emptyset, \mathcal{U}) \quad (\emptyset, \mathcal{A}''_3 \cup \mathcal{A}_4 \cup \{B\}, \emptyset, \mathcal{U}) \leq (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})}{\Delta, \Gamma \vdash_r b[\mathscr{C}^C(\!| \, Q'|a[P']\,|\!)|R']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{B.114}$$

(Red Sign) $P = \mathsf{sign}(a, k).P'|a[Q']$, $Q = P'|a[Q']_k$. Suppose $\Delta, \Gamma \vdash_r \mathsf{sign}(a, k).P'|a[Q'] : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r P'|a[Q']_k : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

Using (Par), we have

$$\frac{\Delta,\Gamma \vdash_r \mathsf{sign}(a,k).P' : (\mathcal{B},\mathcal{A}_1,\mathcal{T},\mathcal{U}_1) \quad \Delta,\Gamma \vdash_r a[Q'] : (\emptyset,\mathcal{A}_2,\emptyset,\mathcal{U}_2)}{\Delta,\Gamma \vdash_r \mathsf{sign}(a,k).P'|a[Q'] : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.115}$$

where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From $\Delta,\Gamma \vdash_r \mathsf{sign}(a,k).P' : (\mathcal{B},\mathcal{A}_1,\mathcal{T},\mathcal{U}_1)$ in (B.115), using (Sign), we have

$$\frac{\Delta,\Gamma \vdash_r P' : (\mathcal{B}',\mathcal{A}_1,\mathcal{T},\mathcal{U}_1)}{\Delta,\Gamma \vdash_r \mathsf{sign}(a,k).P' : (\mathcal{B},\mathcal{A}_1,\mathcal{T},\mathcal{U}_1)} \tag{B.116}$$

where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{sign}(A,K)\}$, $\Gamma(a) = A$, $\Delta(A) = (V_A,\kappa_A,M_A)$, $\Gamma(r) = R$, $\Delta(R) = (V_R,\kappa_R,M_R)$, $V_A \subseteq V_R$, $A \in V_R$.

From $\Delta,\Gamma \vdash_r a[Q'] : (\emptyset,\mathcal{A}_2,\emptyset,\mathcal{U}_2)$, according to Lemma 3, we get

$$\Delta,\Gamma \vdash_r a[Q']_k : (\emptyset,\mathcal{A}_2,\emptyset,\mathcal{U}_2) \tag{B.117}$$

From (B.116) and (B.117), using (Par), we get

$$\frac{\Delta,\Gamma \vdash_r P' : (\mathcal{B}',\mathcal{A}_1,\mathcal{T},\mathcal{U}_1) \quad \Delta,\Gamma \vdash_r a[Q']_k : (\emptyset,\mathcal{A}_2,\emptyset,\mathcal{U}_2)}{\Delta,\Gamma \vdash_r P'|a[Q']_k : (\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.118}$$

where $\mathcal{B} = \mathcal{B}' \cup \{\mathsf{sign}(A,K)\}$. So that we have $(\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U}) \leq (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$ according to (Sub). Using (SubSumption), we have

$$\frac{\begin{array}{c}\Delta,\Gamma \vdash_r P'|a[Q']_k : (\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U}) \\ (\mathcal{B}',\mathcal{A},\mathcal{T},\mathcal{U}) \leq (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})\end{array}}{\Delta,\Gamma \vdash_r P'|a[Q']_k : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.119}$$

(Red Res) $P = (\nu n)P'$, $Q = (\nu n)Q'$, and $P' \to Q'$. Suppose $\Delta,\Gamma \vdash_r (\nu n)P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$, prove $\Delta,\Gamma \vdash_r (\nu n)Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$.

Using (Res), we have

$$\frac{\Delta,\Gamma \vdash_r P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_r (\nu n)P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.120}$$

From $\Delta,\Gamma \vdash_r P' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$ in (B.120), and $P' \to Q'$, according to the hypothesis, we have $\Delta,\Gamma \vdash_r Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})$. Using (Res), we get

$$\frac{\Delta,\Gamma \vdash_r Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})}{\Delta,\Gamma \vdash_r (\nu n)Q' : (\mathcal{B},\mathcal{A},\mathcal{T},\mathcal{U})} \tag{B.121}$$

(Red Amb) $P = a[P']_s$, $Q = a[Q']_s$, and $P' \to Q'$. Suppose $\Delta, \Gamma \vdash_b a[P']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$, prove $\Delta, \Gamma \vdash_b a[P']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})$.

Using (Amb) or (Sign Amb), we have

$$\frac{\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b a[P']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{B.122}$$

where $\Gamma(a) = A$, $\Delta(A) = (V_A, \kappa_A, M_A)$, $\Gamma(b) = B$, $\Delta(B) = (V_B, \kappa_B, M_B)$, $\mathcal{B}' \subseteq \kappa_A$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}'(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D = \emptyset)$, $\forall G \in \mathcal{A}'(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq M_A)$, $\mathcal{U}' \subseteq V_A$, $\mathcal{A} = \mathcal{A}' \cup \{A\}$, $\mathcal{U} = \mathcal{U}' \cup \{A\}$.

From $\Delta, \Gamma \vdash_b P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')$ in (B.122) and $P' \to Q'$, according to the hypothesis, we can get $\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')$. According to (B.122), using (Amb) or (Sign Amb), we get

$$\frac{\Delta, \Gamma \vdash_b Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U}')}{\Delta, \Gamma \vdash_b a[Q']_s : (\emptyset, \mathcal{A}, \emptyset, \mathcal{U})} \tag{B.123}$$

(Red Par) $P = P'|R'$, $Q = Q'|R'$, and $P' \to Q'$. Suppose $\Delta, \Gamma \vdash_r P'|R' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r Q'|R' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

Using (Par), we have

$$\frac{\Delta, \Gamma \vdash_r P' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_r R' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_r P'|R' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.124}$$

where $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

From $\Delta, \Gamma \vdash_r P' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$ in (B.124) and $P' \to Q'$, according to the hypothesis, we can get $\Delta, \Gamma \vdash_r Q' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1)$. Using (Par), we can get

$$\frac{\Delta, \Gamma \vdash_r Q' : (\mathcal{B}_1, \mathcal{A}_1, \mathcal{T}_1, \mathcal{U}_1) \quad \Delta, \Gamma \vdash_r R' : (\mathcal{B}_2, \mathcal{A}_2, \mathcal{T}_2, \mathcal{U}_2)}{\Delta, \Gamma \vdash_r Q'|R' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})} \tag{B.125}$$

(Red BC) $P = P'\wr_C$, $Q = Q'\wr_C$, and $P' \to Q'$. Suppose $\Delta, \Gamma \vdash_r P'\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r Q'\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})$.

Using (Block), we have

$$\frac{\Delta, \Gamma \vdash_r P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_r P'\wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.126}$$

where $\mathcal{B} = \mathcal{B}' - type(C)$, $\forall G \in \mathcal{A}'(\Delta(G) = (V_G, \kappa_G, M_G) \implies M_G \subseteq type(C))$, $\forall \mathsf{auth}(D, K) \in \mathcal{T}'(\Delta(D) = (V_D, \kappa_D, M_D) \implies M_D \subseteq type(C))$.

From $\Delta, \Gamma \vdash_r P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})$ in (B.126), and $P' \to Q'$, according to the hypothesis, we can get $\Delta, \Gamma \vdash_r Q' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})$. Using (Block), we can get

$$\frac{\Delta, \Gamma \vdash_r P' : (\mathcal{B}', \mathcal{A}', \mathcal{T}', \mathcal{U})}{\Delta, \Gamma \vdash_r Q' \wr_C : (\mathcal{B}, \emptyset, \emptyset, \mathcal{U})} \tag{B.127}$$

(Red $\equiv$) $P' \equiv P$, $P \to Q$, $Q \equiv Q'$. Suppose $\Delta, \Gamma \vdash_r P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$, prove $\Delta, \Gamma \vdash_r Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

From $\Delta, \Gamma \vdash_r P' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $P' \equiv P$, according to Proposition 1, we can get $\Delta, \Gamma \vdash_r P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

From $\Delta, \Gamma \vdash_r P : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $P \to Q$, according to the hypothesis, we can get $\Delta, \Gamma \vdash_r Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$.

From $\Delta, \Gamma \vdash_r Q : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $Q \equiv Q'$, according to Proposition 1, we can get $\Delta, \Gamma \vdash_r Q' : (\mathcal{B}, \mathcal{A}, \mathcal{T}, \mathcal{U})$ and $Q \equiv Q'$.

□

Bibliography

[1] L. Cardelli and A. D. Gordon, Mobile ambients, in *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98*, Springer-Verlag, Berlin Germany, 1998.

[2] Digital signature standard (dss).

[3] S. Microsystems, Java security architecture, `http://java.sun.com/j2se/1.5.0/docs/guide/security/spec/security-spec.doc1.html`.

[4] S. Microsystems, Security in java 2, `http://java.sun.com/docs/books/tutorial/security1.2/summary/glossary.html`.

[5] J. L. V. Frontana, *Dynamic Binding of Names in Calculi for Mobile Processes*, PhD thesis, KTH, Stockholm, 2001.

[6] K. S. Søren Nøhr Christensen and M. Thrysøe, Umbrella, 2004.

[7] S. B. Anupam Palit, Bin Ren and Y. Sun, Digital signature for mobile ambients, 2005.

[8] D. Kotz and R. S. Gray, SIGOPS Oper. Syst. Rev. **33**, 7 (1999).

[9] R. Milner, *Communication and Concurrency* (Prentice-Hall, 1989).

[10] K. G. Luca Aceto and A. Ingólfsdóttir, An introduction to milner's ccs, BRICS,Department of Computer Science,Aalborg University,Denmark, 2005.

[11] J.-L. Vivas and M. Dam, From higher-order pi-calculus to pi-calculus in the presence of static operators, in *International Conference on Concurrency Theory*, pp. 115–130, 1998.

[12] DNSRD, Dns resources directory, `http://www.dns.net/dnsrd/`.

[13] T. A. S. Foundation, Authentication, authorization and access control, `http://httpd.apache.org/docs/1.3/howto/auth.html#access`.

[14] L. Cardelli, ACM Comput. Surv. **28**, 263 (1996).

[15] L. Cardelli, G. Ghelli, and A. D. Gordon, Lecture Notes in Computer Science **1877**, 365 (2000).

[16] L. Cardelli, Mobility and security, in *Foundations of Secure Computation*, edited by F. L. Bauer and R. Steinbrüggen, NATO Science Series, pp. 3–37, IOS Press, 2000.