UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

AUTENTICACIÓN DESMENTIBLE EN CANALES ANÓNIMOS

PRESENTACIÓN DEL TEMA DE MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

ALONSO EMILIO GONZÁLEZ ULLOA

PROFESOR GUÍA:
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
GONZALO NAVARRO BADINO
RODRIGO PAREDES MORADELA

SANTIAGO DE CHILE
MARZO 2011

Resumen Ejecutivo

El problema de comunicación anónima autentificada consiste en diseñar un protocolo que permita intercambiar mensajes entre un conjunto de participantes, de forma tal que cada emisor de un mensaje determinia el destinatario de su mensaje y, una vez que se envía el mensaje, este es efectivamente recibido por el destinatario determinado. La información que revela el protocolo en su ejecución debe mantener el anonimato, es decir debe ser tal que no permite a ningún adversario determinar información relacionada a las identidades de los participantes. El protocolo debe permitir a cada destinatario determinar con exactitud quién es el autor de cada mensaje que recibe, sin que esto contradiga el anonimato. Adicionalmente el protocolo debe mantener las garantías anteriores inclusive si es ejecutado en un ambiente concurrente, es decir es ejecutado con indeterminados otros protocolos.

Las aplicaciones de la comunicación anónima autentificada son variadas. Por ejemplo es útil para diseñar sistemas de denuncia anónima de delitos donde adicionalmente se desea discriminar la información recibida segun la identidad del enviador. Esto puede ser útil si algunos informantes son más creibles que otros.

En este trabajo se plantea el problema de comunicación anónima autentificada y se demuestra constructivamente la existencia de un protocolo que resuelve dicho problema. Para ello se estudian tópicos avanzados de Criptografía como *Universal Compossability*, *Generalized Universal Composability*, Anonimato, Desmentibilidad y las distintas primitivas criptográficas asociadas a dichos tópicos. Se definen rigurosamente las propiedades que debe tener un protocolo para resolver el problema planteado. Finalmente se diseña un protocolo eficiente para el cual se puede garantizar matematicamente que satisface las propiedades necesarias para resolver el problema de comunicación anónima autentificada.

Agradecimientos

Agradezco a ....

Alonso González Ulloa.

Índice General

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Para motivar el tema a tratar en este trabajo, presentamos el siguiente problema que perfectamente se podria presentar en un escenario real.
El tribunal de la libre competencia (TDLC) desea desarrollar una plataforma que permita a las distintas empresas presentar pruebas que inculpen a otra empresa en delitos como colusión, monopolio, etc. Para ello pondrá a disposición de las empresas un conjunto de servidores a los cuales enviar sus denuncias.
Para no amedrentar a una empresa denunciante es necesario que la comunicación entre la empresa y cada servidor sea anónima. Pues de este modo ningún observador podria determinar que una empresa a denunciado a otra de cierto delito y tomar represalias.
Por otro lado el TDLC confía más en el testimonio de algunas empresas que en el de otras, ya sea por su conducta anterior u otros factores. Por lo tanto desea estar completamente seguro de la identidad de una empresa cuando esta empresa presente pruebas usando la plataforma, pues así puede determinar un factor de credibilidad en la denuncia.
La plataforma correrá en internet, por lo tanto debe mantener sus propiedades de seguridad inclusive si es ejecutada concurrentemente con protocolos maliciosos.
Para desarrollar la plataforma el TDLC ha determiado que su problema es exactamente el siguiente:
Desea crear un protocolo mediante el cual las empresas se comuniquen con cada uno de los servidores de denuncias. La comuniciación entre empresas y servidores debe ser anónima y autentificada, y esto se debe cumplir inclusive si el protocolo es ejecutado concurrentemente con otros protocolos.

1.2. Descripción del problema

En general el problema anterior puede a aplicarse a cualquier de grupo de personas que desea comunicarse entre sí en una red (internet o una red local) y desea obtener garantías similares. A continuación iniciamos el camino de formalización del problema motivacional. La solución del problema consiste en encontrar un protocolo (un algoritmo distribuido) de cual se puedan garantizar matemáticamente las siguientes tres propiedades :

1. Anonimato
2. Autentificación desmentible
3. Componibilidad

Anonimato

A modo de ejemplo podemos considerar un protocolo "usual"de comunicación, un protocolo IP simplificado. En la figura 1.1 cada flecha de A a B indica que A envió un me saje a B. La etiqueta de una flecha de A a B indica el mensaje intercambiado en la ejecución de protocolo. Por ejemplo Empr-1 envió a Serv-1 el mensaje $(m_{e1s1}, ip_{e1}, ip_{s1})$, donde m_{e1s1} es el contenido del mensaje, ip_{e1} es la dirección IP de la Empresa 1 e ip_{s1} es la dirección IP del Servidor 1. Notemos que estos datos son necesarios para poder *rutear* los mensajes de un participante a otro, pero a la vez se revela a un obsevador que Empr-1 envió un mensaje a Serv-1. Por lo tanto podemos decir que **el protocolo IP simplificado no es anónimo** pues **existe un ataque**.

Para definir el anonimato resulta crucial definir formalmente que es considerado un ataque al anonimato, pues un protocolo será anónimo si y solo si no existe ningún ataque. En [14] se define un ataque con el siguiente juego. El observador o adversario determinada dos posibles ejecuciones del protocolo, las cuales difieren en qué mensajes serán enviados por quién y qué mensajes serán recibidos por quién. Entonces consideraremos que el adversario realiza un ataque si al ejecutar al adversario con cada una de las posibles ejecuciones del protocolo,

el adversario logra identificar cuál es cuál. Por lo tanto un protocolo sera seguro si y solo si cualquier adversario no logra distinguir una ejecución de otra.

Autentificación desmentible

En el protocolo de la figura 1.1 es posible que un adversario (rol que puede asumir una empresa $Empr-2$) se haga pasar por una empresa $Empr-1$ con buena reputación y denuncie falsamente a una empresa enemiga $Empr-3$. Para ello solo es necesario que modifique uno de sus mensajes cambiando ip_e2 por ip_e1. Por lo tanto decimos que el protocolo no implementa canales autenticados.

Con canales autentificado nos referimos protocolos en los cuales es posible estar seguro, con alta probabilidad, de quién es el autor de un mensaje. Sin embargo hay que ser cuidadoso con el protocolo de autentificación usado, pues los más conocidos (firmas digitales por ejemplo) poseen la propiedad de *non repudiability*. Esto es, que el emisor de un mensaje autentificado no puede negar a "la comunidad"que el es el autor del mensaje. Esto estaría contradiciendo el anonimato, pues el adversario también sería capaz de asociar la autoría de un mensaje a el emisor de este.

La autentificación desmentible, introducida en [12], se refiere a los protocolos que implementan canales anónimos con la propiedad adicional de que cada mensaje es autentificado a un receptor específico y el receptor no es capaz de probar a nadie más quién es el autor del mensaje.

Componibilidad

En general el hecho de implementar protocolos con ciertas garantías (por ejemplo anonimato y autentificación desmentible) no garantiza que dichas propiedades se sigan teniendo cuando el protocolo es ejecutado concurrentemente con otros protocolos.

En [4] Canetti introduce el *framework* criptográfico conocido como *Universal Composability* (desde ahora UC). UC propone una metodología definir y demostrar los objetivos de seguridad de un protocolo (por ejemplo anonimato y autentificación) de un protocolo. UC garantiza que el protocolo mantendrá su seguridad inclusive si es ejecutado concurrentemente con cualquier protocolo, siempre y cuando no comparta estado con el protocolo analizado.

Cuando el protocolo sí comparte estado con otros protocolo es necesario hacer uso del *framework Generalized Universal Composabillity* (desde ahora GUC), que generaliza a UC. In-

formalmente GUC propone una metodología para incluir el estado que un protocolo podría compartir con otros.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar un protocolo criptográfico eficiente que cumpla nociones razonables de Anonimato y Autentificación Desmentible. Demostrar matemáticamente su seguridad usando herramientas modernas de análisis criptográficas (GUC), y estudiar su relación con otras primitivas criptográfias.

1.3.2. Objetivos esecíficos

1. Estudiar conceptos asociados a interacciones desmentibles y las técnicas y primitivas criptográficas asociadas (encriptación y mecanismos de auten- tificación como firmas digitales y esquemas de identificación).
2. Estudiar conceptos de anonimato y las técnicas y primitivas criptográficas asociadas.
3. Proponer un protocolo que combine ambas nociones.
4. Analizar dicho protocolo en términos de efectividad y eficiencia.
5. Analizar la efectividad en términos de las garantías de seguridad obtenidas.
6. Analizar la eficiencia en términos de los costos en tiempo asociados al protocolo.
7. Estudiar su relación con otros protocolos criptográficos.

En la figura 1.2 se muestra un diagrama para explicar nuestra solución.

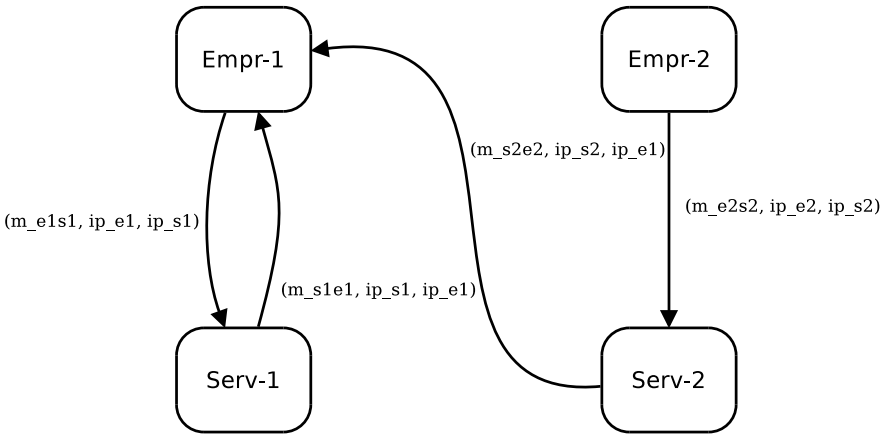

Figura 1.1: Protocolo simple de comunicación

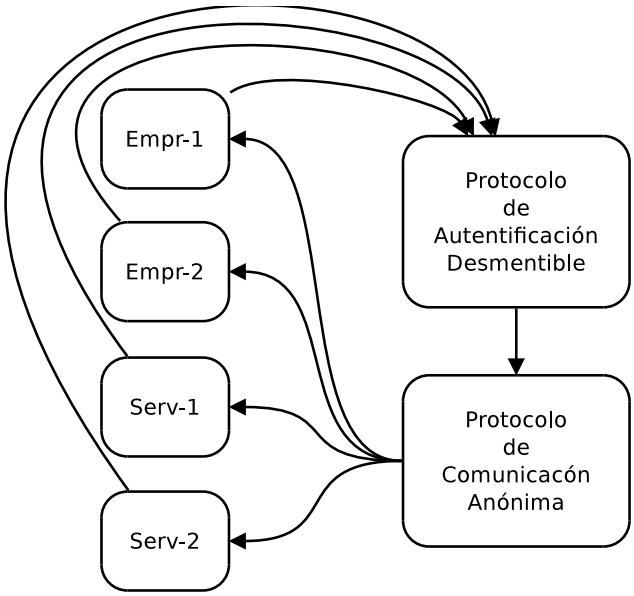

Figura 1.2: Solución propuesta

Capítulo 2

Marco Teórico

En este capítulos hacemos distintas definiciones básicas útiles para este trabajo.

2.1. Definiciones básicas

Definición 1 (String)**.** *Decimos que ω es un string si $\omega \in \{0,1\}^*$.*

Definición 2 (Largo de un string)**.** *Para un string ω denotamos por $|\omega|$ al entero k talque k es el número de caracteres de $|\omega|$.*

Definición 3 (Función despreciable)**.** *Decimos que una función $\eta : \mathbb{N} \to \mathbb{N}$ es despreciables si crece más lento que el inverso de cualquier polinomio. Es decir, para todo polinomio p existe un n_0 talque para todo $n > n_0$ se cumple que*

$$\eta(n) < \frac{1}{p(n)}$$

Las funciones despreciables permiten definir probabilidades "pequeñas" para cualquier variable aleatoria "algorítmica" (es decir, que corresponden a la salida de un algoritmo). La definición busca eliminar el efecto de ejecutar una cantidad polinomial de veces un algoritmo (usar el lema de amplificación).

Definición 4 (Grupo Abeliano)**.** *Decimos que el par $(G, +)$, donde G es un conjunto y $+ : G^2 \to G$, es un grupo abeliano si se cumplen las siguientes propiedades:*

1. *(Asocioatividad) $\forall a, b, c \in G\ (a + b) + c = a + (b + c)$.*
2. *(Elemento neutro) $\exists 0 \in G$ talque $\forall a \in G\ a + 0 = 0 + a = a$.*
3. *(Inverso) $\forall a \in G\ \exists -a \in G$ talque $a + -a = 0$.*
4. *(Conmutatividad) $\forall a, b \in G\ a + b = b + a$.*

En general cuando decimos que G es un grupo nos referimos a que es un grupo abeliano.

Definición 5 (Generador)**.** *Decimos que $g \in G$ es un generador de G si el conjunto generado por g $\langle g \rangle = \{g^n | n \in \mathbb{N}\}$ es igual a G*

Definición 6 (Grupo cíclico)**.** *Decimos que G_q es un grupo cíclico de orden q si existe un generador g de G y $|G| = q$.*

2.2. Probabilidades discretas

Definición 7 (Espacio de probabilidades finito)**.** *Una espacio de probabilidades finito es un conjunto finito* $\Omega = \{\omega_1, \ldots, \omega_{|\Omega|}\}$ *con un conjunto de números* $p_1, \ldots, p_{|\Omega|} \in [0,1]$ *tales que* $\sum_{i=1}^{|\Omega|} p_i = 1$

A menos que se especifique lo contrario, asumimos que la distribución de Ω es uniforme, es decir $p_i = \frac{1}{|\Omega|}$. Al experimento de escoger un ω en Ω al azar denotamos por $\omega \in_R \Omega$.

Definición 8 (Evento)**.** *Decimos que* A *es un evento de* Ω *si* $A \subseteq \Omega$*. Denotamos por* $\Pr[A] = \sum_{\omega_i \in A} p_i$ *a la probabilidad de* A

Definición 9 (Variable aleatoria)**.** *Una variable aleatoria es una función* $X : \Omega \to \mathbb{R}$ *que asigna a cada* $\omega \in \Omega$ *un número real* $x \in \mathbb{R}$*.*

Definición 10 (Distribución)**.** *Dada una varibale aleatoria, definimos su distribución como la función* $f : \mathbb{R} \to [0,1]$ *talque* $f(x) = \sum_{i:X(\omega_i)=x} p_i$

Definición 11 (Distancia estadística)**.** *Dadas dos variables aleatoria* X *e* Y *definidas sobre* Ω*, definimos la distancia estadística entre ellas como*

$$\Delta(X,Y) = \max_{S \subseteq \Omega} \{X(S) - Y(S)\}$$

2.2.1. Indistinguibilidad

La noción de indistinguibilidad es una noción fundamental en criptografía pues ha permitido definir la seguridad mediante la similitud entre distintos eventos.

La indistinguibilidad es una noción de similaridad entre familias de variables aleatorias (o conjuntos de variables aleatorias) y existen tres tipos de indistinguibilidad.

Definición 12 (Indistinguibilidad perfecta)**.** *Decimos que dos familias de variables aleatorias* $U = \{U_x\}_{x \in \{0,1\}^*}$ *y* $V = \{V_x\}_{x \in \{0,1\}^*}$ *son perfectamente indistinguibles si* $\Delta(U_x, V_x) = 0 \quad \forall x$*. Denotamos la indistinguibilidad perfecta por* $U = V$

Definición 13 (Indistinguibilidad estadística)**.** *Decimos que dos familias de variables aleatorias* $U = \{U_x\}_{x \in \{0,1\}^*}$ *y* $V = \{V_x\}_{x \in \{0,1\}^*}$ *son estadísticamente indistinguibles si* $\Delta(U_x, V_x)$ *es despreciable en* $|x|$*. Denotamos la indistinguibilidad estadística por* $U \approx Y$*.*

Definición 14 (Indistinguibilidad computacional)**.** *Decimos que dos familias de variables aleatorias* $U = \{U_x\}_{x \in \{0,1\}^*}$ *y* $V = \{V_x\}_{x \in \{0,1\}^*}$ *son estadísticamente indistinguibles para todo algoritmo polinomial aleatorizado* A *existe una función despreciable* η *y un entero* n_0 *talque si* $|x| > n_0 \ |\Pr[A(U_x) = 1] - \Pr[A(V_x) = 1]| \leq \eta(|x|)$*. Denotamos la indistinguibilidad computacional por* $U \overset{c}{\approx} V$*.*

2.3. Definiciones criptográficas básicas

2.3.1. El problema de Diffie-Hellman decisonal (DDH)

El problema DDH [10] es un problema que se considera difícil, es decir se conjetura que no existe un algoritmo polinomial que lo solucione, para ciertos grupos. DDH es una herramienta fundamental en la Criptografía moderna pues permite la construcción eficientes primitivas con una variada aplicabilidad.

Definición 15 (DDH). *Sea G_q un grupo cíclo de orden q. Decimos que en G_q se cumple DDH si $(g^\alpha, g^\beta, g^\gamma) \overset{c}{\approx} (g^\alpha, g^\beta, g^{\alpha\cdot\beta})$ dado que $\alpha, \beta, \gamma \in_R G_q$.*

2.3.2. Esquemas de Encriptación

Decimos que $\mathcal{E} = (K, E, D)$, con conjunto de textos planos M y conjunto de textos cifrados C,es un esquema de encriptación simétrico si para todo $r \in \{0,1\}^\kappa$, dado $k = K(\kappa; r)$, para todo $m \in M$ $D_k(E_k(m)) = m$.

Decimos que $\mathcal{E} = (K, E, D)$, con conjunto de textos planos M y conjunto de textos cifrados C,es un esquema de encriptación asimétrico si para todo $r \in \{0,1\}^\kappa$, dado $(sk, pk) = K(\kappa; r)$, para todo $m \in M$ $D_{sk}(E_{pk}(m)) = m$.

Indistinguibilidad bajo ataque de texto plano escogido (IND-CPA)

También conocida como *seguridad semántica* es una definición de privacidad de un esquema critográfico. Apunta a que un esquema de encriptación $\mathcal{E} = (K, E, D)$ es seguro si ningún atacante es capaz de dinstinguir entre la encriptación de cualesqueir par de mensajes m_1, m_2.

Definición 16 (Experimento IND-CPA). *Definimos el experimento IND-CPA parametrizado por un aloritmo $\mathcal{A}$, un esquema de encriptación $\mathcal{E}$, parametro de seguridad κ y un bit b, denotado por $\mathrm{EXP}_{\mathcal{E}}^{\mathrm{IND-CPA}}(\mathcal{A}, b)$, como la variable aleatoria resultante de ejecutar el experimento descrito en la figura 2.1*

Definición 17 (Seguridad IND-CPA). *Decimos que un esquema de encriptación $\mathcal{E}$ es IND-CPA seguro si*

$$\mathrm{EXP}_{\mathcal{E}}^{\mathrm{IND-CPA}}(\mathcal{A}, 1) \approx \mathrm{EXP}_{\mathcal{E}}^{\mathrm{IND-CPA}}(\mathcal{A}, 0)$$

2.3.3. Esquemas de autentificación de mensajes

Decimos que $\mathcal{MA} = (K, T, V)$, con M el conjunto de mensajes $\{0,1\}^\kappa$ el conjunto de claves, es un esquema de autentificación de mensajes si K es un algoritmo de generación de claves y para todo $m \in M$ y todo $k \in \{0,1\}^\kappa$ se tiene que $v(m, T_k(m)) = 1$.

El experimento IND-CPA ejecutado con adversario $\mathcal{A}$, esquema de encritación $\mathcal{E} = (K, E, D)$, un bit $b \in \{0, 1\}$ y parámetro de seguridad κ procede como sigue:

1. Escoger las clave del esquema, $k \xleftarrow{R} K(\kappa)$.
2. Ejecutar al adversario $\mathcal{A}$ y cada vez que escriba (m_1, m_2) retornale $E_k(m_b)$.
3. Retornar lo que $\mathcal{A}$ retorne.

Figura 2.1: El experimento IND-CPA

El experimento UF-CMA ejecutado con adversario $\mathcal{A}$, esquema de autetificación de mensajes $\mathcal{MA} = (K, T, V)$ y parámetro de seguridad κ procede como sigue:

1. Escoger la clave del esquema, $k \xleftarrow{R} K(\kappa)$ e inicializar $\Gamma \leftarrow \emptyset$.
2. Ejecutar al adversario $\mathcal{A}$ y cada vez que escriba m retornale $T_k(m)$ y actualizar $\Gamma \leftarrow \Gamma \cup \{m\}$.
3. Cuando $\mathcal{A}$ escriba (m, t), si $m \notin \Gamma$ y $T_k(m) = t$ retornar 1 y 0 de lo contrario.

Figura 2.2: El experimento UF-CMA

Infalsificabilidad ante ataques de mensajes escogidos UF-CMA

UF-CMA es una noción de seguridad que aplica a esquemas de autentificación de mensajes [1]. Un esquema de firmas es UF-CMA si ningún adversario es capaz de falsificar la firma de un mensaje. Formalizamos esta noción con las siguientes definiciones.

Definición 18 (Ventaja UF-CMA)**.** *Definimos la ventaja UF-CMA de un adverario $\mathcal{A}$ con el esquema $\mathcal{MA} = (K, T, V)$ como:*

$$\mathrm{Adv}^{UF\text{-}CMA}_{\mathcal{MA},\mathcal{A}}(\kappa) = \Pr[UF\text{-}CMA_{\mathcal{MA},\mathcal{A}}(\kappa) = 1]$$

donde $UF\text{-}CMA_{\mathcal{MA},\mathcal{A}}(\kappa)$ es la variable aleatoria obtenida al ejecutar el experimento de la figura 2.2.

Definición 19 (Seguridad UF-CMA)**.** *Decimos que un esquema de autentificación de mensajes $\mathcal{MA}$ es UF-CMA seguro si para todo adversario polinomial aleatoriazdo $\mathcal{A}$ existe una función despreciable η y un κ_0 talque para todo $\kappa > \kappa_0$*

$$\mathrm{Adv}^{UF-CMA}_{\mathcal{MA},\mathcal{A}}(\kappa) \leq \eta(\kappa)$$

[1] A veces, si $\mathcal{MA} = (K, T, V)$ y T corresponde a un algoritmo MAC, con MAC nos referimos indistintamente al algoritmo T y al esquema $\mathcal{MA}$.

La funcionalidad ideal $\mathcal{F}_{CRS}^{D}$ parametrizada por una distribución D procede como sigue:

1. Si recibe (`CRS`) de P, verificar que $P \in \mathcal{P}$, donde $\mathcal{P}$ es un conjunto de identidades, si $P \notin \mathcal{P}$ ignorar a P.
2. Si no hay un valor r registrado escoger y registrar $r \xleftarrow{R} D$.
3. Enviar $(\texttt{CRS}, r)$ a P.

Figura 2.3: La funcionalidad ideal $\mathcal{F}_{CRS}$

2.3.4. Supestos de incialización

Los supuestos de incialización son necesarios para implementar ciertos protocolos criptográficos que de otra forma serían imposible de realizar. Los supuestos de incialización corresponden a funcionalidades ejecutadas por una entidad confiable, y en UC son representados por una funcionalidad ideal. A continuación listamos dos supuestos de inicialización.

String público aleatorio CRS

También conocido como string público de referencia, corresponde a una entidad confiable que publica un string con una distribución fija. En general se considera que la distribución del string es una uniforme.
En UC se modela CRS con la funcionalidad ideal $\mathcal{F}_{CRS}$ descrita en la figura 2.3i

Interfaz de clave pública PKI

Una interfaz de clave pública o PKI (del inglés *Public Key Interface*) es un supuesto de inicialización que permite registrar pares de llaves públicas/provadas en una base de datos confiable y autentificada.
En GUC una forma de modelar PKI es con la funcionalidad compartida $\bar{\mathcal{G}}_{KRK}$ definida en la figura 2.4. Notamos que $\bar{\mathcal{G}}_{KRK}$ viene parametrizada por un conjunto Φ, que corresponde a los protocolos a los cuales les es permitido obtener claves privadas.

La funcionalidad compartida $\bar{\mathcal{G}}_{KRK}$ parametrizada por una conjunto de protocolos permitidos Φ, un algoritmo de generación de claves `Gen` y parámetro de seguridad κ, procede como sigue:

1. Si recibe un mensaje (`register`) de un participante honesto P_i que aún no se ha registrado escoger $r \xleftarrow{R} \{0,1\}^\kappa$, luego calcular $(pk_i, sk_i) \leftarrow \texttt{Gen}^\kappa(r)$ y guardar la tupla (P_i, pk_i, sk_i).

2. Si recibe un mensaje $(\texttt{register}, r)$ de un participante corrupto P_i que aún no se ha registrado calcular $(pk_i, sk_i) \leftarrow \texttt{Gen}^\kappa(r)$ y guardar la tupla (P_i, pk_i, sk_i).

3. Si recibe un mensaje $(\texttt{retrieve}, P_i)$ de P_j, si existe una tupla registrada (P_i, pk_i, sk_i) retornar (P_i, pk_i). De lo contrario retornar $(P_i, \perp)$.

4. Si recibe un mensaje $(\texttt{retrievesecret}, P_i)$ de P_j, si $i = j$, P_j es corrupto esta corriendo un código en Φ y existe una tupla registrada (P_i, pk_i, sk_i) retornar (P_i, sk_i). De lo contrario retornar $(P_i, \perp)$.

Figura 2.4: La funcionalidad compartida $\bar{\mathcal{G}}_{KRK}$

Capítulo 3

Modelo Criptográfico

Demostremos las garantías de seguridad de nuestro protocolo en el *framework Generalized Universal Composability* (GUC). GUC [7] es una generalización del *framework Universal Composability* [4]. Ambos sirven para modelar protocolos criptográficos concurrentes, pero GUC adicionalmente modela protocolos concurrentes que comparten estado entre si. Primero revisaremos el framework UC pues GUC se construye a partir de UC con pequeñas, pero muy significativas, modificaciones.

A modo de recomendación instamos al lector que desea tener una visión general de UC y GUC leer las secciones 3.1.1, 3.2.1 y 3.3. Las demás secciones ahondan en detalles útiles para revisar en detalle los resultados de este trabajo, pero quizás engorrosos para tener una primera ideal general de estos.

3.1. Universal Composability framework (UC)

3.1.1. Descripción general

UC es una metodología para modelar modularmente protocolos criptográficos que son ejecutados en redes del "mundo real"(por ejemplo internet). El espíritu de UC es diseñar un protocolo y luego abstraer la seguridad que uno espera de él en un protocolo ideal, ejecutado en condiciones especiales que garantizan su seguridad. Luego se debe demostrar que ejecutar el protocolo real y ejecutar el protocolo ideal es esencialmente lo mismo, por lo tanto el protocolo real es tan seguro como el protocolo ideal.

Los protocolos reales pueden ser ejecutados concurrentemente con muchos otros protocolos y también pueden ser ejecutados distribuidos entre varios participantes. El protocolo puede ser monitoreado por terceros y algunos participantes pueden salirse arbitrariamente del pro-

tocolo atentando con la seguridad. En UC, todo el posible mal comportamiento es ejecutado por una sola máquina, el Adversario (real) $\mathcal{A}$. En una ejecución del protocolo el adversario puede espiar y manipular todos los mensajes intercambiados, también puede manejar la distribución de mensajes a su antojo, y además puede *corromper* participantes del protocolo y ejecutar código arbitrario en ellos.

Por otro lado están los protocolos ideales, llamados *funcionalidades ideales* denotadas por $\mathcal{F}$. Las funcionalidades ideales son ejecutada por una entidad confiable e incorruptible. Para las funcionalidades ideales también existe un adversario ideal o *simulador* $\mathcal{S}$, pero este no puede espiar ni controlar la comunicación más de lo que la funcionalidad ideal permite.

La configuración en la que el protocolo real es ejecutado con el adversario real es conocida como *mundo real*, y la configuración en que la funcionalidad ideal es ejecutada con el adversario ideal es conocida como *mundo ideal*. En ambos mundos la ejecución del protocolo concurrentemente con otros protocolos esta a cargo de una máquina especial conocida como el ambiente y denotado por $\mathcal{Z}$. De este modo en UC los protocolos pueden ser analizados aislados de resto del mundo.

Para demostrar que el protocolo real alcanza la seguridad deseada se debe tener que, para cualquier ejecución del protocolo en el mundo real y para cualquier estrategia adversarial (esto es para todo ambiente y para todo adversario) existe una estrategia adversarial con recursos limitados (un adversario ideal) que tiene el mismo efecto que la estrategia adversarial real (el ambiento no es capaz de percatarse de ninguna diferencia entre la ejecución del protocolo real y el protocolo ideal).

Para definir formalmente las nociones intuitivas descritas anteriormente es necesario introducir un modelo de cálculo conocido como *Máquina de Turing Interactiva* (desde ahora ITM), más precisamente sistemas de ITM. Las definiciones que vienen a continuación sigue, salvo pequeñas modificaciones, a la definiciones dadas po Canetti en [6].

3.1.2. Sistemas de Máquinas de Turing Interactivas

Las Máquinas de Turing Interactivas corresponden a Máquinas de Turing con cintas especiales que pueden ser escritas externamente. A dichas cintas las llamamos escribibles externamente (EW), y son de escritura única, es decir, el cabezal siempre se mueve a la derecha.

Definición 20. *Una Máquina de Turing Interactiva M es una Máquina de Turing con las*

siguientes cintas:

1. *Una cinta EW de identidad.*
2. *Una cinta EW del parámetro de seguridad.*
3. *Una cinta EW de entrada.*
4. *Una cinta EW de comunicación entrante.*
5. *Una cinta EW de salidas de subrutinas.*
6. *Una cinta de salida.*
7. *Una cinta de bits aleatorios.*
8. *Una cinta de activación, de lectura y escritura y de 1 bit de tamaño.*
9. *Una cinta de lectura y escritura para trabajo.*

La cinta de identidad contiene un string que representa la identidad de M, que se interpreta como si estuviera compuesto por dos substrings: el identificador de sesión (SID) y el identificador de participante (PID). Identificamos a cada instancia de una ITM (ITI) por el par $\mu = (\langle M \rangle, id)$, con $\langle M \rangle$ el código de M y id el contenido de la cinta de identidad. En general omitimos los $\langle\rangle$ y con M nos referimos tanto a la máquina como al código.
La cinta del parámetro de seguridad contiene un string de la forma 1^k, con k el parámetro de seguridad [1].
La cinta de salida contendrá la salida de M una vez que haya terminado.
La cinta de bits aleatorios contiene suficientes bits aleatorios para que M pueda realizar sus cálculos.
La cinta de trabajo es la usual cinta de trabajo de las Máquinas de Turing.
La cinta de activación tiene el valor 0 si la M no esta activada y 1 si lo esta. La secuencia de configuraciones [2] de una ejecución de M

esta compuesta por subsecuencias en las que en cada configuración la M esta activada. A dichas subsecuencias se les conoce como *secuencias de activación.*
Las otras cintas toman importancia cuando M es ejecutada "conectadaçon otras máquina en

[1] El parámetro de seguridad indica el nivel de seguridad en el cual se esta ejecutando la máquina, y en general mientras crece se debería tener que la seguridad del protocolo ejecutado con la máquina también crece.

[2] Una configuración corresponde a un objeto que determina completamente un instante en la computación de una MT. Podemos ver la ejecución de una MT como una secuencia de configuraciones, donde la primera configuración corresponde a la MT en su estado inicial y la(s) cinta(s) con la(s) entrada(s), y la configuración final corresponde a la MT en un estado final.

un Sistema de ITMs (sITM).

Definición 21. *Un Sistema de ITMs S viene dado por $S = (I, C)$, donde I es la ITM inicial y C es la función de control $C : \{0,1\}^* \to \{0,1\}$.*

Inicialmente la ITI $\mu_0 = (I, 0)$ es activada, el sITM terminará cuando I termine y su salida sera lo que I deje en su cinta de salida.
Una ITI $\mu = (M, id)$ puede escribir en una de las cintas de otra ITI $\mu' = (M', id')$, para ello es necesario que μ ejecute una instrucción especial llamada `escritura-externa` y debe especificar la cinta en de μ' en la que quiere escribir y los datos a escribir en esa cinta [3]. La semántica de la instrucción de `escritura-externa` es como sigue:

1. Si la función de control C aplicada a toda la secuencia de instrucciones `escritura-externa` que se han realizado hasta ahora retorna 0, entonces la instrucción es ignorada.

2. Si C retorna 1 pero no existe una ITI en el sITM $\mu'' = (M'', id'')$ talque $id'' = id'$, se crea una nueva ITI con código M' y con identidad id'. Para ello se crea una nueva ITM que en la cinta de identidad contiene id', el la cinta del parámetro de seguridad contiene 1^k y en la cinta de bits aleatorios contiene suficientes bits aleatorios. A continuación se evalúa el punto siguiente.

3. Si C retorna 1 y existe una ITI en el sITM $\mu'' = (M'', id'')$ talque $id'' = id'$:

   a) Si la cinta objetivo de la instrucción era la cinta de comunicación entrante de μ', entonces los datos especificados son escritos en la cinta de comunicación entrante de μ'' y μ'' es activada. Notemos que esto es hecho independiente de si el código de μ'' es el mismo código especificado por μ, con el fin de rescatar que una ITI no conoce el código de la ITI con que se comunica a través de escrituras en la cinta de comunicación entrante.

   b) Si la cinta objetivo era la cinta de entrada de μ' y $M' = M''$, entonces los datos especificados son escritos en la cinta de entrada de μ'' y μ'' es activada. En este

[3] Formalmente podríamos decir que M entra en un estado especial y en su cinta de trabajo se encuentra un string x que determina μ', la cinta objetivo y los datos a escribir en la cinta objetivo

caso la instrucción modela llamados a otras ITIs como subrutina, dentro de un entorno seguro (μ conoce el código que esta ejecutando μ'').

c) Si la cinta objetivo es la cinta de salida de subrutina de μ', entonces los datos especificados son escritos en la cinta de salida de subrutina de μ''. En este caso la instrucción modela el retorno de una llamada a subrutina, en que la subrutina no conoce el código de la ITI que la llamó.

Adicionalmente consideramos funciones de control *extendidas* las cuales, además de permitir o no permitir instrucciones `escritura-externa`, son capaces de traducir una instrucción de `escritura-externa` en otra. Por ejemplo, una función de control extendida puede obligar a que todas las ITM creadas tengan un código preespecificado M traduciendo cualquier instrucción `escritura-externa` con código N a una instrucción de `escritura-externa` con código M. Los sITM con función de control extendida los llamaremos sITM extendidos.

Como veremos más adelante, nos interesará especialmente la salida de un sITM.

Definición 22. *Denotamos a la variable aleatoria obtenida al ejecutar el sITM (I, C) con parámetro de seguridad k, entrada x y escogiendo los bits aleatorios necesarios para su ejecución al azar por* $\mathrm{OUT}_{(I,C)}(k, x)$. $\mathrm{OUT}_{(I,C)}$ *denota a la familia de variables aleatorias* $\{\mathrm{OUT}_{(I,C)}(k, x)\}_{k\in\mathbb{N},x\in\{0,1\}^*}$.

Como es usual nos limitaremos a sITMs "eficientes", es decir el tiempo completo de ejecución esta acotado por $p(|x| + k)$, con p un polinomio. Se puede garantizar que un sITM (I, C) es eficiente si la ITM inicial es una PPT (del inglés *Probabilistic Polynomial time Turing machine*).

Definición 23. *Sea $p : \mathbb{N} \to \mathbb{N}$. Decimos que una ITM M esta localmente $p-$acotada si, en cualquier punto de su ejecución, el numero total de pasos tomados por M es a lo más $p(n)$, donde*

$$n = k + n_I - n_O - k \cdot n_N,$$

k es el parámetro de seguridad, n_I es el numero total de bits escritos en la cinta de entrada de M, n_O es el numero total de bits escritos por M a otras ITMs, y n_N es el número total de otras ITMs en las que M escribe.

Definición 24. *Si una ITM M esta localmente $p-$acotada y además M solo escribe en ITMs localmente $p-$acotadas, entonces decimos que M es $p-$acotada.*

Definición 25. *Una ITM M es una PPT si existe un polinomio p talque M es $p-$acotada.*

3.1.3. Ejecución de un protocolo

Definamos primero qué es un protocolo y una instancia de un protocolo.

Definición 26 (Protocolo PPT)**.** *Un protocolo PPT (o simplemente protocolo) π es la PPT que contiene el código a ejecutar por cada participante del protocolo.*

Definición 27 (Instancia de un protocolo)**.** *Dado un sITM S, una instancia de un protocolo π con SID sid es el conjunto de ITIs cuyo código es π y cuyo SID es sid.*

Definición 28 (Participante)**.** *Una ITI μ es un participante de una instancia de π con SID sid si el SID de μ es sid.*

Definición 29 (Sub-participante)**.** *Una ITI μ es sub-participante de una sesión del protocolo π si algún participante o sub-participante de de la sesión de π escribe en la cinta de comunicación entrante o en la cinta de input de μ.*

La ejecución de un protocolo en UC está parametrizada por tres ITMs:

- El protocolo a ser ejecutado π.
- El ambiente $\mathcal{Z}$.
- El adversario $\mathcal{A}$ (o $\mathcal{S}$).

Con las tres ITMs construimos el sITM extendido $(\mathcal{Z}, C^{\pi,\mathcal{A}}_{\text{EXEC}})$. La función de control $C^{\pi,\mathcal{A}}_{\text{EXEC}}$ básicamente se encarga de que la primera ITM invocada por $\mathcal{Z}$ sea $\mathcal{A}$, y de que $\mathcal{Z}$ solo invoque ITMs con código π y SID fijo. Adicionalmente se encarga de que todas las corrupciones, instrucciones ejecutadas por el adversario que le permiten tomar control de un participante, sean producto de una llamada del ambiente al adversario. En la figura 3.1 se detalla su funcionamiento.

Denotamos la salida de la ejecución de un protocolo π en UC por $\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi} = \text{OUT}_{(\mathcal{Z},C^{\pi,\mathcal{A}}_{\text{EXEC}})}$

3.1.4. Teorema de composición

La noción principal que entrega UC es la emulación de protocolos, que viene dada por la inhabilidad de $\mathcal{Z}$ de distinguir entre la ejecución de dos protocolos.

Definición 30. *Decimos que un protocolo π UC-emula a otro protocolo ϕ si y solo si para cualquier ambiente $\mathcal{Z}$ y para cualquier adversario $\mathcal{A}$ existe un adversario $\mathcal{S}$ talque*

$$\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi} \approx \text{EXEC}_{\mathcal{Z},\mathcal{S},\phi}$$

La función de control $C^{\pi,\mathcal{A}}_{\text{EXEC}}$ ejecutada con ambiente $\mathcal{Z}$, adversario $\mathcal{A}$ y protocolo π procede como sigue:

1. Para el ambiente $\mathcal{Z}$:

   a) El código de la primera instrucción `escritura-externa` es cambiado por el código de $\mathcal{A}$.

   b) Para cualquier otra instrucción `escritura-externa`, si la ITI objetivo es distinta de $\mathcal{A}$ entonces cambiar el código por π.

2. Cada vez que el adversario $\mathcal{A}$ corrompe a un participante escribiendo la instrucción `escritura-externa` donde la ITI objetivo es otro participante con el parámetro especial `corrupt`, la función de control se asegura que anteriormente $\mathcal{Z}$ haya escrito en $\mathcal{A}$ la instrucción de corrupción.

3. Cada vez que un participante o subparticipante escribe ejecuta la instrucción `escritura-externa` con ITI objetivo otro participante o subparticipante, es el adversario $\mathcal{A}$ quien es activado y en quien es escrito el contenido de la instrucción `escritura-externa`.

Figura 3.1: La función de control $C^{\pi,\mathcal{A}}_{\text{EXEC}}$

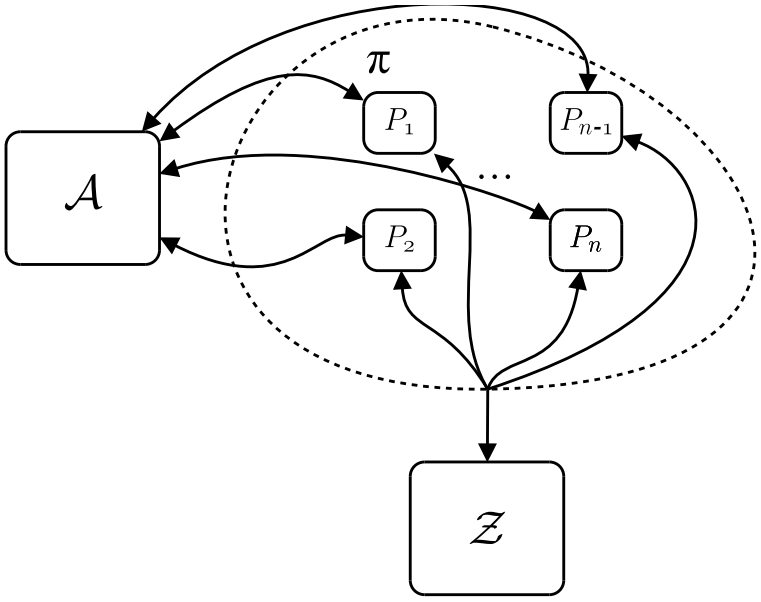

Figura 3.2: Ejecución de un protocolo en UC

Supongamos que tenemos un protocolo π, que llama como subrutina a ρ. Si ningún ambiente puede distinguir entre la ejecución del protocolo ρ de otro protocolo ϕ, es útil preguntarse si se pueden cambiar todas las llamadas de π a ρ por llamadas a ϕ sin cambiar el funcionamiento de π. La conjetura anterior es corroborada por el teorema 1, el teorema de composición. Definamos primero que es "cambiar todas las llamadas a ρ por llamadas a ϕ".

Definición 31 (Operador Composición)**.** *Sea π un protocolo que (posiblemente) hace llamadas a ρ. Definimos el operador composición / de forma tal que $\pi^{\rho/\phi}$ es idéntico que el protocolo π salvo que cada instrucción* `escritura-externa` *con código ρ, es cambiada por la misma instrucción* `escritura-externa` *con código ϕ.*

Una propiedad adicional que necesitan cumplir los protocolos es que sean *subroutine respecting*, que básicamente significa que no comparten estado.

Definición 32 (Protocolo subroutine respecting)**.** *Decimos que un protocolo ρ es subroutine respecting si ningún participante o sub-participante de ρ pasa input o recibe output de una ITI que es participante o sub-participante de otra instancia de algún protocolo.*

Teorema 1 (Composición)**.** *Sean π, ρ, ϕ protocolos tales que ρ UC-emula a ϕ y ρ y ϕ son protocolos subroutine respecting. Entonces el protocolo $\pi^{\rho/\phi}$ UC-emula al protocolo π.*

La demostración del teorema 1 se puede encontrar en [3], a continuación mostramos intuitivamente porque el teorema se debería cumplir.

Notemos que el protocolo π puede invocar una cantidad indeterminada de protocolos que se pueden ejecutar concurrentemente con con ρ. Es precisamente por eso que $\pi^{\rho/\phi}$ podría no UC-emular a π, pues en la configuración en la cual es ejecutado ϕ, donde ningún ambiente puede distinguirlo de ρ, se ejecuta solamente una instancia de ϕ y de ningún otro protocolo más. Lo que permite concluir acerca de $\pi^{\phi/\rho}$ es la cuantificación sobre todos los ambientes y que tanto ρ como ϕ son protocolos *subroutine respecting*. Al cuantificar sobre todos los ambientes estamos diciendo que, en particular, la UC-emulación también se tiene para ambientes que simulan internamente ejecuciones de otros protocolos como las que podría hacer π. El único problema podría ser que las ejecuciones en paralelo de otros protocolos estén "correlacionadas" de alguna forma con ρ, pero esto no es posible pues al ser ρ *subroutine respecting* su ejecución es independiente de cualquier otro protocolo. Dicho de otra forma, para que un protocolo este "correlacionado" con ρ es necesario que ρ lo llame directamente, o que ambos llamen a un protocolo en común. El primer caso no es problemático, pues el protocolo correlacionado seria un subprotocolo de ρ y sería parte de una ejecución de ρ. El segundo caso no es posible, pues el protocolo que ρ compartiría sería una instancia externa

y contradeciría que ρ es *subroutine respecting*.

Otra forma de interpretar el teorema de composición es que nos permite asumir que el protocolo π hace llamadas a ϕ y no a ρ, lo cual puede ser útil cuando no sabemos si ρ es seguro pero asumimos que ϕ si lo es. En tal caso ϕ sería una *funcionalidad ideal*.

3.1.5. Funcionalidades Ideales

Para definir la seguridad de un protocolo en UC es necesario diseñar una funcionalidad ejecutada por una entidad confiable en un protocolo que consideramos seguro. A la funcionalidad se le llama *funcionalidad ideal* y generalmente se le denota por la letra $\mathcal{F}$. Al protocolo se le llama *protocolo ideal* y se denota por IDEAL$_{\mathcal{F}}$.Todo el cómputo del protocolo es realizado por $\mathcal{F}$, pues los participantes solo ejecutan el código del *participante tontito*

Definición 33 (Participante tontito)**.** *Una instancia del participante tontito, denotado por $\tilde{P}$, es una ITI cuyo código es el siguiente:*

1. *Si recibe m, escribe m en cinta de entrada de la funcionalidad $\mathcal{F}$.*
2. *Si $\mathcal{F}$ retorna m, escribe m en su cinta de output.*

La funcionalidad ideal recibirá entradas de los participantes tontitos y escribirá en su cinta de salida de subrutina los resultados del cómputo que corresponden a cada participante. Por lo tanto su funcionamiento entrada/salida debe ser el deseado, esto es la funcionalidad debe ser *correcta*. Adicionalmente debe rebelar solo la información necesaria, de modo tal que la consideremos *segura*. Una ITI que ejecuta el código de una funcionalidad ideal sera creada igual que cualquier otra ITI, a excepción que su cinta de identidad tendrá como contenido un string de la forma $sid||\perp$, de modo tal que $\perp$ es un símbolo especial distinto a cualquier otro PID.

Como en una instancia del protocolo IDEAL$_{\mathcal{F}}$ la comunicación entre participantes y funcionalidades (y de este modo entre participantes) es por la cinta de entrada y la cinta de salida de subrutina, el adversario del protocolo ideal $\mathcal{S}$ se ve limitado en su capacidad de espiar o modificar la comunicación. En primera instancia $\mathcal{S}$ no tiene ninguna injerencia en la comunicación, pero en general para que IDEAL$_{\mathcal{F}}$ sea un protocolo "implementable" por un protocolo real es necesario que $\mathcal{F}$ de cierta injerencia a $\mathcal{S}$ en la comunicación. En general se asume que las respuestas de $\mathcal{F}$ a los participantes tontitos pueden ser retardadas tanto como

$\mathcal{S}$ lo desee, incluso por un tiempo infinito. El filtraje de información dependerá del propósito de $\mathcal{F}$.

De forma simular a la UC-emulación definimos la UC-realización de una funcionalidad ideal.

Definición 34 (UC-realización)**.** *Decimos que un protocolo π UC-realiza a la funcionalidad ideal $\mathcal{F}$ si para todo ambiente $\mathcal{Z}$ y para todo adversario real $\mathcal{A}$ existe un adversario ideal $\mathcal{S}$ talque*

$$\text{EXEC}_{\mathcal{Z},\mathcal{A},\pi} \approx \text{EXEC}_{\mathcal{Z},\mathcal{S},\text{IDEAL}_{\mathcal{F}}}$$

Como corolario del teorema 1 tenemos el siguiente resultado.

Corolario 1. *Sean π, ρ protocolos y $\mathcal{F}$ una funcionalidad ideal tales que ρ UC-realiza $\mathcal{F}$ y ρ es un protocolo subroutine respecting. Entonces el protocolo $\pi^{\rho/\text{IDEAL}_{\mathcal{F}}}$ UC-emula a π.*

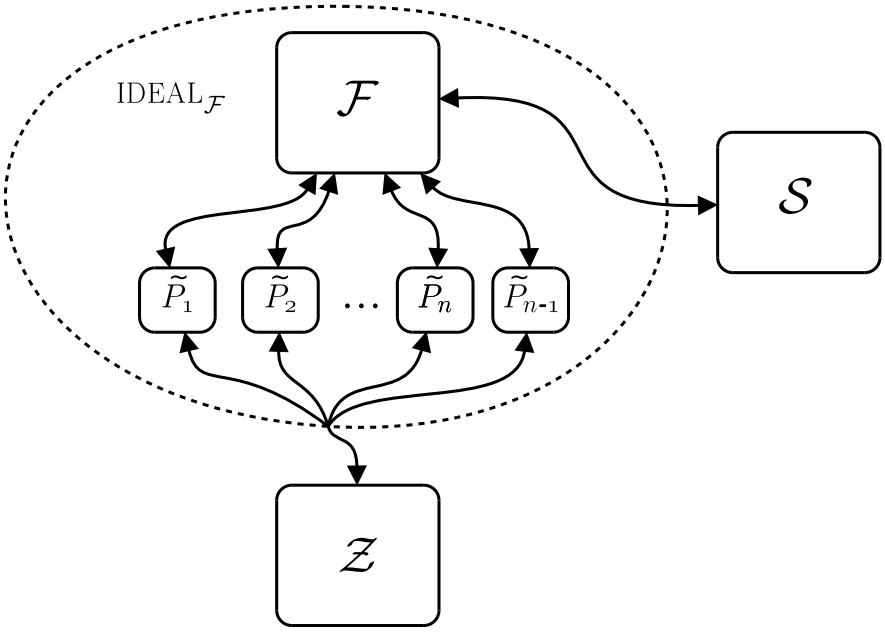

Figura 3.3: Ejecución del protocolo IDEAL$_{\mathcal{F}}$ en UC

El corolario 2 es lo que le da sentido a la "metodología" de UC, pues permite diseñar y probar la seguridad de protocolos criptográficos complejos de forma modularizada, al introducir los *modelos híbridos*.

Definición 35 (Modelo Híbrido)**.** *Decimos que un protocolo π es ejecutado en el modelo $\mathcal{F}$-híbrido si cada participante puede hacer llamados a una instancia local al protocolo π función ideal $\mathcal{F}$.*

De forma similar se puede definir el modelo $\mathcal{F}_1, \ldots, \mathcal{F}_n$-híbrido para funcionalidades ideales $\mathcal{F}_1, \ldots, \mathcal{F}_n$.

En la figura 3.4 se puede apreciar la forma de analizar en UC un protocolo π que llama como subrutina al protocolo ρ. Primero se muestra que el protocolo π UC-emula a la funcionalidad ideal $\mathcal{G}$ en el modelo $\mathcal{F}$-híbrido , y luego se muestra que el protocoloρ UC-emula a la funcionalidad ideal $\mathcal{F}$. Finalmente se puede concluir que π UC-emula $\mathcal{G}$ en el modelo *plano*, esto es, sin acceso a la funcionalidad ideal $\mathcal{F}$.

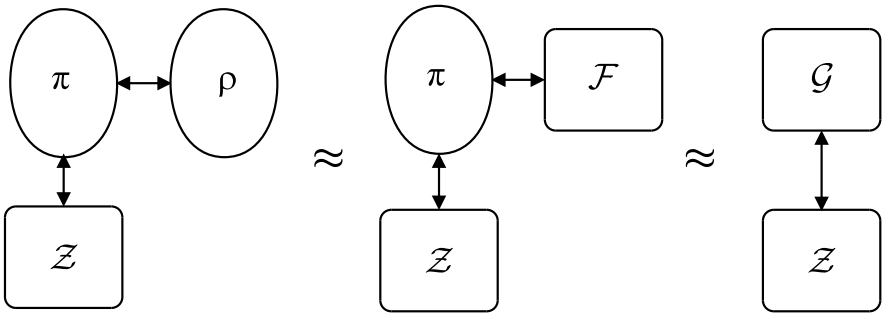

Figura 3.4: Análisis modularizado de un protocolo

3.1.6. Modelos de corrupción

Los modelos de corrupción corresponden a considerar distintos tipos de adversarios. Hay que notar que el teorema de composición se sigue teniendo para cualquier clase de adversarios y por lo tanto para cualquier modelo de corrupción. A continuación listamos dos modelos de corrupción.

Adversarios estáticos

En este caso el conjunto de participantes a los que corrompe el adversario esta fijo a priori, por lo tanto no pueden depender de los distintos valores que se van obteniendo en la ejecución del protocolo. En este caso se puede asumir que el adversario corrompe a los participantes del antes de la ejecución del protocolo.
El conjunto de adversarios estáticos esta estrictamente contenido en el conjunto de adversarios y corresponde a una simplificación irreal de UC. Sin embargo es útil para comenzar a estudiar un protocolo en UC.

Adversarios dinámicos

Al contrario del modelo anterior, en este caso el adversario no se encuentra limitado y las corrupciones pueden irse haciendo conforme avanza el protocolo.

3.2. Generalized Universal Composabillity

3.2.1. Descripción general

Como vimos en la sección 3.1, para poder aplicar el teorema de composición (teorema 1) es necesario que el protocolo sea *subroutine respecting*. Esto es, el protocolo y todas sus subrutinas no deben intercambiar entradas o salidas con otros protocolos externos a la sesión. Así, el protocolo analizado es independiente de cualquier otro protocolo por lo que su comportamiento entrada/salida y los mensajes intercambiados en su ejecución también son independientes. Sin embargo puede resultar poco realista en ciertas ocasiones.

Una escenario donde no es realista asumir que un protocolo es subroutine respecting es donde es necesario asumir la existencia de funcionalidades ideales existenetes a priori, las cuales no son C-realizadas por un protocolo. A esto se le conoce conoce como *Supuestos de inicialización* o *setup assumption*. Esto es necesario cuando se desea desarrollar un protocolo que UC-realice cierta funcionalidad, que se sabe irrealizable sin una setup assumption. Por ejemplo consideremos el caso de *Zero Knowledge Proofs* (o *Pruebas de Nula divulgación* y desde ahora ZKP), es sabido que es imposible UC-realizar ZKP (UC-realizar la funcionalidad ideal $\mathcal{F}_{ZK}^{\mathcal{R}}$ para alguna relación binaria $\mathcal{R}$) sin setup assumptions como por ejemplo el uso de un "string público aleatorio" (CRS) (del inglés *common random string*) cuando la mayoría de los participantes son corruptos [2].

En UC, el CRS es modelado como una funcionalidad ideal $\mathcal{F}_{CRS}$ que hace público un string aleatorio. Ciertamente $\mathcal{F}_{CRS}$ no puede ser UC-realizado por ningún protocolo, pues en ese caso sería posible UC-realizar $\mathcal{F}_{ZK}^{\mathcal{R}}$. Para obtener la funcionalidad ideal $\mathcal{F}_{CRS}$ en el mundo real es necesario que $\mathcal{F}_{CRS}$ sea ejecutado por un participante incorruptible e inimpersonable. Esta suposición es imposible de obtener [4], solo es posible usar técnicas para evitar ataques, y no son infalibles.

Suponiendo la existencia de $\mathcal{F}_{CRS}$, la forma de probar que un protocolo $\pi_{ZK}^{\mathcal{R}}$ que UC-realiza $\mathcal{F}_{ZK}^{\mathcal{R}}$ sería probar que π_{ZK}^{R} en el modelo $\mathcal{F}_{CRS}$-híbrido. Pero esta configuración tiene el inconveniente que se considera que $\mathcal{F}_{CRS}$ es local al protocolo, por lo tanto para cada instancia de $\pi_{ZK}^{\mathcal{R}}$ existiría una instancia de $\mathcal{F}_{CRS}$. Más aún, también debería existir una instancia de $\mathcal{F}_{CRS}$ para cualquier otro protocolo que lo necesite.

La forma correcta de modelar una setup assumption es considerar que es compartida por

[4] Básicamente hay que tener un servidor de CRS que nadie puede *"hackear"*

cada protocolo que hace uso de ella, pero en ese caso los protocolos dejan de ser subroutine respecting. En [16] se muestra que en el caso de un $\mathcal{F}_{CRS}$ compartido usado para UC-realizar ZKP, el protocolo perdería la propiedad de *deniability* que es natural de obtener en ZKP. Más aún, en [21] se muestra que inclusive esto puede llevar a la perdida de la correctitud del protocolo.

3.2.2. Ejecución de un protocolo en GUC

En GUC se define una ejecución de π donde el ambiente no esta limitado a ejecutar solo ITIs con el código del protocolo analizado (a excepción del adversario) y SID fijo, ya que un protocolo en GUC es ejecutado con el sITM $(\mathcal{Z}, C^{\pi,\mathcal{A}}_{GEXEC})$, donde $C^{\pi,\mathcal{A}}_{GEXEC}$ es idéntica a $C^{\pi,\mathcal{A}}_{EXEC}$, salvo que no limita a $\mathcal{Z}$ como en UC pues permite al ambiente ejecutar cualquier protocolo. De este modo un protocolo π ese ejecutado concurrentemente con otros protocolos que comparten estado con π y por lo tanto pueden existir ataques a un protocolo mediante la ejecución paralela de protocolos maliciosamente correlacionados con π, lo que no es posible de modelar en UC.

Denotamos la salida de la ejecución de un protocolo π en GUC por $\text{GEXEC}_{\mathcal{Z},\mathcal{A},\pi} = \text{OUT}_{(\mathcal{Z},C^{\pi,\mathcal{A}}_{GEXEC})}$.

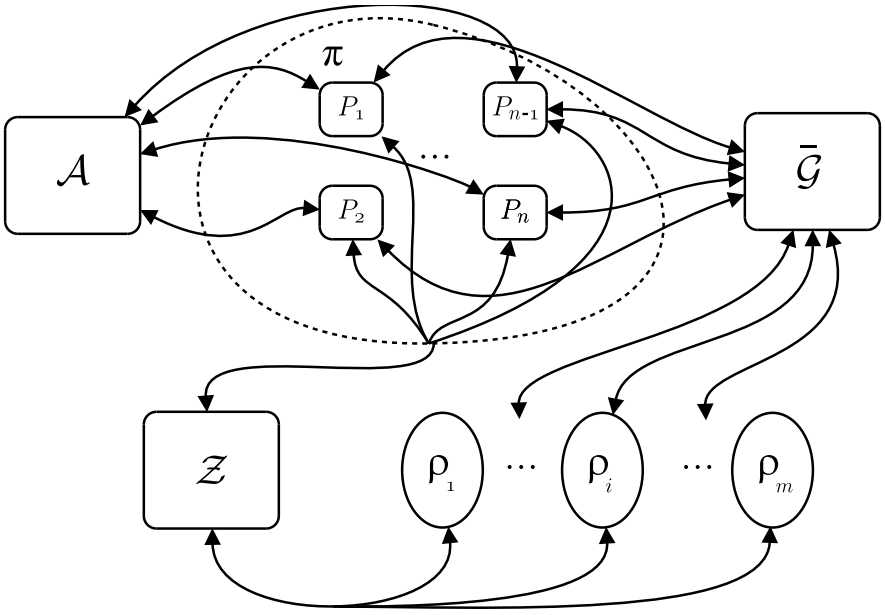

Figura 3.5: Ejecución del protocolo π en GUC

Definimos la GUC-emulación como sigue.

Definición 36 (GUC-emulación)**.** *Decimos que un protocolo π GUC-emula a otro protocolo ρ si para todo ambiente $\mathcal{Z}$ y para todo adversario real $\mathcal{A}$ existe un adversario ideal $\mathcal{S}$ talque:*

$$\text{GEXEC}_{\mathcal{Z},\mathcal{A},\pi} \approx \text{GEXEC}_{\mathcal{Z},\mathcal{A},\rho}$$

3.2.3. Ejecución en EUC

Un ambiente que ejecuta un conjunto arbitrario de protocolos puede resultar complejo de manipular en las pruebas de seguridad, y resulta más cómodo trabajar con un modelo más simple aunque no menos expresivo. En efecto es posible obtener un modelo equivalente a GUC, llamado EUC (del inglés *Externalized-subroutine UC*), en el cual el ambiente solo puede ejecutar una instancia del protocolo más una *funcionalidad compartida*.

Las funcionalidades compartidas permiten al ambiente simular internamente la ejecución concurrente de otros protocolos. Denotadas por la barra superior $\bar{}$, son idénticas que una funcionalidad ideal, salvo que aceptan input de cualquier ITI independiente de su SID. Su cinta de identidad contiene el string $\#||\perp$, donde $\#$ es un SID distinto a todos los otros SID.

En EUC un protocolo π puede no ser subroutine respecting, pero sí debe ser $\bar{\mathcal{G}}$*-subroutine respecting*, donde $\bar{\mathcal{G}}$ es una funcionalidad compartida. Al ser π $\bar{\mathcal{G}}$-subroutine respecting solo puede comunicarse a través de $\bar{\mathcal{G}}$, lo cual no resulta extraño pues en general los protocolos criptográficos comparten estado de forma encapsulable en una funcionalidad (por ejemplo CRS o PKI).

Definición 37 (Protocolo $\bar{\mathcal{G}}$-subroutine respecting)**.** *Decimos que un protocolo es $\bar{\mathcal{G}}$-subroutine respecting si es subroutine respecting sin tomar en cuenta llamadas a una funcionalidad $\bar{\mathcal{G}}$*

A la salida del sITM $(\mathcal{Z}, C^{\pi,\bar{\mathcal{G}},\mathcal{A}}_{EEXEC})$, donde $C^{\pi,\bar{\mathcal{G}},\mathcal{A}}_{EEXEC}$ es una función de control como la de UC pero que adicionalmente permite al ambiente llamar a una funcionalidad compartida $\bar{\mathcal{G}}$, la denotamos por $\text{EEXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi} = \text{OUT}_{(\mathcal{Z},C^{\mathcal{A},\pi,\bar{\mathcal{G}}}_{EEXEC})}$.

Definimos la EUC-emulación como sigue.

Definición 38 (EUC-emulación)**.** *Decimos que un protocolo π EUC-emula con funcionalidad compartida $\bar{\mathcal{G}}$ a otro protocolo ρ si para todo ambiente $\mathcal{Z}$ y para todo adversario real $\mathcal{A}$ existe un adversario ideal $\mathcal{S}$*

$$\text{EEXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{A},\pi} \approx \text{EEXEC}^{\bar{\mathcal{G}}}_{\mathcal{Z},\mathcal{S},\rho}$$

Se puede demostrar que un protocolo que EUC es solo una transformación sintáctica de UC, pues el poder de distinción del ambiente es el mismo en ambos.

Teorema 2 ([7]). *Sean π y ρ dos protocolos $\bar{\mathcal{G}}$-subroutine respecting, π GUC-emula a ρ si y solo si π EUC-emula ρ.*

Intuitivamente podemos notar que todo ambiente $\mathcal{Z}$ que es ejecutado en GUC puede ser simulado por otro ambiente EUC $\mathcal{Z}'$, que simula para $\mathcal{Z}$ todos las instancias de otros protocolos gracias a su acceso a $\bar{\mathcal{G}}$. La otra implicancia es trivial dado que EUC es un caso especial de GUC.

Al igual que en UC existe un teorema de composición, que permite reemplazar protocolos reales por protocolos ideales.

Teorema 3 (Teorema de composición generalizado [7]). *Sean π, ρ, ϕ protocolos tales que ρ GUC-emula a ϕ y ρ y ϕ son protocolos $\bar{\mathcal{G}}$-subroutine respecting. Entonces el protocolo $\pi^{\rho/\phi}$ GUC-emula al protocolo π.*

Corolario 2. *Sean π, ρ protocolos y $\mathcal{F}$ una funcionalidad ideal tales que ρ GUC-realiza $\mathcal{F}$ y ρ es un protocolo $\bar{\mathcal{G}}$-subroutine respecting. Entonces el protocolo $\pi^{\rho/\text{IDEAL}_{\mathcal{F}}}$ GUC-emula a π.*

3.3. Una notación más simple para (G)UC

Revisaremos la notación ocupada en [19] pues simplifica el análisis de protocolos en el caso en que el número de participantes puede estar fijo a priori. En [19] Wikström introduce una nueva máquina a UC, llamada el modelo de comunicación. El modelo de comunicación esta a cargo de todos los aspectos de la comunicación entre ITIs en UC. En el mundo real el modelo de comunicación real $\mathcal{C}$ delega todo el *ruteo* de mensajes al adversario. Por otro lado, en el mundo ideal el modelo de comunicación ideal $\mathcal{C}_{\mathcal{I}}$ solo permite al adversario determinar el retraso (inclusive infinito retraso) con que cada mensaje es entregado de una funcionalidad ideal a un participante.

Esta notación simplifica la definición de qué puede hacer el adversario en una ejecución, en efecto permite definir el mundo real el mundo ideal cambiando solo el modelo de comunicación. La introducción del modelo de comunicación también hace innecesario el uso de SIDs, pues los modelos de comunicación son locales a cada instancia de un protocolo. En las figuras 3.8 3.9 se puede apreciar la confiuración de una ejecución con modelo de comunicación para el mundo real e ideal respectivamente.

Denotamos por ITM el conjunto de todas las ITMs.

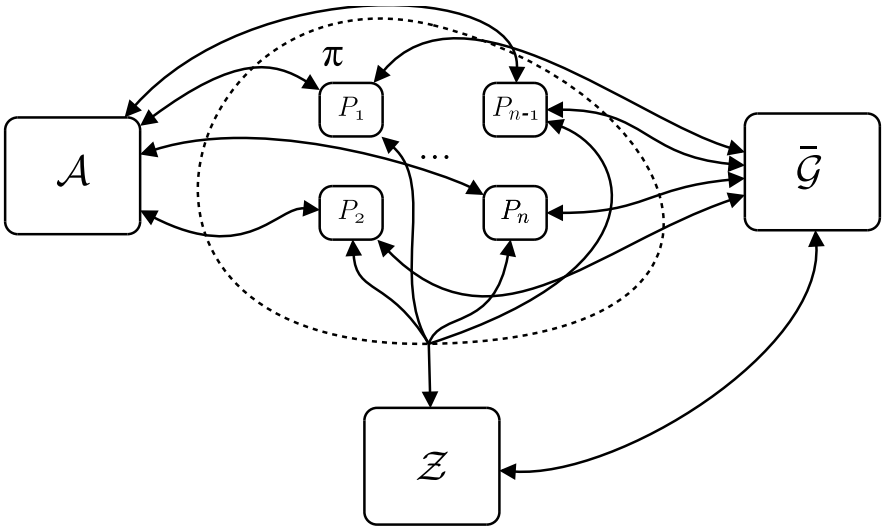

Figura 3.6: Ejecución del protocolo π en EUC

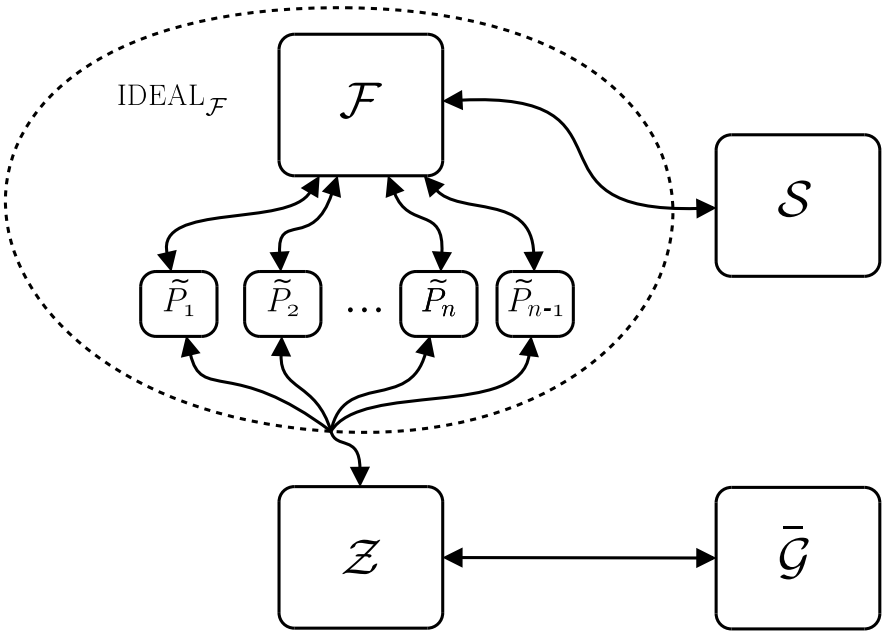

Figura 3.7: Ejecución del protocolo IDEAL$_{\mathcal{F}}$ en EUC

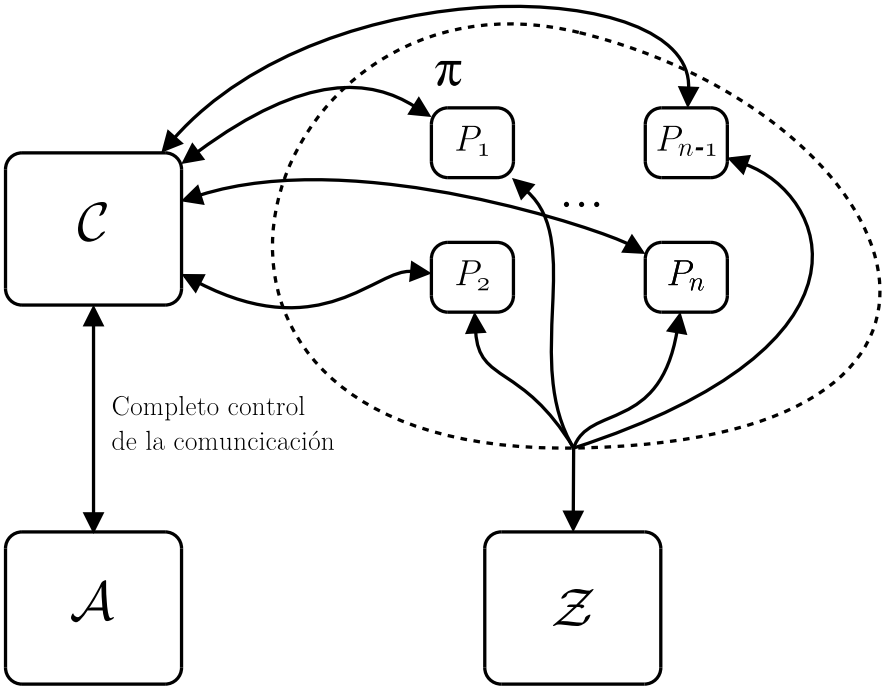

Figura 3.8: Ejecución del protocolo π con modelo de comunicación real $\mathcal{C}$.

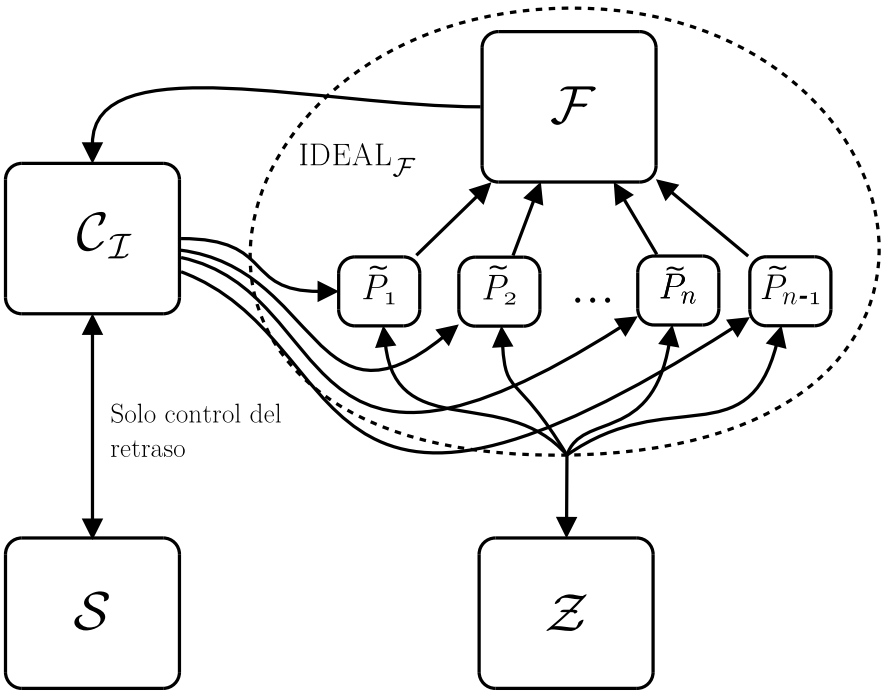

Figura 3.9: Ejecución del protocolo IDEAL$_{\mathcal{F}}$ con modelo de comunicación ideal $\mathcal{C}_{\mathcal{I}}$.

Definición 39 (grafo de ITMs)**.** *Un grafo de ITMs en un conjunto vértices* $V = \{P_1, \ldots, P_t\} \subset$ *ITM con un conjunto de aristas* E *tales que* (V, E) *es un grafo conexo, y ningún* P_i *se puede comunicar con una máquina fuera de* V*. Si* $(P_i, P_j) \in E$ *entonces se dice que* P_i *tiene un link con* P_j *y viceversa* [5]*. Sea ITMG el conjunto de todos los grafos de ITMs.*
Durante la ejecución de un grafo de ITMs, a lo más un participante se encuentra activo. Un participante activo puede desactivarse y activar a alguno de sus vecinos entregándole cierta entrada x*, o puede detenerse en cuyo caso la ejecución de grafo de ITMs se detiene.*

El modelo de comunicación real modela una red con comunicación asíncrona, en donde el adversario puede leer, borrar, modificar e insertar cualquier mensaje de su elección.

Definición 40. *Un modelo de comunicación real* $\mathcal{C}$ *es una ITM con un link* l_{P_i} *a* P_i *para* $i = 1, \ldots, k$*, y un link* $l_{\mathcal{A}}$ *al adversario real* $\mathcal{A}$*. Su código se define como sigue.*

1. *Si* m *es leído en* l_s *donde* $s \in \{P_1, \ldots, P_k\}$*, entonces* (s, m) *es escrito en* $l_{\mathcal{A}}$ *y mathcal*A *es activado.*

2. *Si* (r, m) *es leído en* $l_{\mathcal{A}}$*, donde* $r \in \{P_1, \ldots, P_k\}$*,* m *es escrito en* l_r *y* P_r *es activado.*

Por simplicidad omitimos llamar explícitamente a modelo de comunicación real. Cuando en un protocolo del mundo real escribimos "P_i envía m a P_j" nos referimos a "P_i envía (P_j, m) a $\mathcal{C}$".

El modelo de comunicación ideal captura el hecho de que el adversario ideal puede decidir si y cuando enviar un mensaje desde una funcionalidad ideal a un participante, pero no puede leer ni modificar los contenidos de la comunicación entre participantes y la funcionalidad ideal.

Definición 41. *Un modelo de comunicación ideal* $\mathcal{C}_{\mathcal{I}}$ *es una ITM con un link* l_{P_i} *a* P_i *para* $i = 1, \ldots, k$*, y links* $l_{\mathcal{F}}$ *y* $l_{\mathcal{S}}$ *a una funcionalidad ideal* $\mathcal{F}$ *y a un adversario ideal* $\mathcal{S}$ *respectivamente. Su código se define como sigue.*

1. *Si un mensaje* m *es leído en* l_s*, donde* $s \in \{P_1, \ldots, P_k\}$*, entonces* (s, m) *es escrito en* $l_{\mathcal{F}}$ *y* $\mathcal{F}$ *es activada.*

2. *Si un mensaje* (s, m) *escrito en* $l_{\mathcal{F}}$ *es retornado inalterado,* m *es escrito en* l_s*. Si no, cualquier string leído desde* $l_{\mathcal{F}}$ *es interpretado como una lista* $((r_1, m_1), \ldots, (r_t, m_t))$*, donde* $r_i \in \{\mathcal{S}, P_1, \ldots, P_k\}$*. Por cada* m_i *un string aleatorio* $\tau_i \in \{0,1\}^n$ *es escogido, y* (r_i, m_i) *es guardado en el registro etiquetado con* (τ_i)*. Luego* $((r_1, |m_1|, \tau_1), \ldots, (r_t, |m_t|, \tau_t))$ *es escrito en* $l_{\mathcal{S}}$ *y* $\mathcal{S}$ *es activado.*

3. *Todo string leído desde* $l_{\mathcal{S}}$ *es interpretado como el par* (b, τ)*, donde* $b \in \{0,1\}$ *y* τ *es un string arbitrario. Si* $b = 1$ *y* (r_i, m_i) *esta guardado en el registro etiquetado con* τ*,* m_i *es escrito en* l_{r_i} *y* r_i *es activado. Si* $b = 0$ $(\mathcal{S}, \tau)$ *es escrito en* $l_{\mathcal{F}}$ *y* $\mathcal{F}$ *es activada.*

[5] Las ITMs de esta formulación tienen tantas cintas de comunicación como vecinos.

El modelo ideal es equivalente a una ejecución del protocolo $\text{IDEAL}_{\mathcal{F}}$

Definición 42. *El modelo ideal es definido como la función* $\mathcal{I} : ITM^2 \times I\tilde{T}M^* \to ITMG$, *donde* $\mathcal{I} : (\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k) \mapsto (V, E)$ *viene dado por:*

$$V = \{\mathcal{C}_{\mathcal{I}}, \mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k\}$$

$$E = \{(\mathcal{S}, \mathcal{C}_{\mathcal{I}}), (\mathcal{C}_{\mathcal{I}}, \mathcal{F})\} \cup \bigcup_{i=1}^{k} \{(\tilde{P}_i, \mathcal{C}_{\mathcal{I}})\}$$

Si $\tilde{\pi} = (\tilde{P}_1, \ldots, \tilde{P}_k)$, escribimos $\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}})$ en vez de $\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$ para facilitar la notación.

El modelo real corresponde a la ejecución de un protocolo "real" en UC, es decir donde los participantes se comunican a través de la cinta de comunicación entrante.

Definición 43. *El modelo real es definido como la función*$\mathcal{R} : ITM^* \to ITMG$, *donde* $\mathcal{R} : (\mathcal{A}, P_1, \ldots, P_k) \mapsto (V, E)$ *viene dado:*

$$V = \{\mathcal{C}, \mathcal{A}, P_1, \ldots, P_k\}$$

$$E = \{(\mathcal{A}, \mathcal{C})\} \cup \bigcup_{i=1}^{k} \{(P_i, \mathcal{C})\}$$

Sea $(V, E) = \mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$. Entonces $\mathcal{Z}(\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k))$ para denotar al grafo de ITMs (V', E') definido por $V' = V \cup \{\mathcal{Z}\}$, y $E' = E \cup \{(\mathcal{Z}, \mathcal{S})\} \bigcup_{i=1}^{k} \{(\mathcal{Z}, \tilde{P}_i)\}$. Usamos la misma notación para el modelo real.

El modelo híbrido se define como sigue

Definición 44. *Sean* $(V, E) = \mathcal{R}(\mathcal{A}, \pi)$, $\pi = (P_1, \ldots, P_k)$. *Sean* $(V_j, E_j) = \mathcal{I}(\mathcal{S}_j, \{\tilde{P}_i\}_j^{\mathcal{F}_j})$, $\{\tilde{P}_i\}_j = (\tilde{P}_{j,1}, \ldots, \tilde{P}_{j,k})$ *para* $j = 1, \ldots, t$, *y* $(V_j, E_j) = \mathcal{R}(\mathcal{S}_j, \pi_j)$, $\pi_j = (P_{j,1}, \ldots, P_{j,k})$ *for* $j = t+1, \ldots, s$.
Denotamos por $\mathcal{H}(\mathcal{A}^{\mathcal{S}_1,\ldots,\mathcal{S}_t}, \pi^{\{\tilde{P}_i\}_1^{\mathcal{F}_1},\ldots,\{\tilde{P}_i\}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s})$ *al modelo híbrido que se define como el grafo de ITMs* (V', E'), *donde*

$$V' = V \cup \bigcup_{j=1}^{t} V_j, \text{ y}$$

$$E' = E \cup \bigcup_{j=1}^{t} E_j \cup \bigcup_{i=1}^{k} \left(\{(\mathcal{S}_i, \mathcal{A})\} \cup \bigcup_{j=1}^{t} \{(P_i, \tilde{P}_{j,i})\} \right)$$

De la misma forma que antes denotamos por $\mathcal{Z}(\mathcal{H}(\mathcal{A}^{\mathcal{S}_1,\ldots,\mathcal{S}_t}, \pi^{\{\tilde{P}_i\}_1^{\mathcal{F}_1},\ldots,\{\tilde{P}_i\}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s}))$ *al grafo de ITMs* (V'', E'') *definido por* $V'' = V \cup \{\mathcal{Z}\}$, *y* $E'' = E' \cup \{\mathcal{Z}, \mathcal{A}\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$.

La UC-realización entonces queda como sigue.

Definición 45. *Un protocolo π UC-realiza una funcionalidad ideal $\mathcal{F}$ si para todo ambiente $\mathcal{Z}$ y para todo adversario real $\mathcal{A}$ existe un adversario ideal $\mathcal{S}$ talque*

$$\mathcal{Z}(\mathcal{H}(\mathcal{A}, \pi)) \approx \mathcal{Z}(\mathcal{I}(\mathcal{S}, \mathcal{F}))$$

Usando esta notación, el teorema de composición se escribe como sigue.

Teorema 4. *Supongamos que $\pi^{\{\tilde{P}_i\}_1^{\mathcal{F}_1},\ldots,\{\tilde{P}_i\}_t^{\mathcal{F}_t}}$ es un protocolo que UC-realiza a la funcionalidad ideal $\tilde{\pi}^{\mathcal{F}}$. Sea ρ^{π} un protocolo subroutine respecting. Entonces el protocolo $\rho^{\pi/\mathcal{F}}$ en el modelo $\mathcal{F}$-híbrido UC-emula al protocolo ρ^{π}*

Para obtener GUC hacemos la siguiente modificación. Consideremos el grafo de ITMs (V, E) definido por $\mathcal{H}(\mathcal{A}^{\mathcal{S}_1,\ldots,\mathcal{S}_t}, \pi^{\{\tilde{P}_i\}_1^{\mathcal{F}_1},\ldots,\{\tilde{P}_i\}_r^{\mathcal{F}_r},\{\tilde{P}_i\}_{r+1}^{\bar{\mathcal{G}}_{r+1}},\ldots,\{\tilde{P}_i\}_t^{\bar{\mathcal{G}}_t},\pi_{t+1},\ldots,\pi_s}))$, escribimos $\mathcal{Z}(V, E)$ para denotar el grafo de ITMs (V'', E'') definido por $V'' = V \cup \{\mathcal{Z}\}$, y $E'' = E' \cup \{\mathcal{Z}, \mathcal{A}\} \cup \bigcup_{i=1}^{k} \bigcup_{j=r+1}^{t} \{(\mathcal{Z}, \tilde{P}_{i,j})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$. Nótese que los links $\bigcup_{i=1}^{k} \bigcup_{j=r+1}^{t} \{(\mathcal{Z}, \tilde{P}_{i,j})\}$ son todo lo necesario para dar acceso al ambiente a las funcionalidades y compartidas y obtener GUC.

Capítulo 4

Desmentibilidad

La Desmentibilidad puede tener distintas acepciones, por ejemplo Encriptación desmentible [1]. Aquí nos referimos a la noción de Desmentibilidad asociada a la Autentificación desmentible.

La Autentificación desmentible fue introducida por Dwork, Nahor y Sahai en [12]. Variadas modificaciones, generalizaciones e implementaciones han sido propuestas posteriormente, como por ejemplo en [17]. Aquí consideraremos y adaptaremos la definición hecha por Dodis, Katz, Smith y Walfish en [11], dado que esa definición es la que aplica a una configuración concurrente y distribuida como la necesaria en este trabajo.

Intuitivamente decimos que un protocolo es desmentible si nadie puede probar a otro que una sesión del protocolo, es decir un grupo específico de participantes con identidades públicas definidas, se esta llevando a cabo o alguna vez se llevó a cabo. En [11] se muestra que para el caso de la autentificación la desmentibilidad se puede obtener considerando un juez que en forma *online* debe decidir con quién esta hablando: un informante que esta observando una sesión real del protocolo de autentificación, o un desinformante que no tiene acceso a la sesión real del protocolo pero aún así quiere convencer al juez que la sesión se esta llevando a cabo. El protocolo se dirá entonces que es un protocolo de autentificación desmentible online si para todo juez y para todo informante existe un desinformante tal que el juez no puede distinguir si habla con el informante o el desinformante. En la versión completa de [11] se demuestra se demuestra que esta noción es equivalente a GUC-realizar la funcionalidad ideal $\mathcal{F}_{AUTH}$. Dodis y compañía señalan que en GUC un protocolo que realiza una funcionalidad ideal $\mathcal{F}$ es tan desmentible como $\mathcal{F}$. La funcionalidad ideal $\mathcal{F}_{AUTH}$ es "completamente simulable", lo que significa que el protocolo puede ser simulado completamente sin la participación de

ningún participante de la sesión, luego la funcionalidad $\mathcal{F}_{AUTH}$ es desmentible.
De forma similar que en [11], pero sin restringirnos a protocolos de autentificación, definiremos la noción de desmentibilidad en línea o *online*.

4.1. Desmentibilidad online

Consideraremos dos *mundos*: el mundo real, donde el informante tiene acceso directo a una sesión del protocolo analizado; y el mundo simulado, donde el desinformante no tiene acceso al protocolo. Nos restringiremos a funcionalidades ideales, pues la desmentibilidad es una propiedad a ser chequeada en la funcionalidad ideal.

Definición 46 (Mundo real)**.** *Sea $\mathcal{F}$ una funcionalidad ideal que se ejecuta en el modo $\bar{\mathcal{G}}$-híbrido con participantes $\tilde{P}_1, \ldots, \tilde{P}_n$, sea $\mathcal{D}$ el adversario dummy, sea $\mathfrak{I}$ el informante, $\mathcal{J}$ el juez y sea $(V_{\mathcal{F}}, E_{\mathcal{F}}) = \mathfrak{I}(\mathcal{H}(\mathcal{F}, \mathcal{D}, \tilde{P}_1, \ldots, \tilde{P}_n, \{\tilde{P}_i\}^{\bar{\mathcal{G}}}))$. Definimos el mundo real como el grafo de ITMs $\mathfrak{R} = (V, E)$ donde:*

$$V = \{\mathcal{J}, \bar{\mathcal{G}}\} \cup V_{\mathcal{F}}$$

$$E = \{(\mathcal{J}, \mathfrak{I})\} \cup E_{\mathcal{F}}$$

Denotamos por $\text{Real}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{I}}$ *a la variable aleatoria que describe la salida de $\mathcal{J}$ al ejecutarse el grafo de ITMs $\mathfrak{R}$*

Definición 47 (Mundo simulado)**.** *Sea $\mathcal{F}$ una funcionalidad ideal que se ejecuta en el modo $\bar{\mathcal{G}}$-híbrido con participantes $\tilde{P}_1, \ldots, \tilde{P}_n$, sea $\mathcal{D}$ el adversario dummy, sea $\mathfrak{D}$ el desinformante, $\mathcal{J}$ el juez. Definimos el mundo simulado como el grafo de ITMs $\mathfrak{S} = (V, E)$ donde:*

$$V = \{\mathcal{J}, \bar{\mathcal{G}}, \mathfrak{D}\}$$

$$E = \{(\mathcal{J}, \mathfrak{D}), (\mathcal{J}, \bar{\mathcal{G}}), (\mathfrak{D}, \bar{\mathcal{G}})\}$$

Denotamos por $\text{Sim}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{D}}$ *a la variable aleatoria que describe la salida de $\mathcal{J}$ al ejecutarse el grafo de ITMs $\mathfrak{S}$*

Definimos la desmentibilidad como la incapacidad del juez para distinguir entre una ejecución real informada por el informante de la funcionalidad de una ejecución simulada por el desinformante.

Definición 48 (Desmentibilidad)**.** *Decimos que una funcionalidad $\mathcal{F}$ es desmentible si para todo juez $\mathcal{J}$ y todo informante $\mathfrak{I}$ existe un desinformante $\mathfrak{D}$ talque:*

$$\text{Real}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{I}} \approx \text{Sim}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{D}}$$

Notamos que el experimento $\text{Real}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{I}}$ es una transformación sintáctica de la ejecución en UC de la funcionalidad ideal $\mathcal{F}$, como lo demuestra el siguiente teorema.

Teorema 5. *Para toda funcionalidad $\mathcal{F}$ ejecutada en el modelo $\bar{\mathcal{G}}$-híbrido se tiene que para todo juez $\mathcal{J}$ y todo informante $\mathfrak{I}$ existe un ambiente $\mathcal{Z}$ talque*

$$\text{Real}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{I}} \approx \mathcal{Z}(\mathcal{H}(\mathcal{F}, \mathcal{D}, \tilde{P}_1, \ldots, \tilde{P}_n, \{\tilde{P}_i\}^{\bar{\mathcal{G}}}))$$

y también para todo ambiente $\mathcal{Z}$ existe un juez $\mathcal{J}$ y un informante $\mathfrak{I}$ tales que

$$\text{Real}^{Den}_{\mathcal{F},\mathcal{J},\mathfrak{I}} \approx \mathcal{Z}(\mathcal{H}(\mathcal{F}, \mathcal{D}, \tilde{P}_1, \ldots, \tilde{P}_n, \{\tilde{P}_i\}^{\bar{\mathcal{G}}}))$$

Demostración. (Teorema 5)
En efecto, cualquier combinación juez-informante puede ser simulada por un ambiente $\mathcal{Z}$, lo que prueba la segunda afirmación de teorema.
Si consideramos un informante $\mathfrak{I}$ que solo comunica al juez $\mathcal{J}$ con los participantes y el adversario dummy $\mathcal{D}$, el juez $\mathcal{J}$ es capaz de simular internamente a cualquier ambiente $\mathcal{Z}$ pues $\mathfrak{I}$ le provee una interfaz con vista idéntica a la vista de $\mathcal{Z}$. Lo que nos permite probar la primera afirmación del teorema. □

Como corolario del teorema 5 tenemos el siguiente resultado

Corolario 3. *Sea π protocolo que GUC-emula a una funcionalidad ideal $\mathcal{F}$ desmentible. Entonces π es desmentible.*

Demostración. La demostración es directa dado que el experimento real para π es identico a ejecutarlo en GUC. Como π GUC-emula a $\mathcal{F}$ existe un simulador que simula a π con $\mathcal{F}$, el cual puede ser usado para simular π con el desinformate que existe para $\mathcal{F}$. □

Capítulo 5

Anonimato

5.1. Canales Anónimos

Los *Canales Anónimos* permiten a los usuarios intercambiar mensajes sin revelar sus identidades. Las aplicaciones son variadas y van desde votaciones, donde la identidad de los votantes debe ser anónima, hasta bases de datos con información confidencial por su alta sensibilidad, como bases de datos medicas. Seguiremos la definición de canal anónimo de [14], pues es lo suficientemente general.

Consideremos la situación en que existen n participantes, cuyas identidades son $P_1, \ldots, P_n$, que ejecutan una instancia de un protocolo π. El conjunto de mensajes intercambiados entre $P_1, \ldots, P_n$ en la ejecución de π se puede representar por una matriz M. Cada elemento de la matriz, m_{ij}, es el multiconjunto de mensajes que envía algún participante P_i a otro participante P_j. Intuitivamente la comunicación con el protocolo π será anónima si ningún observador es capaz de obtener más que "cierta" información de M. La información que el observador sí puede obtener es una variable del modelo y lleva a considerar distintos tipos de anonimato.

Consideraremos la ejecución de un protocolo en una configuración similar a UC, en que el adversario es pasivo (no inyecta ni altera mensajes) y adicionalmente el ambiente recibe como parámetro la matriz $M \in \mathcal{M}_{n\times n}(\mathcal{P}(\{0,1\}^{p(k)}))$, con k el parámetro de seguridad y p un polinomio. El ambiente pasa como parámetro a P_i $i \in \{1, \ldots, n\}$ los mensajes $\{(m_{i,j}, P_j)\}_{j=1}^{n}$, indicando que el mensaje $m_{i,j}$ es enviado por P_i a P_j. Nos centraremos en protocolos conocidos como *protocolos de transmisión de mensajes.*

Definición 49 (Protocolo de trasmisión de mensajes)**.** *Decimos que un protocolo π es un protocolo de transmisión de mensajes si al ser ejecutado con entrada M para la salida de cada participante P_i $i \in \{1, \ldots, n\}$ es el multiconjunto $\uplus_{j=1}^{n}\{m_{i,j}\}$.*

Definimos el experimento $\text{Exp}^{\mathcal{R}-anon}_{\pi,\mathcal{A}}$, donde $\mathcal{R} \subseteq \mathcal{M}_{n\times n}(\mathcal{P}(\{0,1\}^{p(k)}))^2$, en la figura 5.1.

Definición 50. *Decimos que un protocolo de transmisión de mensajes π es $\mathcal{R}$-anónimo si para todo adversario PPT $\mathcal{A}$ existe una función despreciable η talque*

$$2 \cdot \Pr[\text{Exp}^{\mathcal{R}-anon}_{\pi,\mathcal{A}}(k) = 1] - 1 \leq \eta(k)$$

Notemos que el adversario $\mathcal{A}$ solo puede elegir matrices M_1, M_2 tales que $(M_1, M_2) \in \mathcal{R}$, de este modo $\mathcal{R}$ se puede usar para determinar la filtración de información permitida al protocolo y con ello el tipo de anonimato. En efecto, sea f una función $f : \mathcal{M}_{n\times n}(\mathcal{P}(\{0,1\}^{p(k)})) \to I$, con I algún conjunto, que llamaremos *función de filtración de información*. Si por ejemplo $f(M) = \sum_{i,j} |m_{i,j}|$, el filtraje de información correspondería al tamaño total de los mensajes intercambiados. Si π permite computar $f(M)$ al adversario, entonces trivialmente este puede distinguir entre la ejecución de π con una matriz M_1 y la ejecución de π con otra matriz M_2 si elije M_1 y M_2 tales que $f(M_1) \neq f(M_2)$. Si definimos $\mathcal{R} = \{(M_1, M_2)|f(M_1) = f(M_2)\}$ entonces $\mathcal{A}$ ya no podrá efectuar este ataque, y cualquier distinción entre la ejecución de π con M_1 y M_2 dependerá de otro "filtraje" de información.

En [14] se definen tres funciones de filtración de información, estas son $f_\cup, f_\Sigma, f_\#$ y se definen como sigue:

$$f_\cup(M) = \left(\biguplus_{j=1}^{n} \{m_{1,j}\}, \ldots, \biguplus_{j=1}^{n} \{m_{n,j}\} \right)$$

$$f_\Sigma(M) = \left(\sum_{j=1}^{n} |m_{1,j}|, \ldots, \sum_{j=1}^{n} |m_{n,j}| \right)$$

$$f_\#(M) = \sum_{i=1}^{n} \sum_{j=1}^{n} |m_{i,j}|$$

$f_\cup(M)$ corresponde al vector donde la componente i-ésima es el multiconjunto de mensajes enviados por P_i, $f_\Sigma(M)$ corresponde al vector donde la componente i-ésima es el tamaño en bits del total de mensajes enviados por P_i y $f_\#(M)$ corresponde al tamaño total en bits de mensajes enviados en el protocolo. Adicionalmente se define $f^T(M) = f(M^T)$ donde M^T denota la matriz M transpuesta, y de ese modo $f^T_\cup(M)$ y $f^T_\Sigma(M)$ corresponden a vectores donde el elemento i-ésimo es un valor calculado sobre los mensajes recibidos por P_i.

Adicionalmente en este trabajo definimos una nueva función de filtración de información, $f_{\cup\cup}$ que viene dada por:

$$f_{\cup\cup}(M) = \biguplus_{i=1}^{n} \biguplus_{i=1}^{n} m_{i,j}$$

Tipo de anonimato	Relación
Invinculabilidad débil	$\mathcal{R}_{\cup} \cap \mathcal{R}_{\cup}^{T}$
Invinculabilidad del enviador	$\mathcal{R}_{\Sigma} \cap \mathcal{R}_{\cup}^{T}$
Invinculabilidad del receptor	$\mathcal{R}_{\cup} \cap \mathcal{R}_{\Sigma}^{T}$
Invinculabilidad	$\mathcal{R}_{\Sigma} \cap \mathcal{R}_{\Sigma}^{T}$
Anonimato del enviador	$\mathcal{R}_{\cup}$
Anonimato del receptor	$\mathcal{R}_{\cup}^{T}$
Anonimato $f_{\cup\cup}$	$\mathcal{R}_{\cup\cup}$
Anonimato del enviador fuerte	$\mathcal{R}_{\Sigma}$
Anonimato del receptor fuerte	$\mathcal{R}_{\Sigma}^{T}$
Anonimato del enviador y receptor	$\mathcal{R}_{\#}$
Inobservabilidad	$\mathcal{M}_{n\times n}(\mathcal{P}(\{0,1\}^{p(k)}))$

Cuadro 5.1: Tipos de anonimato

El experimento $\mathrm{Exp}_{\pi,\mathcal{A}}^{\mathcal{R}-anon}(k)$ procede como sigue:

1. Escoger $b \in_R \{0,1\}$ y ejecutar $(M_0, M_1) \leftarrow \mathcal{A}(k)$
2. Si $(M_0, M_1) \notin \mathcal{R}$ retornar 0
3. Ejecutar π con la matriz M_b y adversario $\mathcal{A}$ hasta que $\mathcal{A}$ retorne un bit $b_{\mathcal{A}}$
4. Si $b = b_{\mathcal{A}}$ retornar 1, de lo contrario retornar 0.

Figura 5.1: El experimento $\mathrm{Exp}_{\pi,\mathcal{A}}^{\mathcal{R}-anon}$

Cada función da lugar a una relación $\mathcal{R}_f = \{(M_1, M_2 | f(M_1) = f(M_2))\}$, adicionalmente denotamos por $\mathcal{R}_\star$ a $\mathcal{R}_{f_\star}$. En [14] se definen diez relaciones dando lugar a diez tipos de anonimato, los que se ilustran en la tabla y adicionalmente, con la función $f_{\cup\cup}$, definimos un nuevo tipo de anonimato que llamamos *Anonimato* $f_{\cup\cup}$.

Notemos que la nueva noción de anonimato es más fuerte que las nociones de anonimato del receptor/enviador. En efecto, solo hay que notar que para cualquier par de matrices $f_{\cup}(M_1) = f_{\cup}(M_2)$ implica que $f_{\cup\cup}(M_1) = f_{\cup\cup}(M_2)$, por lo tanto $\mathcal{R}_{\cup} \subseteq \mathcal{R}_{\cup\cup}$ y $\mathcal{R}_{\cup}^{T} \subseteq \mathcal{R}_{\cup\cup}$.

5.2. Primitivas para Canales Anónimos

Variadas primitivas útiles para implementar canales anónimos han sido propuestas en la literatura. A continuación revisamos cuatro primitivas para canales anónimos.

5.2.1. Redes de Mezcla o *Mix-nets*

El estudio moderno protocolos de canales anónimos comenzó en [8] con las Redes de Mezcla o *mix-nets*. En una mix-net el vector v_0 formado por los mensajes encriptados de todos los participantes es enviado a través de una serie de *mixers* (en español mezcladores). Cada mixer M_i realiza una operación en el vector de textos cifrados v_{i-1}, obteniendo nuevos textos cifrados v'_{i-1}, y envía una permutación aleatoria de los nuevos textos cifrados $v_i = \pi(v'_{i-1})$ al siguiente mixer. La operación que cada mixer realiza debe ser tal que permite ocultar la permutación que efectuó el mixer. Finalmente el último mixer publica una permutación del vector de mensajes $\pi(m_1, \ldots, m_n)$ de los participantes. En la figura 5.2 se puede ver un diagrama de una mix-net.

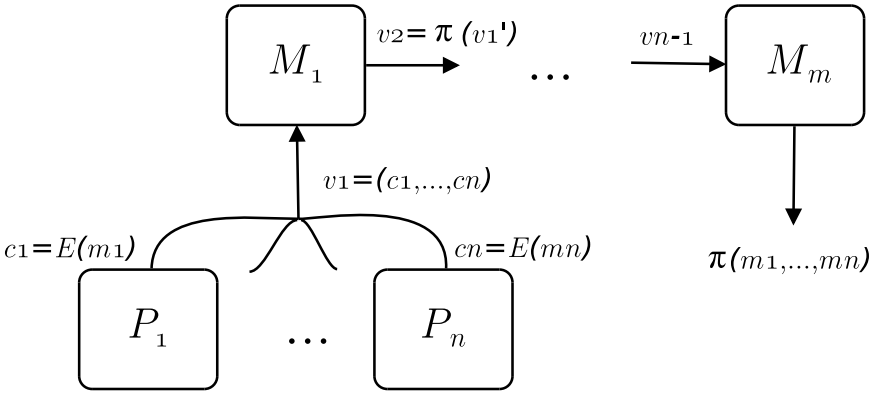

Figura 5.2: Diagrama de una mix-net

Las mix-nets se pueden clasificar en dos, según el tipo de operación que cada mixer realiza en los textos cifrados: mix-nets reencriptantes y mix-nets desecncriptantes.

5.2.2. Mix-nets reencriptantes

En las mix-nets reencriptantes los Mixers permutan aleatoriamente cada vector de mensajes y re-encriptan los textos planos. Consideremos un esquema de encriptación asimétrico $\mathcal{E} = (K, E, D)$ y fijemos una clave pública pk; enotonces una re-encriptación corresponde a aplicar una función $\rho_{pk} : C \times R \to C$, con C el espacio de textos cifrados, tal que para todo m, r, r', c tales que $c = E_{pk}(m, r)$ y $r \neq r'$:

$$D_{sk}(\rho_{pk}(c, r')) = m \quad \text{y además } \rho_{pk}(c, r') \neq c$$

En otras palabras la función ρ cambia la aleatoriedad de un texto cifrado. Al cambiar la aleatoriedad y si $\mathcal{E}$ es un esquema de encriptación seguro (IND-CPA) no es posible para ningún adversario correlacionar las entradas y salidas de un mixer.

Un ejemplo de esquema de encriptación en el cual es posible reencriptar los textos cifrados es ElGamal. Se puede implementar un algoritmo de reencriptación para ElGamal usando que $E_y(m,r) \cdot E_y(m',r') = E_y(m \cdot m', r + r')$ (es homomórfico). En efecto:

$$E_y(m,r) \cdot E_y(m',r') = (g^r \cdot g^{r'}, y^r \cdot m \cdot y^{r'} \cdot m') = (g^{r+r'}, y^{r+r'} m \cdot m')$$

Por lo tanto

$$E_y(m,r) \cdot E_y(m',r') = E_y(m \cdot m', r + r')$$

Usando lo anterior es posible construir ρ tomando $m' = 1$, de este modo

$$\rho_y(E(m,r), r') = E_y(m,r) \cdot E_y(1,r') = E_y(m, r + r')$$

Una desventaja de un esquema de encriptación que permite re-encriptación como ElGamal es que para ocupar ρ es necesario conocer la clave pública de quien encriptó del emisor del mensaje. En el contexto de una mix-net esto significaría que cada mixer debería primero obtener la clave de cada enviador, lo cual puede resultar impráctico. En [13] proponen un esquema de encriptación en el cual no es necesario conocer las claves públicas de los enviadores. Dicha propiedad la llaman Universal Re-encryption y para obtenerla en un esquema basado en ElGamal básicamente adjuntan en el texto cifrado lo necesario para homomorficamente cambiar la aleatoriedad del texto cifrado. El esquema propuesto en [13] esta compuesto por los algoritmos (K, E, D, ρ) donde

$$K(k) = (x, y), \quad x \in_R G_q, y = g^x$$

$$E_y(m, (k_0, k_1)) = [(g^{k_0}, y^{k_0} \cdot m); (g^{k_1}, y^{k_1})]$$

$$D_x([(u,v);(\alpha,\beta)]) = \frac{v}{u^x}$$

Notemos que la segunda componente de $E_y(m,(k_0,k_1))$ (esto es g^{k_1} y y^{k_1}) es la encriptación de 1 bajo la clave pública y y aleatoriedad k_1, entonces ρ sigue esencialmente igual, salvo que no es necesario conocer la clave pública. En efecto

$$\rho([(u,v);(\alpha,\beta)],(k_0,k_1)) = [(u\cdot\alpha^{k_0}, v\cdot\beta^{k_0});(\alpha^{k_1},\beta^{k_1})]$$

5.2.3. Mixnet con desencriptación

En estos protocolos los Mixers desencriptan parcialmente los textos cifrados y los permutan aleatoriamente, de modo que finalmente se obtiene una permutación de los textos planos originales.

Una forma de implementar una mixnet con reencriptación es usando un esquema de encriptación semánticamente seguro $\mathcal{E} = (K, E, D^{dist}, D)$. Para enviar un texto plano m_i un enviador P_i encriptará m_i bajo una clave pública pk, cuya correspondiente clave privada es una función f de las claves privadas de los Mixers $\{sk_i\}_{i=1}^N$. Cada mixer M_k toma un vector de textos cifrados $(v_1, v_2 \ldots, v_M)$ y desencripta parcialmente cada texto cifrado con su clave secreta sk_k, de modo talque si nos restringimos a vectores de textos cifrados de tamaño 1 el aporte total de los mixers es:

$$D_{sk_1}(D_{sk_2}(\ldots D_{sk_N}(E_{pk}(m,r)\ldots)) = D_{f(sk_1,sk_2,\ldots,sk_N)}(m) = m$$

Adicionalmente cada mixer M_i escoge una permutación al azar $\pi \in_R \Pi_M$, con Π_M el conjunto de todas las permutaciones de $\{1,\ldots,M\}$, y la aplica al vector de textos cifrados parcialmente desencriptados para retornar otro vector $(D_{sk_i}(v_{\pi^{-1}(1)}), D_{sk_i}(v_{\pi^{-1}(2)}), \ldots, D_{sk_i}(v_{\pi^{-1}(i)}))$.

Mixnet de Wikström

Para realizar nuestro protocolo (capítulo 6) utilizaremos la mixnet UC-segura propuesta por Wikström en [20]. Básicamente a mixnet de Wikström's procede como sigue:

1. Cada enviador P_i espera por las claves públicas de los mixers y computa el producto de todas las claves públicas $y = \prod_{i=1}^k y_i$. Luego cada enviador encripta su mensaje con la clave pública y, publica el texto cifrado en una *pizarra pública* [1] y prueba que es un

[1] Una funcionalidad ideal $\mathcal{F}_{BB}$ que se comporta como una pizarra visible por todos los participantes y donde todos pueden escribir. Vista de otra forma, es una funcionalidad que permite hacer broadcast a todos los participantes.

texto cifrado válido usando un protocolo ZKP.

2. Cada mixer M_j $j \in 1, \ldots, k$ descarta todos los textos cifrados que no son válidos. Luego, para $l = 1, \ldots, k$ si $l = j$ el mixer M_l desencripta parcialmente la lista de textos cifrados obtenida de la pizarra pública, les aplica una permutación escogida al azar, publica la lista en la pizarra pública y prueba usando un protocolo ZKP que la lista publicada es una reencriptación de una permutación aleatoria de la lista anterior. Si $l \neq j$ el mixer M_j debe chequear que la permutación publicada por M_l es valida. Finalmente el mixer con índice mayor ordena la lista de textos planos resultantes de la desencriptación conjunta, y publica la lista.

Adicionalmente, para fines de este trabajo, modificamos la mixnet de [20] para que el protocolo no empiece a menos que haya una cantidad mínima de mensajes κ. La modificación se puede ver en la figura 5.3.

En [19] se muestra que este protocolo UC-realiza a la funcionalidad ideal $\mathcal{F}_{MN}$, definida en la figura 5.3, en el modelo $\mathcal{F}_{KG}$-híbrido.

Teorema 6. *El protocolo de [20] UC-realiza a la funcionalidad ideal $\mathcal{F}_{MN}$ en el modelo $\mathcal{F}_{KG}$-híbrido con respecto a adversarios estáticos y bajo DDH en un grupo G_q.*

Notamos que con la modificación, si bien se generaliza a la mixet de [20], el teorema anterior se sigue teniendo. En efecto si $\mathcal{C}_{Env}$ es el conjunto de todos los ambientes y $\mathcal{C}_{Adv}$ es el conjunto de todos los adversarios, entonces el conjunto de ambientes y adversarios $\mathcal{C}_\kappa$ para los que la cantidad de mensajes enviados a la mixnet es mayor que κ es claramente subconjunto de $\mathcal{C}_{Env} \times \mathcal{C}_{Adv}$. Por lo tanto para cada $\mathcal{Z}$ y $\mathcal{A}$ tales que $(\mathcal{Z}, \mathcal{A}) \in \mathcal{C}_\kappa$ debe existir un simulador $\mathcal{S}^{\mathcal{Z},\mathcal{A}}$ talque el $\mathcal{Z}$ no distingue entre el mundo real e el mundo ideal.

5.2.4. Anonimous Broadcast o DC-nets

A diferencia de las mixnets, las DC-nets, propuestas por Chaum en [9], no son interactivas. No es necesaria la existencia de otros participantes mas que los enviadores y receptores, por el contrario en las mixnets se necesita que los mixer manipulen los textos cifrados antes de que puedan ser leídos por los receptores. Cada enviador P_i publica un vector de textos cifrados $c_i = ((\delta_1(m_i, r_i) \cdot c'_{i,1}, \ldots, \delta_m(m, r_i) \cdot c'_{i,n})$ con $r_i \in 0^{\ell_i}10^{m-\ell_i}$, $\ell_i \in \{1, \ldots m\}$ distinto para cada

La funcionalidad ideal $\mathcal{F}_{MN}^{\kappa}$ corriendo con mixers $M_1, \ldots, M_k$, enviadores $P_1, \ldots, P_N$, y adversario ideal $\mathcal{S}$:

1. Inicializar una lista $L = \emptyset$, conjuntos $J_P = \emptyset$ y $J_M = \emptyset$ y variable $c = 0$.
2. Si $(P_i, \texttt{Send}, m_i)$ $m_i \in G_q$ es recibido desde $\mathcal{C}_{\mathcal{I}}$. Si $i \notin J_P$, hacer $J_P \leftarrow J_P \cup \{i\}$, $c \leftarrow c+1$, adjuntar m_i a la lista L. Luego enviar $(\mathcal{S}, P_i, \texttt{Send})$ a $\mathcal{C}_{\mathcal{I}}$.
3. Supongamos que $(M_j, \texttt{Run})$ es recibido desde $\mathcal{C}_{\mathcal{I}}$. Hacer $J_M \leftarrow J_M \cup \{j\}$. Si $|J_M| \geq k/2$ y $c \geq \kappa$, luego ordenar la lista lexicográficamente para formar una lista L', y enviar $((\mathcal{S}, M_j, \texttt{Output}, L'), \{M_l, \texttt{Output}, L'\}_{l=1}^{k})$ a to $\mathcal{C}_{\mathcal{I}}$. De lo contrario, enviar a $\mathcal{C}_{\mathcal{I}}$ la lista $(\mathcal{S}, M_j, \texttt{Run})$

Figura 5.3: La funcionalidad ideal $\mathcal{F}_{MN}$

i y:

$$\delta_k(m, r) = \begin{cases} m & r[k] = 1 \\ 1 & r[k] = 0 \end{cases} \quad \forall i \in \{1, \ldots, n\}$$

$$\prod_{i=1}^{n} c'_{i,j} = 1 \quad \forall j \in \{1, \ldots, n\}$$

$$\ell_i \neq \ell_j \text{ si } i \neq j$$

Posteriormente es posible obtener una permutación el vector $(\prod_{i=1}^{n} c_{i,j})_{j=1}^{n}$, que corresponde a una permutación del vector $(m_i)_{i=1}^{n}$. Si la distribución que sigue c_i es indistinguible de la que sigue $m_i \cdot c_i$ $\forall i$, entonces es imposible para un adversario identificar al autor de algún texto plano.

Capítulo 6

El protocolo

En este capítulos detallamos los resultados de esta memoria. Primero diseñamos una funcionalidad ideal y demostramos que es anónima y desmentible, posteriormente diseñamos un protocolo que GUC-emula a la funcionalidad ideal. Adicionalmente demostramos que el protocolo diseñado es anónimo y desmentible solucionando el problema a resolver en este trabajo.

6.1. Canales Anónimos Autentificados

Un "canal anónimo autentificado" debe permitir a los participantes enviar mensajes a cualquier otro participante sin revelar su identidad mas que al destinatario de mensaje. Definimos formalmente un canal anónimo autentificado a través de la definición de una funcionalidad ideal que llamaremos $\mathcal{F}_{AAC}$ (figura 6.1).

La funcionalidad de la figura 6.1 es anónima según la definición de anonimato descrita en el capítulo 5 demostrando el siguiente lema.

Lema 1. *La funcionalidad $\mathcal{F}_{AAC}$ es Anónima $f_{\cup\cup}$.*

Demostración. (Lema 1) La demostración es directa notando que para cualquier par de matrices $M_1, M_2 \in \mathcal{R}_{\cup\cup}$ las vistas del adversario son la misma, pues el conjunto de mensajes intercambiados es el mismo. Por lo tanto para todo ℓ

$$2 \cdot \Pr[\mathrm{Exp}_{\pi,\mathcal{A}}^{\mathcal{R}-anon}(\ell) = 1] - 1 = 0$$

Y 0 es una función despreciable. □

Una acotación al lema 1 es que la definicion de anonimato usada no necesariamente es valida para protocolos ejecutados en un ambiente concurrente. Sin embargo hacemos las siguientes acotaciones.

La funcionalidad ideal $\mathcal{F}_{AAC}$ corriendo con participantes $P_1, \ldots, P_N$ y adversario $\mathcal{S}$, parametrizada por un grupo G_q y $n \in \mathbb{N}$ procede como sigue:

1. Inicializar$\Gamma \leftarrow \emptyset$, $M \leftarrow \emptyset$ y $k \leftarrow 0$.
2. Si $(\tilde{P}_i, \texttt{Send}, m_{i,j}, j)$ es recibido desde $\mathcal{C}_{\mathcal{I}}$ y mientras $k < n$:
   a) Si P_i ó P_j no están registrados en $\bar{\mathcal{G}}_{KRK}$ o $m_i \notin G_q$ enviar $(\tilde{P}_i, \perp)$ a $\mathcal{C}_{\mathcal{I}}$.
   b) y hacer $\Gamma \leftarrow \Gamma \cup \{(m_i, i, j)\}$, $M \leftarrow M \cup \{m_{i,j}\}$ y $k \leftarrow k + 1$.
   c) Si $\tilde{P}_j$ es corrupto enviar $(\mathcal{S}, \tilde{P}_i, \tilde{P}_j, \texttt{Corruptsend}, m_{i,j})$ a $\mathcal{C}_{\mathcal{I}}$. En caso contrario enviar $(\mathcal{S}, P_i, \texttt{Send})$ a $\mathcal{C}_{\mathcal{I}}$.
3. Una vez que $k = n$, para cada $j \in \{1, \ldots, N\}$ sea el multiconjunto $M_j = \{(m_i, i) | , (m_i, i, j) \in \Gamma\}$ y enviar $(\tilde{P}_j, \texttt{Messages}, M_j)$ y $(\mathcal{S}, \texttt{Messages}, M)$ a $\mathcal{C}_{\mathcal{I}}$.

Figura 6.1: La funcionalidad ideal $\mathcal{F}_{AAC}$

Primero, si comparamos la funcionalidad $\mathcal{F}_{AAC}$ con otras funcionalidades de anonimato *ad-hoc* definidas en la literatura, como la funcionalidad Anon de [15], notamos que $\mathcal{F}_{AAC}$ es estrictamente mas fuerte. En efecto Anon adicionalmente revela el multiconjunto de mensajes recibidos por cada participante P_j.
Segundo, afirmamos que nuestra funcionalidad es intuitivamente anónima. En este caso estaríamos afirmando que $\mathcal{F}_{AAC}$ es nuestra definción de anonimato.

Ahora demostraremos que $\mathcal{F}_{AAC}$ es desmentible

Lema 2. *La función $\mathcal{F}_{AAC}$ es desmentible.*

Demostración. (Lema 2)
La demostración es simple notando que para las firmas honestas (donde tanto el emisor como el destinatario son honestos) solo es necesario escoger $k_{i,j} \in_R G_q$, y cuando el emisor o el destinatario son deshonestos las firmas pueden ser obtenidas con la clave publica del honesto y la clave privada del deshonesto. Usando lo anterior el desinformante $\mathfrak{D}$ puede simular $\mathcal{F}_{AAC}$ y su simulación sigue la misma distribución que una ejecución real de $\mathcal{F}_{AAC}$. Por lo tanto para cualquier informante $\mathfrak{I}$ es posible construir un desinformante $\mathcal{D}^{\mathcal{I}}$ que simula $\mathcal{F}_{AAC}$ para una simulación del informante $\mathfrak{I}$. De este modo la vista del juez $\mathcal{J}$ es idéntica tanto en Real como en Sim, lo que nos permite concluir. □

6.2. El protocolo SIGMIX

Una primera forma "natural" de realizar $\mathcal{F}_{AAC}$ es simplemente combinando un canal anónimo con un protocolo que GUC-realice la funcionalidad ideal $\mathcal{F}_{CERT}$ descrita en [5],

pues esta es la funcionalidad "clásica para autentificar mensajes. Pero este intento falla pues la funcionalidad ideal $\mathcal{F}_{CERT}$ permite que cualquier participante verifique la autenticidad de un par (m, σ). Esto trae consigo la pérdida del anonimato al relacionar públicamente la identidad del enviador de m con (m, σ). Además cada instancia de $\mathcal{F}_{CERT}$ esta restringida a solo dos participantes, por lo que por trivialmente se conoce la identidad del enviador y receptor de cada mensaje.

En consecuencia, proveer anonimato y autentificación puede parecer contradictorio. Pero notamos que dicha noción puede ser alcanzada por un protocolo que satisface los siguientes puntos:

1. Los mensajes están firmados.
2. Solo el destinatario puede probar que el participante P_i es autor de un mensaje que recibió.
3. El destinatario no puede probar a nadie que P_i es el autor de un mensaje que recibió.
4. El envío de mensajes es hecho en forma anónima.

De este modo, para implementar canales anónimos autentificados, usamos una versión modificada del protocolo de autentificación desmentible GUC-seguro con respecto a adversarios estáticos de [11]. Notamos que en [11] se usa un protocolo de firmado desmentible que nos es útil para satisfacer los puntos 1, 2 y 3 mencionados anteriormente. El proceso de firma es hecho a través de una firma que depende no solo del contenido del mensaje y la identidad del enviador, si no que adicionalmente depende en la identidad del destinatario. Así, solo le es permitido al receptor verificar la autenticidad del par (m, σ). El punto 4 es satisfecho usando la mixnet propuesta por Wikström descrita en la figura 5.3. El protocolo SIGMIX es ejecutado en el modelo $\mathcal{F}_{MN}, \bar{\mathcal{G}}_{KRK}$-híbrido con adversarios estáticos. La funcionalidad compartida *Key registration with knowledge* $\bar{\mathcal{G}}_{KRK}$ de [11] descrita en la figura 2.4 provee un PKI para cualquier protocolo que es ejecutado concurrentemente con el protocolo SIGMIX. Remarcamos que cualquier protocolo que usa $\bar{\mathcal{G}}_{KRK}$ puede compartir el par clave pública y privada (sk, pk) con SIGMIX, siempre y cuando no revelen la clave privada a terceros. Por otro lado consideramos a la funcionalidad $\mathcal{F}_{MN}$ como una funcionalidad ideal tradicional de UC, esto significa que cada instancia de $\mathcal{F}_{MN}$ es local a cada protocolo que la llama.

El protocolo SIGMIX$^{\kappa}$ corriendo con participantes $P_1, \ldots, P_N$ y Mixers $M_1, \ldots, M_k$ en el modelo $\mathcal{F}_{MN}^{\kappa}, \bar{\mathcal{G}}_{KRK} - hybrid$ con $\kappa \in \mathbb{N}$:

Enviador P_i: Cada enviador P_i procede como sigue:

1. Esperar a recibir la entrada $(\texttt{Send}, P_j, m_{i,j})$.
2. Enviar $(\texttt{Retrieve}, P_j)$ a $\bar{\mathcal{G}}_{KRK}$ y sea y_j la respuesta.
3. Si la respuesta fue $\perp$ retornar $\perp$. De lo contrario calcular $k_{i,j} \leftarrow y_j^{x_i}$ y luego calcular $\sigma_{i,j} = \mathrm{MAC}_{k_{i,j}}(m_{i,j})$.
4. Enviar $(\texttt{Send}, m_{i,j}||\sigma_{i,j})$ a $\mathcal{F}_{MN}$ y enviar $(\texttt{Write}, m_{i,j})$ a $\mathcal{F}_{\mathcal{BB}}$.
5. Retornar $(\texttt{Sent}, P_j, m_{i,j})$

Destinatario P_j: Cada destinatario P_j procede como sigue:

1. Esperar a recibir la entrada $(\texttt{Output}, L)$ de $\mathcal{F}_{MN}$.
2. Sean $y_1, \ldots, y_N$ las claves publicas de todos los participantes del protocolo. Para cada $i \in \{1, \ldots, N\}$ computar el secreto compartido $k_{ij} \leftarrow y_i^{x_j}$.
3. Sea el multiconjunto $M_j \leftarrow \emptyset$. Para cada $(m_l, \sigma_l) \in L$ y para cada k_{ij}, si $\sigma_l = \mathrm{MAC}_{k_{ij}}(m_l)$ entonces $M_j \leftarrow M_j \uplus \{(m_l, l)\}$.
4. Retornar $(\texttt{Messages}, M_j)$.

Mixer M_i: Cada Mixer M_i envía $(\mathcal{F}_{MN}, \texttt{Run})$ a $\mathcal{C}_{\mathcal{I}}$.

Figura 6.2: El protocolo SIGMIX

Para proceder con SIGMIX cada enviador P_i firma un mensaje m_i a P_j con una función MAC que es UF-CMA (sección 2.3.3). La clave con que MAC es usada es la clave secreta $k_{i,j}$ compartida entre P_i y P_j. La clave $k_{i,j}$ es calculada por P_i y P_j como sigue. Sean G_q un grupo cíclico de orden q donde DDH se cumple, y sea g un generador para G_q. Supongamos que P_i y P_j tienen registrados los pares de claves publicas/privadas $(x_i, y_i = g^{x_i})$ y $(x_j, y_j = g^{x_j})$ respectivamente, tales que $x_i, x_j \in_R G_q$. Entonces la clave secreta compartida $k_{i,j}$ puede ser no interactivamente computada [1] por P_i con $k_{i,j} = y_j^{x_i}$ y por P_j con $k_{i,j} = y_i^{x_j}$. El mensaje firmado $(m_i, \sigma_{i,j} = MAC_{k_{i,j}}(m_i)$ es enviado a P_j usando la mixnet, y finalmente P_j puede chequear la autenticidad del mensaje recalculando la firma. El protocolo SIGMIX se encuentra descrito en la figura 6.2.

[1] Es decir que puede se computada sin necesidad de intercambiar mensajes entre P_i y P_j.

La seguridad del protocolo SIGMIX viene garantizada por el teorema 7.

Lema 3. *El protocolo SIGMIX GUC-emula a la funcionalidad ideal $\mathcal{F}_{AAC}$ en el modelo $\mathcal{F}_{MN}, \bar{\mathcal{G}}_{KRK}$-híbrido con respecto a adversarios estáticos que corrompen a lo más $k/2-1$ mixers.*

Demostración. (Teorema 3)
La demostración procede de la siguiente forma. Para cada adversario real $\mathcal{A}$ construimos un adversario ideal $\mathcal{S}^{\mathcal{A}}$ que ataca $\mathcal{F}_{AAC}$. Si existe un adversario $\mathcal{A}$ y un ambiente $\mathcal{Z}$ que es capaz de distinguir $\mathcal{H}(\mathcal{S}, \{\tilde{P}_i\}_1^{\mathcal{F}_{AAC}}, \{\tilde{P}_i\}_2^{\bar{\mathcal{G}}_{KRK}})$ de una ejecución de $\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \{\tilde{P}_i\}_1^{\bar{\mathcal{G}}_{KRK}}, \{\tilde{P}_i\}_2^{\mathcal{F}_{MN}})$, entonces podemos contradecir DDH sobre G_q o la seguridad de MAC. Sea $I_{\mathcal{A}} \subseteq \{1, \ldots, N\}$ el conjunto de índices de los participantes que son corruptos por $\mathcal{A}$. El adversario ideal $\mathcal{S}^{\mathcal{A}}$ esta descrito en la figura 6.3, y simula una ejecución de SIGMIX solo con acceso a $\mathcal{F}_{AAC}$. Como los valores de los mensajes enviados honestamente (tanto el enviador como el destinatario son honestos) permanecen desconocidos para $\mathcal{S}^{\mathcal{A}}$ hasta que todos los mensajes son enviados, $\mathcal{S}^{\mathcal{A}}$ engaña a la simulación interna de $\mathcal{A}$ haciendo que $\mathcal{F}_{MN}$ le diga a $\mathcal{A}$ que los mensajes fueron enviados siendo que esto no es realmente así. Finalmente, cuando el conjunto de mensajes honestamente enviados le hes revelado a $\mathcal{S}$, hace que los participantes honestos de SIGMIX envíen "silenciosamente" sus mensajes a $\mathcal{F}_{MN}$. Es es para $\mathcal{A}$ indistinguible de una ejecución donde un adversario **hipotético** $\mathcal{S}'^{\mathcal{A}}$ adivina los mensajes enviados por $\mathcal{Z}$ a cada participante honesto, puesto que la vista de $\mathcal{A}$ es la misma en ambos casos.
El adversario $\mathcal{S}^{\mathcal{A}}$ corrompe a los mismos mixers que $\mathcal{A}$ corrompe, que corresponden a los mixers M_i $i \in I_{\mathcal{A}}^M$, y los simula copiando la forma en que $\mathcal{A}$ los ejecuta. Los Mixers honestos son ejecutados honestamente.

Definimos $\text{Real}(\ell) = \mathcal{Z}(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \{\tilde{P}_i\}_1^{\bar{\mathcal{G}}_{KRK}}, \{\tilde{P}_i\}_2^{\mathcal{F}_{MN}}))$ y $\text{Ideal}(\ell) = \mathcal{Z}(\mathcal{H}(\mathcal{S}^{\mathcal{A}}, \{\tilde{P}_i\}_1^{\mathcal{F}_{AAC}}, \{\tilde{P}_i\}_2^{\bar{\mathcal{G}}_{KRK}}))$, donde ℓ es el parámetro de seguridad conque se ejecutan los grafos de ITMs. Supongamos que existe un ambiente $\mathcal{Z}$, un adversario $\mathcal{A}$, un polinomio p y entero $\bar{\ell}$ talque para todo $\ell > \bar{\ell}$ se tiene que:

$$|\Pr[\text{Real}(\ell) = 1] - \Pr[\text{Ideal}(\ell) = 1]| \geq \frac{1}{p(\ell)} \tag{6.1}$$

Consideremos al adversario $\mathcal{D}_{\text{DDH}}$ defindo en la figura 6.4.

Notemos que como en el experimento $\alpha \in_R G_q$ y $\beta \in_R G_q$, entonces para el ambiente son indistinguibles las modificaciones echas por $\mathcal{D}_{DDH}$. Luego

$$\Pr[\text{Real}(\ell) = 1] = \Pr[\mathcal{D}_{DDH} = 1 | \gamma = \alpha\beta] \tag{6.2}$$

Ahora, la única posible diferencia entre ejecutar $\mathcal{D}_{DDH}$ cuando $\gamma \in_R G_q$ y ejecutar $\mathcal{Z}(\mathcal{H}(\mathcal{S}^{\mathcal{A}}, \{\tilde{P}_i\}_1^{\mathcal{F}_{AAC}}, \{\tilde{P}_i\}^{\bar{\mathcal{G}}_{KRK}}))$ es la salida de los participantes honestos. Definimos un adversario $\mathcal{D}_{UF-CMA}$ en la figura 6.5 de modo tal que si existe un participante honesto con salida distinta en ambos casos, $\mathcal{D}_{UF-CMA}$ puede falsificar una firma de MAC.

Condicionamos $\Pr[\text{Ideal}(\ell) = 1]$ sobre el evento $\mathcal{U} = \text{UF-CMA}_{\text{MAC},\mathcal{D}_{UF-CMA}}(\ell)$

$$\Pr[\text{Ideal}(\ell) = 1] = \Pr[\text{Ideal}(\ell) = 1 | \mathcal{U} = 0] \Pr[\text{U} = 0] + \Pr[\text{Ideal}(\ell) = 1 | \text{U} = 1] \Pr[\text{U} = 1]$$

$$\Pr[\text{Ideal}(\ell) = 1] = \Pr[\text{Ideal}(\ell) = 1 | \mathcal{U} = 0] \text{Adv}_{\text{MAC},\mathcal{A}}^{UF-CMA}(\ell) + \Pr[\text{Ideal}(\ell) = 1 | \mathcal{U} = 1](1 - \text{Adv}_{\text{MAC},\mathcal{A}}^{UF-CMA}(\ell))$$

El adversario ideal $\mathcal{S}^{\mathcal{A}}$ corriendo con participantes $\tilde{P}_1, \ldots, \tilde{P}_N$, Mixers $\tilde{M}_1, \ldots, \tilde{M}_k$ y funcionalidad ideal compartida $\bar{\mathcal{G}}_{KRK}$ procede como sigue:

Inicialmente $\mathcal{S}^{\mathcal{A}}$ corrompe a los participante $\tilde{P}_i$ $i \in I_{\mathcal{A}}$ y corrompe a los Mixers $\tilde{M}_i$ $i \in I^M_{\mathcal{A}}$. Adicionalmente ejecuta una simulación de $\mathcal{Z}'(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\pi}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\rho}^{\mathcal{F}_{MN}}))$, donde $\mathcal{Z}'$ es una ITM controlada por $\mathcal{S}^{\mathcal{A}}$, y $\bar{\mathcal{G}}_{KRK}$ y $\mathcal{F}_{MN}$ son ejecutadas honestamente con algunas modificaciones menores.

Simulación de links $(\mathcal{Z}', \mathcal{A})$ con $(\mathcal{Z}, \mathcal{S})$:
Si m es recibido de $\mathcal{Z}$ entonces hacer que $\mathcal{Z}'$ envíe m a $\mathcal{A}$. Si m es enviado de $\mathcal{A}$ a $\mathcal{Z}'$ entonces enviar m a $\mathcal{Z}$

Simulación de participantes corruptos $\tilde{P}_i$ $i \in I_{\mathcal{A}}$:

1. Si P_i $i \in I_{\mathcal{A}}$ envía $m||\sigma$ a $\mathcal{F}_{MN}$ y $\sigma = \text{MAC}_{y_j^{x_l}}(m)$ para alguna clave pública registrada y_j $j \in \{1, \ldots, N\}$ y alguna clave secreta registrada x_l $l \in I_{\mathcal{A}}$, entonces enviar $(\texttt{Send}, m, j)$ a $\tilde{P}_l$. Si $\sigma \neq \text{MAC}_{y_j^{x_l}}(m)$ para toda clave pública registrada y_j $j \in \{1, \ldots, N\}$ y toda clave privada registrada x_l $l \in I_{\mathcal{A}}$ no hacer nada.

Simulación de participantes honestos P_i $i \notin I_{\mathcal{A}}$:

1. Sea $M'_j = \emptyset$ para $j = 1, \ldots, k$, $l_s \leftarrow 0$ y $l_r \leftarrow 0$.

2. Si $(\tilde{P}_i, \texttt{Send})$ es recibido de $\mathcal{C}_{\mathcal{I}}$ hacer que $\mathcal{Z}'$ envíe $(\texttt{Register})$ a P_i y hacer que $\mathcal{F}_{MN}$ envíe $(\mathcal{S}_{\mathcal{F}_{MN}}, \tilde{P}_i, \texttt{Send})$ a su copia de $\mathcal{C}_{\mathcal{I}}$.

3. Si $(\tilde{P}_i, \tilde{P}_j, \texttt{Corruptsend}, m_{i,j})$ es recibido de $\mathcal{C}_{\mathcal{I}}$ recuperar x_j e y_i de $\bar{\mathcal{G}}_{KRK}$ (no la simulada). Sea $\sigma_{i,j} \leftarrow \text{MAC}_{y_i^{x_j}}(m_{i,j})$, hacer que $\mathcal{Z}'$ envíe $(\texttt{Register})$ a P_i y que luego envíe $(\texttt{Send}, m, j)$ a P_i.

4. Si $(\texttt{Messages}, M)$ es recibido desde $\mathcal{C}_{\mathcal{I}}$ entonces para cada $m \in M$ escoger $i, j \overset{R}{\leftarrow} \{1, \ldots, N\} \setminus I_{\mathcal{A}}$, hacer que $\mathcal{Z}'$ envíe $(\texttt{Send}, m, j)$ a P_i y eliminar el mensaje $(\mathcal{S}, P_i, \texttt{Send})$ que $\mathcal{F}_{MN}$ envía a su copia de $\mathcal{C}_{\mathcal{I}}$ (similarmente $\mathcal{S}^{\mathcal{A}}$ puede solo adjuntar M a la lista L de $\mathcal{F}_{MN}$).

Figura 6.3: El adversario ideal $\mathcal{S}^{\mathcal{A}}$

El adversario $\mathcal{D}_{DDH}(g^\alpha, g^\beta, g^\gamma)$ atacando DDH funciona como sigue:

1. Simular $\mathcal{Z}(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\phi}^{\mathcal{F}_{MN}}))$ con aletoridad r escogida al azar hasta que este participante P_i honesto envíe un mensaje a otro participante honesto P_j y volver a simular $\mathcal{Z}(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\phi}^{\mathcal{F}_{MN}}))$ con aleatoridad r.
2. Cuando $\tilde{P}_i$ se registra en $\bar{\mathcal{G}}_{KRK}$ actualizar la clave pública registrada a g^α y cuando $\tilde{P}_j$ se registra actualizar la clave pública a g^β.
3. Cuando $\mathcal{Z}$ envía $(\texttt{Send}, m, j')$ a P_i reemplazar la firma computada por P_i por $\text{MAC}_{y_j^\alpha}(m)$ para cada $j' \in \{1, \ldots, N\} \setminus j$ y $\text{MAC}_{g^\gamma}(m)$ para $j' = j$.
4. Cuando $\mathcal{Z}$ envía $(\texttt{Send}, m, j')$ a P_j reemplazar la firma computada por P_j por $\text{MAC}_{y_{j'}^\beta}(m)$ para cada $j' \in \{1, \ldots, N\} \setminus i$ y $\text{MAC}_{g^\gamma}(m)$ para $j' = i$.
5. Cuando $\mathcal{Z}$ envía $(\texttt{Send}, m, j)$ a $P_{i'}$ con $i' \neq i$, reemplazar la firma computada por $P_{i'}$ por $\text{MAC}_{y_j^\alpha}(m)$.
6. Cuando $\mathcal{Z}$ envía $(\texttt{Send}, m, i)$ a $P_{j'}$ con $j' \neq j$, reemplazar la firma computada por $P_{i'}$ por $\text{MAC}_{y_i^\beta}(m)$.
7. Hacer lo anterior para cualquier copia de SIGMIX que $\mathcal{Z}$ ejecute concurrentemente.
8. Cuando la simulación se detenga retornar lo que $\mathcal{Z}$ tenga en su cinta de salida.

Figura 6.4: El adversario para DDH sobre G_q $\mathcal{D}_{\text{DDH}}$

El adversario $\mathcal{D}_{\text{UF}-\text{CMA}}$ atacando MAC funciona como sigue:

1. Simular $\mathcal{Z}(\mathcal{H}(\mathcal{A}, \text{SIGMIX}, \tilde{\rho}^{\bar{\mathcal{G}}_{KRK}}, \tilde{\phi}^{\mathcal{F}_{MN}}))$ y $\mathcal{Z}(\mathcal{H}(\mathcal{S}^{\mathcal{A}}, \tilde{P}_1, \ldots, \tilde{P}_N, \mathcal{F}_{AAC}, \{\tilde{P}_i\}_1^{\bar{\mathcal{G}}_{KRK}}, \{\tilde{P}_i\}_2^{\mathcal{F}_{MN}}))$.
2. Para cada $j \in \{1, \ldots, N\}$, sean M_j^{Real} los mensajes recibidos por P_j en la primera simulación y M_j^{Ideal} los recibidos por $\tilde{P}_j$ en la segunda simulación. Si existe un $m \in M_j^{real}$ y $m \notin M_j^{Ideal}$, buscar el par $m||\sigma$ en la lista L de $\mathcal{F}_{MN}$ en la primera simulación.
3. Retornar el par (m, σ)

Figura 6.5: El adversario UF-CMA para MAC

Como MAC es UF-CMA entonces $\text{Adv}^{UF-CMA}_{\mathcal{MA},\mathcal{A}}(\ell) < \eta(\ell)$, con η una función despreciable. Reemplazando en la ecuación anterior

$$\Pr[\text{Ideal}(\ell) = 1] = \Pr[\text{Ideal}(\ell) = 1 | \text{UF-CMA}_{\text{MAC},\mathcal{A}}(\ell) = 0] + \delta \cdot \eta(\ell)$$

y δ viene dado por $\delta = \Pr[\text{Ideal}(\ell) = 1 | \mathcal{U} = 1] - \Pr[\text{Ideal}(\ell) = 1 | \mathcal{U} = 0]$, por lo que $-1 \leq \delta \leq 1$ y podemos escribir.

$$\Pr[\text{Ideal}(\ell) = 1] \approx \Pr[\text{Ideal}(\ell) = 1 | \text{UF-CMA}_{\text{MAC},\mathcal{A}}(\ell) = 0] \tag{6.3}$$

En el caso en que $\mathcal{U} = 0$ los mensajes recibidos por los receptores son los mismos en Ideal y Real, por lo tanto al la simulación de $\mathcal{S}^{\mathcal{A}}$ es idéntica a la simulación que hace $\mathcal{D}_{DDH}$ cuando $\gamma \in_R G_q$. Es decir

$$\Pr[\text{Ideal} = 1 | \mathcal{U} = 0] \approx \Pr[\mathcal{D}_{DDH} = 1 | \gamma \in_R G_q] \tag{6.4}$$

Combinando las ecuaciones 6.2, 6.4, **??** 6.1 contradecimos que DDH se cumple en G_q. Por lo tanto el teorema se tiene. □

Notamos que el modo híbrido donde es ejecutado SIGMIX puede ser simplificado.

Teorema 7. *El protocolo SIGMIX GUC-emula a la funcionalidad $\mathcal{F}_{AAC}$ en el modo $\mathcal{F}_{KG}, \bar{\mathcal{G}}_{KRK}$*

Demostración. (Teorema 7)
Directo usando el lema 3, el teorema 6 y el teorema de composición (teorema 1) □

Capítulo 7

Conclusiones

En este trabajo se planteó el problema de comunicación anónima autentificada y se demostró constructivamente que existe un protocolo que resuelve dicho problema. Para ello se estudiaron tópicos avanzados de Criptografía como UC, GUC, Anonimato, Desmentibilidad y distintas primitivas criptográficas asociadas a dichos tópicos. Se definieron rigurosamente las propiedades que debe tener un protocolo para resolver el problema planteado. Se desarrolló un protocolo para el cual se puede garantizar matematicamente que satisface las propiedades necesarias para resolver el problema inicial. El protocolo desarrollado es eficiente, pues se construye a partir de un protocolo que se sabe eficiente por la comunidad [20] y adicionalmente efectua una operación que es lineal en el número de participantes del protocolo.

7.1. Preguntas abiertas

Adicionalmente, del trabajo realizado se desprenden las siguientes preguntas que han quedado abiertas, pues escapan a los alcances de este trabajo. Sin embargo su sola formulación debe ser considerada un resultado de este trabajo, pues estas preguntas proponen interesantes líneas de investigación.

7.1.1. Mixnet GUC-segura

El protocolo SIGMIX se construye a partir de una mixnet que en [20] se demuestra UC-segura. Sin embargo, dado que para poder UC-realizar la mixnet es necesario ocupar una setup assumption $\mathcal{F}_{KG}$, es válido preguntarse si el protocolo de [20] sigue siendo seguro cuando $\mathcal{F}_{KG}$ es una funcionalidad compartida. En caso negativo es necesario preguntarse si existe

algún protocolo seguro para una mixnet con la setup assumption $\mathcal{F}_{KG}$. Finalmente, si todos los esfuerzos han fracasado es necesario preguntarse si existe una setup assumption compartida razonable con la cual se pueda GUC-realizar una mixnet.

7.1.2. Una modelación más exacta de la componibilidad en GUC

El objetivo principal de GUC era modelar protocolos que se pueden componer concurrentemente con otros protocolos que posiblemente comparten estado y de los cuales nada se puede asumir. Sin embargo en el trabajo de esta memoria nos dimos cuenta de que en la literatura, para el caso de PKI y CRS, [1] **sí se hacen suposiciones sobre los protocolos con los cuales el protcolo analizado** se componen.

Para caso de PKI se asume que los protocolos que hacen uso de la funcionalidad ideal $\bar{\mathcal{G}}_{KRK}$ se encuentran dentro de un conjunto de prtocolos Φ. En general general Φ solo contiene al protocolo analizado, por lo que la situación puede resultar muy similar a JUC.

El caso de CRS es algo distinto, pero se puede mostrar que es equivalente. En efecto, a la funcionalidad ideal que modela CRS se le especifica que debe distinguir entre los participantes honestos y no honestos. Si bien en [18] notan que asumir esto es irreal, afirman que en la práctica los participantes deben tener cuidado con no revelar cierto valor secreto. Lo anterior no es más que una limitación a los protocolos que puede ejecutar cada participante, pues Φ correspondería a protocolo que no revelan el valor antes mencionado.

Adicionalmente, que solo protocolos en un conjunto Φ pueden acceder a la funcionalidad compartida no es posible de formalizar en EUC. Si bien es posible de formalizar en GUC, es posible pensar en un modelo más general donde Φ no depende de propiedades estáticas de los protocolos si no que de propiedades dinámicas. Lo anterior no es formalizable en GUC, y para ellos es necesario hacer ciertas modificaciones a GUC

[1] En este caso se hace uso de uno funcionalidad compartidad más fuerte que CRS conocida como *Augmented Common Random String*

Referencias

[1] Canetti, Dwork, Naor, and Ostrovsky. Deniable encryption. In *CRYPTO: Proceedings of Crypto*, 1997.

[2] Canetti, Kushilevitz, and Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *JCRYPTOL: Journal of Cryptology*, 19, 2006.

[3] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`.

[4] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[5] R. Canetti. Universally composable signature, certification, and authentication. In *CSFW*, page 219. IEEE Computer Society, 2004.

[6] R. Canetti. Security and composition of cryptographic protocols: A tutorial. Cryptology ePrint Archive, Report 2006/465, 2006.

[7] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

[8] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[9] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[10] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(5):644–654, 1976.

[11] Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2009.

[12] Dwork, Naor, and Sahai. Concurrent zero-knowledge. *JACM: Journal of the ACM*, 51, 2004.

[13] Golle, Jakobsson, Juels, and Syverson. Universal re-encryption for mixnets. In *CTRSA: CT-RSA, The Cryptographers' Track at RSA Conference, LNCS*, 2004.

[14] A. Hevia and D. Micciancio. An indistinguishability-based characterization of anonymous channels. In N. Borisov and I. Goldberg, editors, *Privacy Enhancing Technologies*, volume 5134 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2008.

[15] Ishai, Kushilevitz, Ostrovsky, and Sahai. Cryptography from anonymity. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[16] Pass. On deniability in the common reference string and random oracle model. In *CRYPTO: Proceedings of Crypto*, 2003.

[17] M. D. Raimondo and R. Gennaro. New approaches for deniable authentication. *J. Cryptology*, 22(4):572–615, 2009.

[18] S. Walfish. *Enhanced security models for network protocols*. PhD thesis, New York, NY, USA, 2008. AAI3310580.

[19] Wikstrom. A universally composable mix-net. In *Versión completa obtenida directamente del autor.*, 2004.

[20] Wikstrom. A universally composable mix-net. In *Theory of Cryptography Conference (TCC), LNCS*, volume 1, 2004.

[21] A. C.-C. Yao, F. F. Yao, and Y. Zhao. A note on universal composable zero-knowledge in the common reference string model. *Theor. Comput. Sci*, 410(11):1099–1108, 2009.