UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROTOCOLOS CRIPTOGRÁFICOS DE COMPUTAÇÂO DISTRIBUÍDA COM SEGURANÇA UNIVERSALMENTE COMPOSTA

ADRIANA CRISTINA BASTOS PINTO

ORIENTADOR: ANDERSON CLAYTON ALVES NASCIMENTO

DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGEE.DM - 493/2012

BRASÍLIA/DF: AGOSTO - 2012.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROTOCOLOS CRIPTOGRÁFICOS DE COMPUTAÇÃO DISTRIBUÍDA COM SEGURANÇA UNIVERSALMENTE COMPOSTA

ADRIANA CRISTINA BASTOS PINTO

DISSERTAÇÃO DE MESTRADO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

Anderson C A Nascimento

ANDERSON CLAYTON ALVES NASCIMENTO, Dr., PPGEE/UNB
(ORIENTADOR)

Flávio Elias

FLÁVIO ELIAS GOMES DE DEUS, Dr., PPGEE/UNB
(EXAMINADOR INTERNO)

G. Amvame

GEORGES DANIEL AMVAME-NZE, Dr., FGA/UNB
(EXAMINADOR EXTERNO)

Brasília, 09 de agosto de 2012.

FICHA CATALOGRÁFICA

PINTO, ADRIANA CRISTINA BASTOS
Protocolos Criptográficos de Computação Distribuída com Segurança Universalmente Composta. [Distrito Federal] 2012.
xiv, 120p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2012).
Dissertação de Mestrado - Universidade de Brasília.
Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Composabilidade Universal
2. Segurança Incondicional
3. Inicializador Confiável
4. Provas de Segurança

I. ENE/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

Pinto, A. C. B. (2012). Protocolos Criptográficos de Computação Distribuída com Segurança Universalmente Composta. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGEE.DM 493/2012, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 120p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Adriana Cristina Bastos Pinto.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Protocolos Criptográficos de Computação Distribuída com Segurança Universalmente Composta.

GRAU / ANO: Mestre / 2012

Adriana Cristina Bastos Pinto
QNM 36 conjunto C casa 48, Taguatinga
72.145-603 Brasília - DF - Brasil.

DEDICATÓRIA

A minha família.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Anderson Nascimento por me apresentar à pesquisa de criptografia e pelo constante apoio à realização deste trabalho.
Agradeço a todas as pessoas que me ajudaram na realização deste trabalho, em especial à Bernardo David e Diógenes Reis pelas discussões construtivas e pelo imenso apoio durante todo o mestrado.
Agradeço também ao meu amigo Wallison Lourenço Amorim por sempre me incentivar a terminar o quanto antes esse trabalho.
Agradeço especialmente aos co-autores dos trabalhos que compõe essa dissertação: Rafael Baião Dowsley, Kiril Morozov e Anderson Nascimento.
Agradeço ao Prof. Flávio de Deus e ao Prof. Georges Amvame por participarem da banca.
Finalmente, agradeço aos meus amigos e minha família pelo apoio durante todos esses anos de estudo.

Adriana Cristina Bastos Pinto

RESUMO

PROTOCOLOS CRIPTOGRÁFICOS DE COMPUTAÇÃO DISTRIBUÍDA COM SEGURANÇA UNIVERSALMENTE COMPOSTA

Autor: Adriana Cristina Bastos Pinto
Orientador: Anderson Clayton Alves Nascimento
Programa de Pós-graduação em Engenharia Elétrica
Brasília, Agosto de 2012

A computação distribuída segura ganha cada vez mais destaque com a expansão no armazenamento de dados e na conectividade. Neste contexto, a primitiva criptográfica *Oblivious Transfer* (OT) se torna um dos elementos chaves, pois com ela pode-se implementar qualquer computação de duas ou múltiplas partes. Será apresentado um protocolo de duas partes para *String Oblivous Transfer* baseado em canais com apagamentos generalizado seguro no modelo malicioso. Além disso, esse protocolo atinge a capacidade de *oblivious transfer* do canal com apagamentos generalizados quando a probabilidade de ocorrer apagamentos é de no mínimo 1/2. Outra característica explorada por esse protocolo é que ele é seguro quando composto concorrentemente com protocolos arbitrários. Esse tipo de segurança, chamada de segurança universalmente composta, é um dos focos principais deste trabalho. Além do protocolo para *string oblivious transfer*, será apresentado protocolos para álgebra linear distribuída segura. A saber, será apresentado um protocolo para calcular o determinante da soma de duas matrizes, um protocolo para calcular os autovalores da soma de duas matrizes e, finalmente, um protocolo para calcular os autovetores associados a um autovalor. Todos esses protocolos de álgebra linear são construídos no modelo criptográfico baseado em commodities e são provados seguros quando universalmente compostos.

ABSTRACT

CRYPTOGRAPHIC PROTOCOLS OF DISTRIBUTED COMPUTING WITH UNIVERSALLY COMPOSABLE SECURITY

Author: Adriana Cristina Bastos Pinto
Supervisor: Anderson Clayton Alves Nascimento
Programa de Pós-graduação em Engenharia Elétrica
Brasília, August of 2012

Secure distributed computation is becoming increasingly prominent with the growth in data storage and connectivity. In this context, the cryptographic primitive Oblivious Transfer (OT) becomes one of the keys, since with it one can implements any two-party or multi-party computation. We will present a two-party protocol for string oblivious transfer based on generalized erasure channels secure on the malicious model. Furthermore, this protocol reaches the oblivious transfer capacity of the generalized erasure channel when the erasure probability is at least 1/2. Another feature exploited by this protocol is that it is secure when concurrently composed with arbitrary protocols. This type of security called universally composable security is one of the main focuses of this work. In addition to the protocol for the string oblivious transfer, we will show protocols for secure distributed linear algebra. Namely, we will show a protocol to calculate the determinant of the sum of two matrices, a protocol to calculate the eigenvalues of the sum of two matrices, and, finally, a protocol to calculate the eigenvector associated with an eigenvalue. All these protocols for linear algebra are built in the commodity-based cryptography model and they are proved secure when universally composed.

SUMÁRIO

1 INTRODUÇÃO **1**

1.1 ORGANIZAÇÂO DO TRABALHO 4

2 PRELIMINARES **6**

2.1 NOTAÇÃO 6

2.2 MÁQUINAS DE TURING INTERATIVAS 7

2.3 INDISTINGUIBILIDADE 9

2.4 COMPROMETIMENTO DE BIT 10

2.5 ENTROPIA, INFORMAÇÃO MÚTUA E ESTATÍSTICA 12

2.5.1 Entropia 12

2.5.2 Informação Mútua 13

2.5.3 Informação Estatística 13

2.6 SEQUÊNCIAS TÍPICAS E CONJUNTAMENTE TÍPICAS 14

2.7 EXTRATORES 17

2.8 LIMITE DE CHERNOFF E DESIGUALDADE DE FANO 18

2.9 CRIPTOGRAFIA BASEADA EM COMMODITIES 19

3 COMPUTAÇÃO DE DUAS PARTES SEGURA **21**

3.1 MODELO ADVERSARIAL 22

3.2 CONSIDERAÇÕES SOBRE HIPÓTESES ADICIONAIS 23

3.3 CONDIÇÕES BASEADAS NA TEORIA DA INFORMAÇÃO PARA SEGURANÇA 24

3.4 COMPOSIÇÂO DE PROTOCOLOS 25

4 COMPOSIBILIDADE UNIVERSAL **27**

4.1 MODELO BÁSICO DE COMPUTAÇÃO 29

4.2 EXECUÇÃO DO PROTOCOLO NO MODELO REAL 32

4.3 EXECUÇÃO DO PROTOCOLO NO MODELO IDEAL 36

4.4 DEFINIÇÃO DE SEGURANÇA UC 39

4.5 TEOREMA DA COMPOSIÇÃO 42

4.5.1 Modelo de Computação Híbrida 42

4.5.2 Operação de Composição Universal 43

4.5.3 Enunciado do Teorema 43

4.5.4 Teorema da Composição Universal com Estado Conjunto 44

5 OBLIVIOUS TRANSFER BASEADO EM UM CANAL COM APAGAMENTOS GENERALIZADO 46
5.1 DEFINIÇÕES 48
5.1.1 Canal com Apagamentos Generalizado 48
5.1.2 Segurança baseada na Teoria da Informação para OT 49
5.1.3 Capacidade de *Oblivious Transfer* 50
5.1.4 Esquema de Codificação de Subconjuntos 51
5.1.5 *Hashing* Interativo 52
5.1.5.1 Segurança do *Hashing* Interativo 53
5.1.5.2 Protocolo de *Hashing* Interativo com Rodadas Constantes 54
5.2 O PROTOCOLO 56
5.2.1 Prova de Segurança Baseada em Teoria da Informação 60
5.2.2 Atingindo a Capacidade de *Oblivious Transfer* 65
5.2.2.1 Parte direta: 66
5.2.2.2 Parte inversa: 67
5.3 PROVA DE SEGURANÇA UNIVERSALMENTE COMPOSTA 71
5.3.1 Funcionalidades ideais 72
5.3.2 Construindo o Simulador 73
5.3.2.1 Alice Corrupta e Bob Honesto 74
5.3.2.2 Alice Honesta e Bob Corrupto 76
5.3.2.3 Alice e Bob Corruptos 77
5.3.2.4 Alice e Bob Honestos 78
5.3.3 Provando a Indistinguibilidade 78

6 PROTOCOLOS PARA ÁLGEBRA LINEAR DISTRIBUÍDA SEGURA 82
6.1 PRODUTO INTERNO 84
6.2 DETERMINANTE 86
6.2.1 Implementação 88
6.2.2 Funcionalidades Ideais 92
6.2.3 Prova de Segurança Universalmente Composta 93
6.2.3.1 Alice Corrupta e Bob Honesto 93
6.2.3.2 Alice Honesta e Bob Corrupto 95
6.2.3.3 Alice e Bob Corruptos 97
6.2.3.4 Alice e Bob Honestos 98
6.3 AUTOVALOR 98
6.3.1 Implementação 99
6.3.2 Funcionalidades Ideais 102
6.3.3 Prova de Segurança Universalmente Composta 103

6.3.3.1 Alice Corrupta e Bob Honesto 104
6.3.3.2 Alice Honesta e Bob Corrupto 105
6.3.3.3 Alice e Bob Corruptos ou Alice e Bob Honestos 107
6.4 AUTOVETOR 107
6.4.1 Implementação 107
6.4.2 Funcionalidades Ideais 111
6.4.3 Prova de Segurança Universalmente Composta 112
6.4.3.1 Alice Corrupta e Bob Honesto 112
6.4.3.2 Alice Honesta e Bob Corrupto 113
6.4.3.3 Alice e Bob Corruptos ou Alice e Bob Honestos 114

7 CONCLUSÕES E RECOMENDAÇÕES 115

REFERÊNCIAS BIBLIOGRÁFICAS 116

LISTA DE TABELAS

6.1 Funcionalidade do Produto Interno 84
6.2 Funcionalidade de Multiplicação de Matrizes 85
6.3 Funcionalidade de Determinante 88
6.4 Funcionalidade de Autovalores 100
6.5 Funcionalidade de Autovetores 108

LISTA DE FIGURAS

2.1 Inicializador Confiável 20

4.1 Execução do protocolo no modelo real 34
4.2 Execução do protocolo no modelo ideal 38
4.3 Simulação Caixa-Preta 41

5.1 One-out-of-Two Oblivious Transfer 46
5.2 Canal com Apagamentos Generalizado 49
5.3 Hashing Interativo 53
5.4 Resumo dos passos seguidos no protocolo para *String* OT 58

6.1 Fase de Inicialização e Fase de Execução do Protocolo de Produto de Matrizes e Seus Subprotocolos 86
6.2 Fases de inicialização e de Execução do Protocolo e dos Subprotocolos necessários para Calcular Determinante 89
6.3 Troca de Mensagens no Protocolo para Cálculo de Determinante 91
6.4 Troca de Mensagens no Protocolo para Cálculo de Autovalores 101
6.5 Troca de Mensagens no Protocolo para Cálculo de Autovetores 110

LISTA DE PROTOCOLOS

4.1 Protocolo de *Coin Tossing* 27
5.1 Protocolo de *Hashing* Interativo 55
5.2 Protocolo de *Oblivious Transfer* 59
6.1 Protocolo para Produto Interno 84
6.2 Protocolo para Produto de Matrizes 85
6.3 Protocolo para Determinante 90
6.4 Protocolo para Autovalor 101
6.5 Protocolo para Autovetor 109

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

AEP: Propriedade da Equipartição Assintótica

BC: Bit Commitment ou Comprometimento de Bit

$\mathbb{F}_q$: Corpo finito de ordem q

$\mathbb{F}_q^n$: Espaço de todas as n-tuplas com elementos em $\mathbb{F}_q$

$\mathbb{F}_q^{n\times n}$: Espaço de todas as matrizes $n \times n$ com elementos em $\mathbb{F}_q$

GEC: Generalized Erasure Channel ou Canal Com Apagamentos Generalizado

i.i.d.: Independente e Indenticamente Distribuído

IH: Interactive Hashing ou Hashing Interativo

ITI: Interactive Turing Machine Instance ou Instância de uma Máquina de Turing Interativa

ITM: Interactive Turing Machine ou Máquina de Turing Interativa

OT: Oblivious Transfer ou Transferência Inconsciente

PPT: Probabilístico e de tempo polinomial

$\mathbb{R}$: Conjunto dos números reais

$SL(\mathbb{F}_q)$: Conjunto de todas as matrizes $n \times n$ não-singulares com elementos em $\mathbb{F}_q$

TI: Trusted Initializer ou Inicializador Confiável

UC: Universal Composability ou Composabilidade Universal

$\mathbb{N}$: Conjunto dos números naturais, isto é, o conjunto dos números $\{1, 2, 3, ...\}$

1 INTRODUÇÃO

Em 1982 Andrew C. Yao (YAO, 1982) introduziu o conceito de computação distribuída segura estudando o Problema dos dois Milionários. Neste problema dois milionários desejam saber qual deles é o mais rico sem revelar a quantia exata do seu patrimônio. Generalizando, tem-se m participantes que desejam computar o valor de uma função $f(x_1, x_2, ..., x_m)$ onde cada participante conhece apenas a sua entrada x_i. Ao final da computação, deseja-se que cada participante não obtenha informações sobre as demais entradas da função além daquelas que podem ser inferidas através da saída da função e de sua própria entrada. Neste trabalho será focado as computações envolvendo apenas duas partes, isto é, computação de duas partes.

Claramente se os participantes confiam em alguém, isto é, existe uma parte confiável, eles podem simplesmente enviar as suas respectivas entradas para a parte confiável e receber a saída da função desejada. Contudo, é possível que os participantes atinjam o objetivo apenas trocando mensagens entre eles. Surpreendentemente, com uma primitiva criptográfica chamada *Oblivious Transfer* é possível implementar qualquer computação entre duas (ou múltiplas) partes de maneira incondicionalmente segura, isto é, um adversário com poder computacional infinito não pode quebrar o sistema (KILIAN, 1988). Então, usando *Oblivious Transfer* como um subprotocolo pode-se implementar qualquer computação multi-parte segura como Jogos Mentais, Mineração de Dados, Sistemas de Votação Eletrônica, etc.

Oblivious Transfer (OT) é uma primitiva em que o remetente transmite alguma informação ao destinatário sem saber precisamente qual informação foi recebida. Inicialmente existiam vários tipos de OT (RABIN, 1981; WIESNER, 1983), mas Crépeau em (CRÉPEAU, 1987) provou que todos eles são equivalentes. Neste trabalho será tratado a variante *one-out-of-two string oblivious transfer*. Nesta variante, o remetente tem como entrada duas strings $r_0, r_1 \in \{0,1\}^k$ e o destinatário tem como entrada um bit c. O remetente envia as suas duas strings de entrada e o destinatário usa o seu bit de entrada para receber uma das strings r_c. Também assegura-se que o remetente não sabe qual das duas strings de entrada foi recebida pelo destinatário e o destinatário recebe no máximo uma string do remetente.

O primeiro a demonstrar que os canais ruidosos eram úteis para implementar protocolos criptográficos seguros foi Wyner (WYNER, 1975) em 1975. Mais tarde, em 1988, Crépeau e Kilian (CRÉPEAU; KILIAN, 1988) provaram que canais ruidosos podem ser usados para implementar *oblivious transfer*. Neste trabalho implementa-se *oblivious transfer* utilizando um *Canal Com Apagamentos Generalizado* (GEC, do inglês *Generalized Erasure Channel*). Este canal pode ser visto como a combinação de um canal com apagamentos e um canal discreto sem memória uma vez que a entrada e a saída do canal são não correlacionados com uma certa probabilidade (comportando-se como um apagamento em um canal com apagamentos) e de outra forma a saída vai se comportar de acordo com um canal discreto sem memória associado. Observa-se que este canal é bem realístico uma vez que os bits transmitidos podem ser perdidos durante a transmissão (isto é, apagados) ou chegar com erros no destinatário de acordo com a distribuição do canal.

A capacidade do *oblivious transfer* foi introduzida por Nascimento e Winter em (NASCIMENTO; WINTER, 2008). Resumidamente, capacidade de *oblivious transfer* é uma medida de quão eficiente pode-se implementar *oblivious transfer* a partir de um canal ruidoso ou uma distribuição ruidosa. Então quando se fala de capacidade do *oblivious transfer* deve-se levar em consideração o tipo de canal ruidoso utilizado. Imai et al (IMAI; MOROZOV; NASCIMENTO, 2006) obteve a capacidade do *oblivious transfer* de Canais com Apagamentos e Ahlswede e Csiszár (AHLSWEDE; CSISZAR, 2007) obtiveram a capacidade do *oblivious transfer* para GEC no modelo adversarial passivo, isto é, quando os participantes do protocolo o seguem, mas desejam retirar informações da comunicação feita durante a execução do protocolo. Observa-se que Imai et al em (IMAI; NASCIMENTO; WINTER, 2003) introduziram a noção similar para capacidade de comprometimento.

Para provar que o protocolo de OT apresentado neste trabalho atinge a capacidade de *oblivious transfer* para canais com apagamentos generalizados utiliza-se a primitiva criptográfica *Hashing Interativo* (IH, do inglês *Interactive Hashing*). A solução para OT é baseado no protocolo para *oblivious transfer* a partir de canais com apagamentos proposto por Savvides (SAVVIDES, 2007), que foi construído nos resultados de (CRÉPEAU; SAVVIDES, 2006), e que emprega o *Interactive Hashing* de (CACHIN; CREPEAU; MARCIL, 1998) como subprotocolo. Contudo mostra-se que é possível usar o protocolo de IH de Ding et al (DING et al., 2007) obtendo um número constante de rodadas.

É usual definir um protocolo e analisar sua segurança a partir da uma cópia isolada. Contudo, este tipo de análise de segurança não garante que a execução de um determinado protocolo concorrentemente a outros permanecerá seguro. Para sanar esta dificuldade procura-se provar que os protocolos permanecem seguros quando compostos universalmente. A composibilidade universal (UC, do inglês *Universal Composability*) foi introduzido por Canetti em (CANETTI, 2000). Ele é um modelo em que se representa protocolos criptográficos e define a segurança que captura os requerimentos da tarefa em questão. Neste modelo, a segurança é mantida quando o protocolo é usado como um componente dentro de um sistema maior, isto é, a segurança dos protocolos é preservada quando o protocolo é executado com um número ilimitados outros protocolos (sendo estes protocolos arbitrários ou até múltiplas cópias do mesmo protocolo) concorrentemente.

Um dos resultados de segurança UC em (CANETTI, 2001) é que quando o adversário é limitado apenas na quantidade de tempo disponível (modelo padrão) não há implementação segura de alguns protocolos de duas partes sem hipóteses iniciais, como é o caso de protocolos de comprometimento.

Dado que álgebra linear é uma das principais ramos da matemática com aplicações em diversas áreas como engenharia, física, economia e assim por diante, o desenvolvimento de protocolos criptográficos que realizam as principais ferramentas da álgebra linear se torna bastante atrativo. Em (DOWSLEY et al., 2010) Dowsley et al demonstraram um protocolo criptográfico universalmente composto para produto interno e o utilizaram em uma construção de um protocolo para sistemas lineares que também era universalmente composto. Seguindo a mesma linha do trabalho deles e utilizando o resultado para o produto interno, apresenta-se neste trabalho protocolos de duas partes para computar determinante, autovalores e autovetores. É provado neste trabalho que os protocolos propostos são seguros no modelo de composibilidade universal.

Em especial, todos os protocolos de álgebra linear são construídos no modelo de criptografia baseada em *commodities*. Esse modelo foi inicialmente proposto por Beaver em (BEAVER, 1997). Ele estipula que existe um Inicializador Confiável (TI, do inglês *Trusted Initializer*) que distribui dados correlacionados para as partes que irão executar um determinado protocolo no início, antes da execução do protocolo. Dessa forma, é possível à ambas as partes executar a tarefa desejada preservando a privacidade das entradas das partes. A principal vantagem do uso de um inicializador confiável reside no fato de que o inicializador pode calcular tarefas complexas para as partes. Assim, as

partes apenas precisam executar as tarefas mais simples, o que simplifica os protocolos.

1.1 ORGANIZAÇÂO DO TRABALHO

Preliminares. No capítulo 2 apresenta-se alguns conceitos básicos e resultados auxiliares. Como a definição de Entropia, Informação Mútua, Informação Estatística; Sequências Típicas e Sequências Conjuntamente Típicas e Extratores Fortes. Além disso, apresenta-se uma discussão extremamente breve do modelo criptográfico que postula a presença de um inicializador confiável que será utilizado nos protocolos de álgebra linear distribuída segura.

Computação de Duas Partes Segura. No capítulo 3 apresenta-se uma breve discussão sobre computação de duas partes segura, mostrando as definições de protocolos de duas partes e adversários; segurança no modelo malicioso; e composição de protocolos.

Composibilidade Universal. No capítulo 4 apresenta-se uma análise detalhada do modelo UC e os principais resultados como o Teorema de Composição Universal.

***Oblivious Transfer* Baseado em Um Canal Com Apagamentos Generalizado.** No capítulo 5 será apresentado a primeira contribuição deste trabalho, que é apresentar um protocolo de *oblivious transfer* baseado em um canal com apagamentos generalizado que atinge a capacidade de OT no modelo malicioso e é seguro no modelo de composibilidade universal. Estritamente falando, apresenta-se a primitiva *Oblivious Transfer*, a definição de segurança de *string* OT baseada em teoria da informação e de capacidade de OT; apresenta-se o protocolo que atinge a capacidade do OT para GEC; e demonstra-se que o protocolo de OT segue a definição de segurança baseada em teoria da informação. Ao final do capítulo, tem-se a segunda contribuição deste trabalho que é demonstrar que o protocolo de OT apresentado na seção anterior segue a definição de segurança UC;

Protocolos para Álgebra Linear Distribuída Segura. No capítulo 6 tem-se a terceira contribuição deste trabalho que é apresentar protocolos para calcular ferramentas de álgebra linear que sejam seguros quando executados concorrentemente com protocolos arbitrários. Especificamente, neste capítulo apresenta-se um pequena revisão dos conceitos utilizados referentes ao produto interno, determinante, autovalor e autovetor; e apresenta-se os protocolos para cada um deles. Ao final do capítulo, tem-se a última

contribuição deste trabalho, provando que os protocolos apresentados na seção anterior são UC seguros.

Conclusões e Recomendações. Finalmente no capítulo 7 obtém-se as conclusões a que se chegaram ao final deste trabalho e a exposição dos problemas abertos.

2 PRELIMINARES

Este capítulo apresenta os conceitos básicos e resultados auxiliares usados nos demais capítulos deste trabalho.

2.1 NOTAÇÃO

Será utilizado letras caligráficas, tais como $\mathcal{A}, \mathcal{B}, \mathcal{X}, \mathcal{Y}$, para os domínios de variáveis aleatórias e outros conjuntos. $|\mathcal{X}|$ denotará a cardinalidade do conjunto $\mathcal{X}$. As variáveis aleatórias, matrizes e algoritmos serão representadas com letras maiúsculas; e uma realização de uma variável aleatória com letra minúscula. Dada uma variável aleatória X sobre $\mathcal{X}$, a distribuição de probabilidade é dada por $P_X : \mathcal{X} \to [0, 1]$ com

$$\sum_{x \in \mathcal{X}} P_X(x) = 1. \tag{2.1}$$

Dado a distribuição de probabilidade conjunta $P_{XY} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$, tem-se

$$P_X(x) := \sum_{y \in \mathcal{Y}} P_{XY}(x, y) \tag{2.2}$$

$$P_{X|Y}(x|y) := \frac{P_{XY}(x, y)}{P_Y(y)} \tag{2.3}$$

onde a equação 2.2 denotará a distribuição de probabilidade marginal e a equação 2.3 denotará a distribuição de probabilidade se $P_Y(y) \neq 0$. $X \in_R \mathcal{X}$ denotará a variável aleatória X distribuída uniformemente sobre $\mathcal{X}$.

Se a e b são dois bits, $a \oplus b$ denotará o ou-exclusivo entre eles. Se a e b são duas strings (cadeias de bits) de mesmo comprimento, $a \oplus b$ denotará o ou-exclusivo bit-a-bit entre eles. Todos os logaritmos deste trabalho serão considerados base 2.

Denota-se $[n]$ como o conjunto $\{1, 2, ..., n\}$ e $\binom{[n]}{l}$ como o conjunto de todos os subconjuntos $\mathcal{K} \subseteq [n]$, onde $|\mathcal{K}| = l$. Para $X^n = (X_1, X_2, ..., X_n)$ e $\mathcal{S} \subset [n]$, $X^{\mathcal{S}}$ denota a restrição de X^n para as posições no subconjunto $\mathcal{S}$. Similarmente, $\mathcal{R}^{\mathcal{S}}$ denota o subconjunto de $\mathcal{R}$ formado pelas posições determinadas por $\mathcal{S}$.

Durante as provas de segurança no *framework* de composibilidade universal, será usado um apóstrofo ($'$) em todas as variáveis no ambiente simulado.

Denota-se $\mathbb{F}_q$ como o corpo finito de ordem q; $\mathbb{F}_q^n$ como o espaço de todas as n-tuplas com elementos em $\mathbb{F}_q$; $\mathbb{F}_q^{n\times n}$ como o espaço de todas as matrizes $n \times n$ com elementos em $\mathbb{F}_q$; e $SL(\mathbb{F}_q)$ como o conjunto de todas as matrizes $n \times n$ não-singulares com elementos em $\mathbb{F}_q$.

2.2 MÁQUINAS DE TURING INTERATIVAS

Os protocolos, as partes que executam o protocolo e qualquer outra entidade que interage com os protocolos, como adversários por exemplo, são formalmente modelados como máquinas de Turing interativas. Em criptografia, geralmente utiliza-se as definições de máquinas de Turing interativas e interação entre pares de máquinas de Turing interativas de Golwasser, Micali e Rackoff (GOLDWASSER; MICALI; RACKOFF, 1989) e de Goldreich (GOLDREICH, 2000a). A seguir, apresenta-se a definição de máquina de Turing interativa de (GOLDREICH, 2000a, Definição 4.2.1).

Definição 1. *Uma* ***máquina de Turing interativa*** *(ITM -* Interactive Turing Machine*) é uma máquina de Turing com as seguintes fitas:*

1. *Uma fita de entrada de leitura;*
2. *Uma fita aleatória de leitura;*
3. *Uma fita de trabalho de leitura e escrita;*
4. *Uma fita de comunicação de entrada de leitura que se movimenta em apenas uma direção;*
5. *Uma fita de comunicação de saída de escrita que se movimenta em apenas uma direção; e*
6. *Uma fita de* switch *de leitura e escrita de apenas uma célula.*

Cada ITM possui um único bit de identidade. Quando o conteúdo da fita de switch *é igual à identidade da máquina, diz-se que a ITM está ativa em uma determinada configuração. De outro modo, diz-se que a ITM está ociosa e, neste caso, o estado da máquina, o posicionamento das cabeças em cada fita e os conteúdos das fitas de escrita não são modificados.*

A definição anterior permite separar a entrada e saída de um programa da comunicação com outras partes. Cabe salientar que a relação de entrada e saída está relacionado com a funcionalidade de um programa, já a comunicação entre as partes deve ser tratado pelo protocolo que implementa a funcionalidade ao invés de ser tratado pela funcionalidade. Maiores discussões sobre esse tópico serão apresentadas no Capítulo 3 de computação multiparte segura.

O principal uso do conceito de ITM neste trabalho será no *framework* de composibilidade universal, pois ele é constuído com base em tais máquinas. Contudo, será seguido a apresentação feita por Canetti em (CANETTI, 2000) que extende a abordagem de (GOLDWASSER; MICALI; RACKOFF, 1989; GOLDREICH, 2000a), adicionando alguma sintaxe com intenção de facilitar a formalização de contextos multipartes multiexecução e cujo número de máquinas interagindo não é limitado a priori. De fato, a principal modificação se refere às permissões de leitura e escrita das fitas. Considera-se que uma fita é "externamente escrita"(EW) se ela pode ser escrita por uma ITM e lida por outra. Além disso, considera-se que em todas as fitas que são EW a cabeça de leitura pode se mover em apenas uma direção. Ademais, são consideradas EW todas as fitas, com exceção da fita de saída e da fita de ativação que são escritas apenas pela própria ITM. A fita aleatória pode ser escrita externamente apenas durante a inicialização da ITM. Um outro detalhe é que opta-se por chamar a fita de comunicação de saída por fita de saída de sub-rotina. A ideia é deixar que as comunicações através das fitas de entrada, de saída e de saída de sub-rotina são confiáveis e a comunicação através da fita de comunicação de entrada não é confiável. Mais informações sobre as escritas nas fitas e confiabilidade das comunicações será dado no Capítulo 4 sobre composibilidade universal. A seguir, apresenta-se a definição de ITM usada no *framework* UC.

Definição 2. *Uma máquina de Turing Interativa (ITM) M é uma máquina de Turing com as seguintes fitas:*

1. *uma* fita EW de identidade *que especifica a identidade de M. Mais precisamente, a identidade consiste de um identificador de sessão (*SID*) de M e o identificador da parte (*PID*) de M;*
2. *uma* fita EW de parâmetro de segurança *que especifica o parâmetro de segurança do protocolo.*
3. *uma* fita EW de entrada *que possui a entrada privada de M;*
4. *uma* fita aleatória *que possui os bits aleatórios de entrada de M;*

5. *uma* fita de ativação de escrita e leitura *que possui um único bit. Sem perda de generalidade, é dito que quando a fita de ativação possui bit 1, M está ativo. Quando ativado, M segue as configurações pré-estabelecidas até que alcance um estado de espera ou um estado de parada. Quando M alcança um estado de espera, ele escreve o bit 0 na fita de ativação e espera até sua próxima ativação. No caso de M alcançar um estado de parada, ele escreve o bit 0 na fita de ativação e não executa nada nas futuras ativações.*

6. *uma* fita de trabalho de leitura e escrita *que é usada para as computações internas e privadas;*

7. *uma* fita EW de comunicação de entrada *que recebe as mensagens de outras partes. Cada mensagem tem dois campos: o campo com a identidade do emissor e o campo de conteúdo;*

8. *uma* fita EW de saída de sub-rotina *que possui as mensagens que serão enviadas para outras partes. Cada mensagem nesta fita possui também os dois campos: o campo com a identidade do destinatário e o campo de conteúdo; e*

9. *uma* fita de saída *que possui a saída privada de M.*

2.3 INDISTINGUIBILIDADE

Existem três variações padrões para indistinguibilidade: computacional, estatística ou perfeita. Cada uma dessas variações definem noções de segurança computacional, de segurança estatística ou de segurança perfeita, respectivamente. Os conceitos de indistinguibilidade surgem diretamente da teoria da probabilidade como será visto a seguir.

Se duas distribuições P_X e P_Y sobre o mesmo domínio $\mathcal{V}$ são idênticas, então elas serão consideradas perfeitamente indistinguíveis, $X \equiv Y$ (*indistinguibilidade perfeita*). Relaxando essa condição, tem-se que duas distribuições P_X e P_Y são estatisticamente indistinguíveis se a distância estatística entre as duas distribuições é negligível.

Definição 3. *A* distância estatística *entre duas distribuições de probabilidade P_X e P_Y sobre o mesmo domínio $\mathcal{V}$ é dado pela equação 2.4.*

$$\mathsf{SD}(P_X, P_Y) := \frac{1}{2} \sum_{v \in \mathcal{V}} |P_X(v) - P_Y(v)|. \tag{2.4}$$

Com a noção de distância estatística, tem-se que uma variável aleatória X sobre $\mathcal{V}$ é *ϵ-próxima de uniforme com respeito a Y* se $\mathsf{SD}(P_{XY}, P_U P_Y) \leq \epsilon$, onde P_U é uma distribuição uniforme sobre $\mathcal{V}$.

Uma função $f(n) : \mathbb{N} \to \mathbb{R}$ é negligível se para todo polinômio $p(n)$ existir um n_0 tal que para todo $n \geq n_0$, segue a equação 2.5.

$$f(n) < \frac{1}{p(n)} \tag{2.5}$$

Então, formaliza-se a seguir a *indistinguibilidade estatística* tomando uma variável aleatória como representante da distribuição de probabilidade associada a ela.

Definição 4. *Duas sequências $X := \{X_n\}_{n\in\mathbb{N}}$ e $Y := \{Y_n\}_{n\in\mathbb{N}}$ de variáveis aleatórias são* estatisticamente indistinguíveis*, denotado por $X \overset{s}{\approx} Y$, se existir uma função negligível $\epsilon(\cdot)$ tal que para todo $n \in \mathbb{N}$, segue a equação 2.6*

$$\mathsf{SD}(X_n, Y_n) < \epsilon(n). \tag{2.6}$$

Finalmente, define-se a *indistinguibilidade computacional*, onde um algoritmo eficiente "testa"se dois conjuntos de variáveis aleatórias possuem a mesma distribuição ou não. Formalmente:

Definição 5. *Duas sequências $X := \{X_n\}_{n\in\mathbb{N}}$ e $Y := \{Y_n\}_{n\in\mathbb{N}}$ de variáveis aleatórias são* computacionalmente indistinguíveis*, denotado por $X \overset{c}{\approx} Y$, se para todo algoritmo (distinguidor) não-uniforme, probabilístico e de tempo polinomial D existir uma função negligível $\epsilon(\cdot)$ tal que para todo $n \in \mathbb{N}$, segue a equação 2.7*

$$|Pr[D(X_n)] = 1 - Pr[D(Y_n)] = 1| < \epsilon(n). \tag{2.7}$$

2.4 COMPROMETIMENTO DE BIT

Comprometimento de bit é uma primitiva que envolve dois jogadores, o emissor(ou *committer*) e o receptor (ou *verifier*). Nesta primitiva, o emissor deseja enviar um bit dentro de um envelope selado para o receptor. Além disso, o receptor não pode ver o que está dentro do envelope até que ele possa abri-lo. Quando ambos as partes estiverem de acordo, o remetente envia as informações pertinentes para que o receptor possa abrir o envelope. Após aberto o envelope o receptor aceita o bit se todas as informações de abertura estiverem de acordo com o envelope recebido inicialmente.

Requer-se que o emissor não possa abrir o envelope e fazer o receptor acreditar que o valor do bit é diferente daquele que ele enviou inicialmente.

Formalmente, em um esquema de comprometimento de bit, a entrada do emissor é um bit e o receptor não possui nenhuma entrada. Ao fim do protocolo, o receptor aceita o bit revelado ou não. O esquema possui duas fases: a de comprometimento e a de revelação.

Na fase de comprometimento, o emissor faz uma computação com o bit de entrada de tal forma que o resultado desta computação seja aleatória do ponto de vista do receptor e envia o resultado desta computação para o receptor junto com alguma informação para a verificação posterior. Pode-se dizer então que nesta fase, o receptor não aprende nada sobre o bit b através do comprometimento $[b]$.

Na fase de revelação, o emissor envia a informação necessária para abrir o comprometimento $|b|$. Esta informação pode permitir ao receptor fazer a computação inversa ou fazer a mesma computação do emissor e comparar os resultados. Após o receptor usar a informação de abertura, ele usa a informação de verificação para conferir se o bit b' que ele recuperou é consistente com o comprometimento recebido inicialmente.

Um esquema de comprometimento de bit deve ter três propriedades para ser considerado seguro:

- *Correctness*: Se tanto o emissor quanto o receptor forem honestos, o esquema não abortará e ao final o emissor aceitará o bit recuperado.
- *Hiding*: Se o emissor for honesto, o receptor não aprenderá coisa alguma sobre o bit comprometido antes da fase de abertura.
- *Binding*: Se o receptor for honesto, o emissor não é capaz de abrir um bit $\bar{b} \neq b$ sem ser pego com alta probabilidade.

As noções a seguir serão necessárias para a construção e análise do protocolo de *oblivious transfer* que atinge a capacidade de OT de canais com apagamentos generalizado no modelo malicioso.

2.5 ENTROPIA, INFORMAÇÃO MÚTUA E ESTATÍSTICA

Segue as definições formais de entropia, informação mútua e informação estatística. De maneira geral, a entropia mede a quantidade de informação de uma variável aleatória, ou dito de outra maneira, a entropia é o número de bits necessários para descrever a variável aleatória. A entropia condicional de X dado Y, onde X e Y são variáveis aleatórias, mede a quantidade de informação sobre X que permanece após observar Y.

A informação mútua entre X e Y mede a quantidade de informação que X e Y compartilham, isto é, ela mede o quanto de informação sobre X a variável aleatória Y contém. A informação mútua, por exemplo, diz quantos bits podem ser transmitidos em um canal ruidoso. Observe que informação mútua e entropia condicional são conceitos parecidos, mas não são os mesmos. A entropia condicional diz a quantidade de informação de X que não está correlacionado com Y, enquanto a informação mútua mede exatamente a correlação entre X e Y.

Finalmente, a informação estatística mede a probabilidade de uma variável aleatória X possuir informação sobre a variável aleatória Y, mais precisamente a informação estatística mede o quão próximo a distribuição de três variáveis aleatórias X, Y e Z estão de uma cadeia de Markov. Pode-se dizer que a informação mútua e a informação estatística são muito próximas. A diferença reside na medida de distância em que a informação mútua e a informação estatística se baseiam: a informação mútua se baseia na entropia condicional e a informação estatística se baseia na distância estatística.

2.5.1 Entropia

A *entropia de Shannon* de uma variável aleatória discreta X é dada pela equação 2.8:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x), \tag{2.8}$$

e a *entropia condicional de Shannon* de uma variável aleatória discreta X dado uma variável aleatória Y é dada pela equação 2.9

$$H(Y|X) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x). \tag{2.9}$$

Alguns resultados relevantes sobre entropia são a Regra da Cadeia para Entropia dada pela equação 2.10

$$H(X, Y) = H(X) + H(Y|X), \tag{2.10}$$

a versão condicional da regra da cadeia dada pela equação 2.11

$$H(X,Y|Z) = H(X|Z) + H(Y|X,Z), \tag{2.11}$$

e o resultado que mostra que condicionar reduz a entropia dada pela inequação 2.12

$$H(X|Y) \leq H(X) \tag{2.12}$$

2.5.2 Informação Mútua

A *informação mútua* de uma variável aleatória discreta X dado Y é dada pela equação 2.13

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = H(X) - H(X|Y), \tag{2.13}$$

a *informação mútua condicional* das variáveis aleatórias discretas X, Y dado Z é dada pela equação 2.14

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z). \tag{2.14}$$

Um resultado relevante sobre a informação é a regra da cadeia para informação mútua dada pela equação 2.15

$$I(X,Y,Z;V) = I(X;V) + I(Y;V|X) + I(Z;V|XY). \tag{2.15}$$

2.5.3 Informação Estatística

Em 2008, Crépeau e Wullschleger (CRÉPEAU; WULLSCHLEGER, 2008) introduziram a definição de Informação Estatística. Esta medida foi introduzida inicialmente com a intenção de obter a probabilidade com que um adversário reúne qualquer informação.

Definição 6. *A* informação estatística *entre as variáveis aleatórias X e Y dado a variável aleatória Z é definida pela equação 2.16*

$$I_S(X;Y|Z) = \mathsf{SD}(P_{XYZ}, P_Z P_{X|Z} P_{Y|Z}), \tag{2.16}$$

onde as distribuições de probabilidade P_X, P_Y e P_Z são tomadas sobre o mesmo domínio $\mathcal{V}$.

2.6 SEQUÊNCIAS TÍPICAS E CONJUNTAMENTE TÍPICAS

Esta seção segue os capítulos 3 e 7 do livro (COVER; THOMAS, 2006). A propriedade de equipartição assintótica (AEP) estabelece que para variáveis aleatórias i.i.d. (independentes e identicamente distribuídas) X_i tal que $-\frac{1}{n}\log p(X_1, X_2, ..., X_n)$ converge em probabilidade para a entropia $H(X)$. Isto significa que

$$\forall \epsilon > 0, \Pr\left\{\left|-\frac{1}{\mathrm{n}}\mathrm{logp}(\mathrm{X_1}, \mathrm{X_2}, ..., \mathrm{X_n}) - \mathrm{H(X)}\right| > \epsilon\right\} \longrightarrow 0. \tag{2.17}$$

Esse resultado permite dividir o conjunto de todas as sequências X^n em dois conjuntos: o conjunto das *sequências típicas*, onde a entropia empírica é próxima da verdadeira entropia, e o conjunto das *sequências não típicas*. O poder do AEP permanece no fato de que o conjunto de sequências típicas é um conjunto altamente representativo. Em outras palavras, a probabilidade de que a sequência pertença ao conjunto de sequências típicas é próximo de um. Vale a pena salientar que o conjunto típico é representativo relativo à probabilidade. O número de sequências típicas é um conjunto ligeiramente menor. Veja o Teorema 1.

Precisamente, o conjunto típico é definido da seguinte forma:

Definição 7. *O* conjunto típico a_ϵ^n *com respeito à função massa de probabilidade* $p(x)$ *é o conjunto das sequências* $(x_1, x_2, ..., x_n)$ *com a propriedade*

$$2^{-n(H(x)+\epsilon)} \leq p(x_1, x_2, ..., x_n) \leq 2^{-n(H(x)-\epsilon)}. \tag{2.18}$$

Com essa definição, pode-se estabelecer o seguinte teorema que mostra algumas propriedades das sequências típicas:

Teorema 1.

1. $(x_1, x_2, ..., x_n) \in a_\epsilon^n \Rightarrow H(X) - \epsilon \leq -\frac{1}{n}\log p(x_1, x_2, ..., x_n) \leq H(X) + \epsilon$;
2. $\Pr\{a_\epsilon^n\} > 1 - \epsilon$ *para n suficiente grande;*
3. $|a_\epsilon^n| \leq 2^{n(H(X)+\epsilon)}$; *e*
4. $|a_\epsilon^n| \geq (1-\epsilon)2^{n(H(X)-\epsilon)}$ *para n suficiente grande.*

Com as definições já estabelecidas, pode-se estender o conceito de sequências típicas para sequências conjuntamente típicas. O conjunto A_ϵ^n de sequências conjuntamente típicas é o conjunto de n-sequências x^n e y^n tal que as entropias empíricas são ϵ-próximas da verdadeira entropia e verdadeira entropia conjunta.

Definição 8. *O conjunto A_ϵ^n de sequências conjuntamente típicas $\{(x^n, y^n)\}$ com respeito à distribuição $p(x, y)$ é:*

$$\begin{aligned} A_\epsilon^n &= \{(x^n, y^n) \in \mathcal{X}^n \mathrm{x} \mathcal{Y}^n : \left|-\frac{1}{n} logp(x^n) - H(X)\right| < \epsilon, \\ &\quad \left|-\frac{1}{n} logp(y^n) - H(Y)\right| < \epsilon, \left|-\frac{1}{n} logp(x^n, y^n) - H(X,Y)\right| < \epsilon\}. \quad (2.19) \end{aligned}$$

onde

$$p(x^n, y^n) = \prod_{i=1}^{n} p(x_i, y_i)$$

Similarmente ao caso de sequências típicas, pode-se estabelecer o seguinte teorema:

Teorema 2. *Seja (X^n, Y^n) sequências de comprimento n feitas i.i.d. de acordo com*

$$p(x^n, y^n) = \prod_{i=1}^{n} p(x_i, y_i). \quad (2.20)$$

Então:

1. $\mathrm{Pr}((\mathrm{X^n}, \mathrm{Y^n}) \in \mathrm{A_\epsilon^n}) \to 1$ *enquanto* $n \to \infty$*;*
2. *O número de sequências x^n típica conjuntamente com y^n é $|A_\epsilon^n| \leq 2^{n(H(X,Y)+\epsilon)}$;*
3. *Se $(\tilde{X}^n, \tilde{Y}^n)$ são independentes das mesmas probabilidades marginais de X, Y, isto é $p(\tilde{x}) = p(x)$ e $p(\tilde{y}) = p(y)$, então:*

$$Pr((\tilde{X}^n, \tilde{Y}^n) \in A_\epsilon^n) \leq 2^{-nI(X;Y)-3\epsilon}. \quad (2.21)$$

Ao leitor interessado na prova dos Teoremas 1 e 2, é fortemente recomendado a leitura dos capítulos 3 e 7 do livro (COVER; THOMAS, 2006).

Teorema 3. *Seja (X^n, Y^n) sequências de comprimento n feitas i.i.d. de acordo com*

$$p(x^n, y^n) = \prod_{i=1}^{n} p(x_i, y_i).$$

Então o número de sequências x^n conjuntamente típicas com y^n é $|A_\epsilon^n| \leq 2^{n(H(Y|X)+\epsilon)}$.

Demonstração: A prova segue do item 2 do último teorema.

$$\begin{aligned} 1 &= \sum p(x^n, y^n) \\ &\geq \sum_{A_\epsilon^n} p(x^n, y^n) \\ &\geq \sum_{A_\epsilon^n} p(y^n|x^n) \\ &\geq |A_\epsilon^n| 2^{-n(H(Y|X)+\epsilon)} \end{aligned}$$

Então, $|A_\epsilon^n| \leq 2^{n(H(Y|X)+\epsilon)}$. □

Definições similares para sequências típicas e sequências conjuntamente típicas são encontradas no livro de Csiszár e Körner (CSISZáR; KÖRNER, 1981) e serão úteis na prova do Lema 5 do Capítulo 5. A Definição 7 de sequências típicas pode ser vista da seguinte maneira:

Definição 9. *Para uma distribuição de probabilidade p em $\mathcal{X}$ e $\epsilon > 0$ as sequências ϵ-típicas formam o conjunto*

$$a_{p,\epsilon}^n = \{x^n \in \mathcal{X}^n : \forall x \in \mathcal{X} | N(x|x^n) - np(x)| \leq \epsilon n, p(x) = 0 \Rightarrow N(x|x^n) = 0\},$$

onde $N(x|x^n)$ denota o número de símbolos x na string *x^n.*

O *tipo* de x^n é a distribuição de probabilidade dada pela equação 2.22

$$p_{x^n}(x) = \frac{1}{n} N(x|x^n). \tag{2.22}$$

Então, tem-se a seguinte implicação:

$$x^n \in a_{p,\epsilon}^n \Rightarrow |p_{x^n}(x) - p(x)| \leq \epsilon, \forall x \in \mathcal{X}. \tag{2.23}$$

Similarmente, a definição equivalente à Definição 8 é a seguinte:

Definição 10. *Considere um canal $W : \mathcal{X} \to \mathcal{Y}$ e uma* string *de entrada $x^n \in \mathcal{X}^n$. Para $\epsilon > 0$, as sequências ϵ-condicionalmente típicas formam o conjunto*

$$\begin{aligned} A_{W,\epsilon}^n &= \{y^n : \forall x \in \mathcal{X}, y \in \mathcal{Y} : |N(xy|x^n y^n) - nW(y|x)p_{x^n}(x)| \leq \epsilon n, \\ &\quad W(y|x) = 0 \Rightarrow N(xy|x^n y^n) = 0\} \end{aligned}$$

Como as Definições 7 e 8 são equivalentes às Definições 9 e 10, as propriedades que delas advêm são as mesmas.

2.7 EXTRATORES

Extratores foram introduzidos em (NISAN; ZUCKERMAN, 1996). Eles são ferramentas que agem extraindo aleatoriedade de uma fonte com uma distribuição arbitrária usando um pequeno número de bits verdadeiramente aleatórios.

Para definir extratores, precisa-se dos conceitos de distância estatística (Definição 3), δ-fonte e *min-entropy*.

Definição 11. *Uma distribuição D em $\{0,1\}^n$ é chamado de uma δ-fonte se a desigualdade a seguir for satisfeita*

$$D(x) \leq 2^{-\delta n}, \forall x \in \{0,1\}^n. \tag{2.24}$$

A seguinte definição, de (NISAN; ZUCKERMAN, 1996), estabelece que de uma entrada com n bits e usando r bits verdadeiramente aleatórios, a função Ext dá como resultado m bits com distribuição ϵ-próxima da distribuição uniforme.

Definição 12. *Seja $Ext : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$. Ext é chamado de um (δ, ϵ)-*extrator *se para toda δ-fonte D, a distribuição de $Ext(x,y) \circ y$ induzida pela escolha de x a partir de D e y uniformemente em $\{0,1\}^r$ tem uma distância estatística similar a ϵ a partir de uma distribuição uniforme em $\{0,1\}^m \times \{0,1\}^r$.*

Entretanto, neste trabalho será usado uma versão um pouco diferente de extratores (DODIS et al., 2008). Esta definição não usa entropia de Shannon, ao invés usa-se *min-entropy*. Intuitivamente, se uma distribuição tem *min-entropy* k, ele é *"ao menos tão aleatório"* quanto uma distribuição uniforme de strings de k bits. [1]

A seguir define-se estritamente o que vem a ser uma *min-entropy*.

Definição 13. *Para um alfabeto finito $\mathcal{X}$, a* min-entropy *de uma variável aleatória $X \in \mathcal{X}$ é definida pela equação 2.25*

$$H_\infty(X) = \min_x \log(1/P_X(x)). \tag{2.25}$$

Sua versão condicional, definida sobre $\mathcal{Y}$ com alfabeto finito é dada pela equação 2.26

$$H_\infty(X|Y) = \min_y H_\infty(X|Y = y). \tag{2.26}$$

[1] De fato, quando se trabalha com min-entropy está sendo considerado o pior caso. Além disso, é impossível extrair aleatoriedade de distribuições que não são próximas de ter alta *min-entropy*. (Veja (SHALTIEL, 2002))

Definição 14 (Extratores Fortes Aleatórios). *Seja* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ *uma função de tempo polinomial probabilístico que usa* r *bits de aleatoriedade. Um* Ext *é um eficiente* $(n, m, l, \epsilon)-$strong extractor *se para toda distribuição de probabilidade* P_W *com* $\mathcal{W} = \{0,1\}^n$ *e* $H_\infty(W) \geq m$*, tem-se que* $\mathsf{SD}(P_{\mathsf{Ext}(W;U_r),U_r}, P_{U_l}P_{U_r}) \leq \epsilon$.

Observe que uma δ-fonte tem min-entropy $\delta n \geq m$.

Em (RADHAKRISHNAN; TA-SHMA, 2000) tem-se o resultado de que extratores fortes podem extrair no máximo $l = m - 2\log(\epsilon^{-1}) + O(1)$ bits de bits quase aleatórios, e seu limite ótimo é atingido por uma Função de Hash 2-Universal (CARTER; WEGMAN, 1979) que será definida a seguir.

Definição 15 (Função de Hash 2-Universal). *Uma classe* $\mathcal{G}$ *de funções* $\mathcal{A} \to \mathcal{B}$ *é* 2-universal *se, para qualquer distintos* $x_1, x_2 \in \mathcal{A}$*, a probabilidade de que* $g(x_1) = g(x_2)$ *é no máximo* $|\mathcal{B}|^{-1}$ *quando* g *é escolhido uniformemente aleatório de* $\mathcal{G}$.

O *Leftover-Hash Lemma* (similarmente o Lema de Amplificação de Privacidade) (IMPAGLIAZZO; LEVIN; LUBY, 1989; HASTAD et al., 1999; BENNETT; BRASSARD; ROBERT, 1988; BENNETT et al., 1995; DODIS et al., 2008) garante que as funções de hash universal permite extrair $l = m - 2\log(\epsilon^{-1}) + 2$ bits.

Lema 1 (Leftover-Hash Lemma). *Assume que uma classe* $\mathcal{G}$ *de funções* $G : \{0,1\}^n \to \{0,1\}^l$ *é 2-universal. Então para* G *selecionado uniformemente aleatório de* $\mathcal{G}$ *tem-se que*

$$\mathsf{SD}(P_{G(W),G}, P_{U_l}P_G) \leq \frac{1}{2}\sqrt{2^{-H_\infty(W)}2^l}. \tag{2.27}$$

Em particular, funções de hash universal são $(n, m, l, \epsilon)-$strong extractors *sempre que*

$$l \leq m - 2\log(\epsilon^{-1}) + 2. \tag{2.28}$$

2.8 LIMITE DE CHERNOFF E DESIGUALDADE DE FANO

Lema 2 (Limite de Chernoff (CHERNOFF, 1952)). *Para variáveis aleatórias i.i.d* $X_1, \ldots, X_n$ *com* $0 \leq X_n \leq 1$ *e com esperança* $E(X_n) = p$*:*

$$Pr\left\{\frac{1}{N}\sum_{n=1}^{N} X_n \geq (1+\eta)p\right\} \leq \exp\left(-N\frac{p\eta^2}{2\ln 2}\right), \tag{2.29}$$

$$Pr\left\{\frac{1}{N}\sum_{n=1}^{N} X_n \leq (1-\eta)p\right\} \leq \exp\left(-N\frac{p\eta^2}{2\ln 2}\right). \tag{2.30}$$

Teorema 4 (Desigualdade de Fano)**.** *Para qualquer estimador* $\hat{X}$ *tal que* $X \to Y \to \hat{X}$ *com* $P_e = Pr(\hat{X} \neq X)$*, tem-se:*

$$H(P_e) + P_e \log|\mathcal{X}| \geq H(X|\hat{X}) \geq H(X|Y). \tag{2.31}$$

2.9 CRIPTOGRAFIA BASEADA EM COMMODITIES

Esta seção discutirá brevemente o modelo criptográfico baseado em *commodities* que será utilizado nos protocolos para calcular álgebra linear distribuída segura apresentados no Capítulo 6.

Esse modelo foi introduzido na literatura por Beaver em (BEAVER, 1997) para construir um protocolo de *oblivious transfer*. Ele estipula que as partes que executam um protocolo criptográfico multiparte utilizam-se de dados (os *commodities* que dá nome ao modelo) recebidos de servidores confiáveis antes do protocolo começar. Dito de outra forma, o servidor, também conhecido como Inicializador Confiável (TI - do inglês *Trusted Initializer*), envia dados para as partes antes de um determinado protocolo começar e essas partes podem usar os dados recebidos durante a execução do protocolo.

Os dados recebidos pelas partes podem ser resultados de funções complexas calculadas pelo inicializador confiável. Esta característica permite que os protocolos formulados neste modelo sejam mais simples. Portanto, esse modelo se torna uma alternativa eficiente para se projetar protocolos multiparte seguros.

A figura 2.1 exemplifica a execução de um protocolo por duas partes, Alice e Bob, onde o inicializador confiável escolhe dois conjuntos de dados de tal forma que ambos os conjuntos sejam correlatos e entrega um conjunto de dados μ_A para Alice e o outro conjunto de dados μ_B para Bob. Observe que Alice inicia o protocolo com a sua entrada privada X e o dado recebido μ_A. De forma similar, Bob inicia o protocolo com a sua entrada privada Y e o dado recebido μ_B.

Apesar da presença de uma terceira parte confiável parecer ser uma hipótese forte, em várias situações essa suposição pode ser facilmente atendida. É o caso, por exemplo, de um centro de distribuição de chaves. Ele pode ser visto como um inicializador confiável que distribui pares de chaves de forma que a chave privada fica na posse de uma das partes e a chave pública é distribuída entre as demais partes que executaram

um determinado protocolo. A partir desses dados compartilhados as partes podem executar o protocolo com comunicação privada.

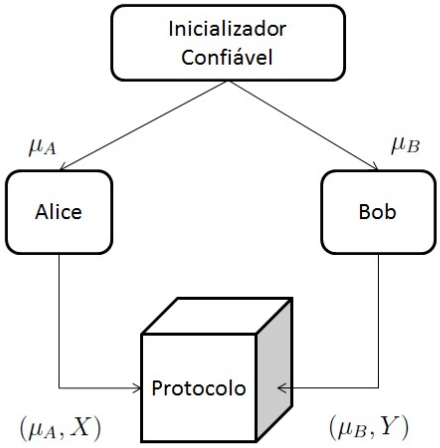

Figura 2.1: Inicializador Confiável

Desse exemplo também pode ser visto que os protocolos que utilizam um TI são mais simples. Se o centro de distribuição de chaves não distribuísse pares de chaves entre os participantes no início do protocolo, eles deveriam executar, durante a execução do protocolo, um esquema de acordo de chaves.

Como a participação do TI é limitado à fase inicial do protocolo e a entrega do conjunto de dados correlatos não depende das entradas das partes, garante-se a manutenção da privacidade dos dados das partes em relação ao TI. Salienta-se que a interação entre as partes e o TI é feita antes mesmo do uso dos dados iniciais dos participantes no protocolo.

Esse modelo foi utilizado em diversas situações: *oblivious transfer* (BEAVER, 1997), comprometimentos de bits (RIVEST, 1999; BLUNDO et al., 2002), compartilhamentos de segredos verificáveis (*verifiable secrete sharing*) (NASCIMENTO et al., 2004), entre outros.

O capítulo a seguir apresentará os conceitos de computação de duas partes segura e que podem ser facilmente estendidos para computação de múltiplas partes. Além disso, será apresentado as condições de segurança baseadas na teoria da informação que serão utilizadas para deixar bem claro que a prova de capacidade de *oblivious transfer* atingida nesse trabalho é no modelo malicioso.

3 COMPUTAÇÃO DE DUAS PARTES SEGURA

Neste capítulo, será feito uma breve discussão sobre protocolos de duas partes seguras. As definições apresentadas seguem os trabalhos de (GOLDREICH, 2000b) e (CRÉPEAU; WULLSCHLEGER, 2008). Para realizar uma dada tarefa de forma segura, deve-se definir de forma precisa a tarefa em si e quais requisitos de segurança devem ser compridos ao realizar essa tarefa. A entidade que captura a especificação de uma dada tarefa é chamada de *funcionalidade* desejada. A partir da definição da funcionalidade, procura-se implementar essa funcionalidade através de um protocolo.

Uma forma de se definir protocolos de duas partes é através de um conjunto formado pela especificação de um processo aleatório e um par de entradas e um par de saídas. O processo aleatório descreve a funcionalidade desejada e é denotado por uma função $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ que mapeia o par de entradas para o par de saídas. Portanto, se (x, y) é um par de entradas com uma determinada distribuição de probabilidade, o par da saída também é uma variável aleatória.

A definição anterior é interessante, mas não deixa claro as interações entre as partes que executam o protocolo. A maneira mais usual de se definir protocolos de duas partes é através de máquinas de Turing interativas (Definição 1, Capítulo 2). Então, um protocolo de duas partes consiste em um par de algoritmos interativos probabilísticos que são definidos formalmente como ITMs. Uma outra definição de protocolo de duas partes pode ser extraída da definição de protocolos dada no Capítulo 4. Entretanto, as definições apresentadas aqui são mais simples e são suficientes para grande parte das aplicações de computação de duas partes distribuída.

Os requisitos de segurança básicos para os protocolos de duas partes seguros é a *correção* e a *privacidade*. A correção é a propriedade do protocolo sempre terminar com a saída desejada se as partes seguem as especificações do protocolo. Já a privacidade é a propriedade de que ao término da execução do protocolo, uma parte não deve ser capaz de extrair informações a respeito da entrada da outra parte mais do que é possível extrair através da sua saída.

De fato, é chamado de visão da parte toda a informação em posse de um jogador ao

final do protocolo incluindo os resultados de todas computações locais, amostragens de aleatoriedade local, entradas locais e mensagens trocadas. Um protocolo de duas partes seguro pode exigir outras propriedades como, por exemplo, *fairness*: ou as duas partes recebem saídas ou nenhuma parte recebe saída.

3.1 MODELO ADVERSARIAL

Existem dois tipos de jogadores (partes que executam um determinado protocolo): *honesto* e *corrupto*. Os adversários honestos são aqueles que seguem exatamente o protocolo e se recusam obter qualquer informação que eles não eram supostos obter. Os adversário corruptos podem ser: *ativos*, *passivos*, *adaptativos* e *não adaptativos*.

Os adversários ativos são também chamados de adversários maliciosos. Eles podem desviar arbitrariamente das regras especificadas pelo protocolo. Em especial, um dos principais resultados desse trabalho se refere a um protocolo para *oblivious transfer* baseado em um canal com apagamentos generalizado (GEC) que atinge a capacidade do GEC quando se considera adversários maliciosos. Observe que em (AHLSWEDE; CSISZAR, 2007) um protocolo baseado em GEC foi provado atingir a capacidade do GEC, mas apenas quando se considera adversários honestos-mas-curiosos.

Os adversários passivos são também chamados de honestos-mas-curiosos ou jogador semi-honesto. Eles seguem o protocolo, mas reúnem todos os dados trocados para tentar deduzir informações através deles.

Adversários adaptativos são aqueles que podem corromper outros jogadores que executam um determinado protocolo de acordo com as informações que eles reuniram até então e eles assim o fazem durante a execução do protocolo.

Adversários não-adaptativos são também chamados de adversários estáticos. Eles corrompem um determinado número de jogadores antes do início da execução do protocolo.

Formalmente, um adversário é definido como uma máquina de Turing probabilística.

Nos protocolos para álgebra linear segura, permite-se que eles abortem a execução quando é detectado uma parte desonesta. No caso dele ser suspenso, a parte honesta não obtém sua saída desejada e a segurança do protocolo garante que a parte desonesta

não obtém nenhuma informação além daquela que ele teria obtido se a execução não fosse suspensa.

3.2 CONSIDERAÇÕES SOBRE HIPÓTESES ADICIONAIS

O trabalho de (GOLDREICH; MICALI; WIGDERSON, 1987) foi o primeiro a demonstrar um resultado de "completude"para computação de duas partes segura, isto é, o resultado garante que se uma hipótese adicional é satisfeita, então pode-se computar seguramente qualquer funcionalidade entre duas partes. Mais precisamente, no trabalho em questão supõe-se que se as partes compartilham um canal autenticado mas inseguro, isto é, o adversário pode ter acesso a todas as mensagens enviadas mas não pode modificar as mensagens trocadas pelas partes honestas. Esse requisito se traduz como um adversário limitado computacionalmente. Além disso, considera-se o adversário sendo não adaptativo. Nessas condições, qualquer computação de duas partes segura é possível assumindo a existência de uma permutação *trapdoor*. (A prova desse resultado pode ser vista nas seções 7.3 e 7.4 de (GOLDREICH, 2000b).) Em poucas palavras, uma permutação *trapdoor* é uma permutação que é difícil de inverter sem o *trapdoor*.

Vale a pena ressaltar que quando se considera uma computação de múltiplas partes segura, o trabalho de (GOLDREICH; MICALI; WIGDERSON, 1987) diz que assumindo a existência de permutações *trapdoor*, qualquer computação de múltiplas partes segura é possível na presença de um adversário estático e malicioso corrompendo menos do que metade dos jogadores.

Para adversários maliciosos e adaptativos, o resultado de (RABIN; BEN-OR, 1989) diz que qualquer computação múltiplas partes segura é possível se um canal de *broadcast* está disponível e se o adversário corrompe menos do que a metade dos jogadores.

De maneira geral, é um resultado bem conhecido que computação de duas partes segura só é possível quando se assume hipóteses adicionais. As seguintes condições são as mais comuns para se postular a existência de qualquer computação de duas partes segura:

- Existência de permutações *trapdoor*;
- Ambos os jogadores serem limitados em memória;
- Existência de um canal ruidoso disponível entre os jogadores; e

- Existência de uma terceira parte confiável.

Particularmente, será construído neste trabalho protocolos de duas partes seguros com as hipóteses iniciais de canal ruidoso (no caso do protocolo para *oblivious transfer*) e da existência de uma terceira parte confiável (no caso dos protocolos para álgebra linear distribuída).

3.3 CONDIÇÕES BASEADAS NA TEORIA DA INFORMAÇÃO PARA SEGURANÇA

Será usado no protocolo para *oblivious transfer* baseado em GEC uma segurança baseada na teoria da informação. Essa seção introduz a definição formal de um protocolo de duas partes. Como é de interesse adversários maliciosos, será mostrado a definição de segurança nesse modelo. A definição para o modelo honesto-mas-curioso é similar, com a diferença que os adversários passivos não podem modificar suas entradas.

Antes de apresentar a definição de segurança no modelo malicioso, será mostrado a definição de algoritmo admissível.

Definição 16. *Dado uma funcionalidade f, um par de algoritmos $P = (A_1, A_2)$, representando as partes cujas estratégias podem interagir por meio de trocas de mensagens entre as duas partes e as quais têm acesso a alguma funcionalidade f, será dito que um* par de algoritmos $\widetilde{A} = (\widetilde{A_1}, \widetilde{A_2})$ é admissível *para o protocolo P se ao menos uma das partes é honesta(segue precisamente o protocolo), isto é se $\widetilde{A_1} = A_1$ ou $\widetilde{A_2} = A_2$.*

Para definir segurança no modelo malicioso: Seja X e Y variáveis aleatórias representando as entradas dos jogadores distribuídas de acordo com uma distribuição P_{XY} desconhecida dos jogadores. Seja U e V variáveis aleatórias representando a saída das partes. Assume-se que a funcionalidade f é determinística, mas pode-se generalizar a para funcionalidades probabilísticas sem causar impacto na definição.

Definição 17 ((CRÉPEAU; WULLSCHLEGER, 2008), Teorema 1)**.** *Um protocolo* Π *calcula seguramente a funcionalidade determinística f com erro de no máximo 3ϵ se, para todo par de algoritmos $\widetilde{A} = (\widetilde{A_1}, \widetilde{A_2})$ admissíveis no modelo real para o protocolo* Π *e para qualquer entradas (X, Y) distribuídas de acordo com P_{XY} sobre $\mathcal{X} \times \mathcal{Y}$, $\widetilde{A}$ produz saídas (U, V) distribuídas de acordo com $P_{UV|XY}$ tal que as seguintes condições são satisfeitas:*

- *(Correção) Se ambas as partes são honestas, tem-se que:*

$$\Pr[(U,V) = f(X,Y)] \geq 1 - \epsilon$$

- *(Segurança para Jogador 1) Se o jogador 1 é honesto, então existem variáveis aleatórias Y' e V' distribuídas de acordo com$P_{Y'V'|X,Y,U,V}$, tal que*

$$\Pr[(U,V') = f(X,V')] \geq 1 - \epsilon$$
$$I_S(X;Y'|Y) \leq \epsilon \ e$$
$$I_S(UX;V|YY'V') \leq \epsilon.$$

- *(Segurança para Jogador 2) Se o jogador 2 é honesto, então existem variáveis aleatórias X' e U' distribuídas de acordo com $P_{X'U'|X,Y,U,V}$, tal que*

$$\Pr[(U',V) = f(X',V)] \geq 1 - \epsilon$$
$$I_S(Y;X'|X) \leq \epsilon \ e$$
$$I_S(VY;U|XX'U') \leq \epsilon.$$

Ambos $P_{y'V'|X,Y,U,V}$ e $P_{X'U'|X,Y,U,V}$ devem ter construções explícitas.

3.4 COMPOSIÇÂO DE PROTOCOLOS

Desde que um protocolo pode ser implementado usando outros como sub-rotinas, uma definição de protocolos seguros deve garantir que o protocolo composto de protocolos seguros é ainda seguro. De maneira geral, um protocolo é composto com outro quando eles executam conjuntamente. Normalmente, considera-se três tipos de composição:

- **Composição sequencial**: Os protocolos são executados um atrás do outro de maneira sequencial, isto é, para um protocolo começar a ser executado, o que está sendo executado no momento deve terminar(um começa quando o outro termina);
- **Composição paralela**: Todos os protocolos começam a ser executados ao mesmo tempo (as execuções são paralelas);
- **Composição concorrente**: Um protocolo pode começar a ser executado a qualquer momento. Ele ele pode começar junto com outros protocolos, começar durante a execução de outros protocolos, ou começar quando outros terminam. Observe que esse é o tipo mais geral de composição.

Como explicitado na introdução, grande parte do objetivo deste trabalho é provar que os protocolos aqui apresentados são seguros quando executados concorrentemente com outros protocolos. Este objetivo é alcançado desde que os protocolos estudados neste trabalho são provados seguros no *framework* UC. O capítulo a seguir traz um estudo detalhado desse *framework*.

4 COMPOSIBILIDADE UNIVERSAL

Os protocolos criptográficos são, usualmente, provados seguros quando analisados como tarefas isoladas, sem levar em consideração ambientes de execução complexos. Contudo, a segurança dos protocolos é afetada pelo ambiente na qual esses protocolos são executados, principalmente pelos protocolos que são executados concorrentemente neste ambiente. Por exemplo [2], suponha que Alice e Bob querem jogar uma versão de *Coin Tossing* na qual Alice e Bob escolhem aleatoriamente um bit cada um. Bob vence o jogo se o valor do seu bit é igual ao da Alice, ou seja ele adivinha corretamente o valor do bit da Alice. Alice vence se os valores são diferentes. Um protocolo criptográfico para realizar este jogo com ajuda de um esquema seguro de comprometimento de bit [3] é sugerido abaixo.

Protocolo 4.1 Protocolo de *Coin Tossing*

1: Alice se compromete com sua entrada x.
2: Bob se compromete com sua entrada y.
3: Alice abre x.
4: Bob abre y.
5: Bob ganha se $y = x$.

No protocolo acima, se o comprometimento de bit é *hiding*, Bob não pode trapacear porque ele se compromete com a entrada dele antes de aprender a entrada da Alice. Alice não pode trapacear, se o comprometimento de bit é *binding*, pois Alice tem que abrir o seu comprometimento antes de aprender a entrada do Bob. Finalmente, o protocolo "implementa"corretamente o jogo se o comprometimento de bit é *correctness*.

Então, a análise do protocolo como tarefa isolada mostra que o protocolo é seguro, no sentido que nem Alice nem Bob podem trapacear e se os jogadores não se desviam do protocolo, ele sempre termina com o resultado correto. Entretanto, se mais de um protocolo igual a esse é executado simultaneamente [4], Bob pode trapacear. Como Alice é a

[2]Esse exemplo foi extraído de uma palestra ministrada pelo professor Dr. Dominique Unruh, PhD, sobre composibilidade universal.

[3]Ver a Seção 2.4 para rever as definições de segurança de um esquema de comprometimento de bit.

[4]Observe que o protocolo em questão também é inseguro quando composto sequencialmente. Bob pode guardar os comprometimentos da Alice e utilizá-los nas próximas execuções do protocolo.

primeira a se comprometer com a entrada dela, Bob pode receber o comprometimento de Alice em uma execução do protocolo e passar como entrada dele em outra execução do protocolo. Se a entrada da Alice se repetir em duas execuções, Bob terá sucesso em trapacear e o protocolo não é seguro.

O *framework* de composibilidade universal (UC) surge para tratar da segurança de protocolos que executam em ambientes arbitrários e possivelmente desconhecidos. Os protocolos continuam sendo considerados como tarefas isoladas, isto é as definições de segurança se aplicam na inspeção de uma instância isolada do protocolo, mas o *framework* garante que a composição de protocolos é segura. Com a operação de composição universal, se as condições forem satisfeitas, pode-se construir protocolos que permanecem seguros quando são compostos entre si, mesmo na presença de um número ilimitado de cópias concorrentes que podem ter entradas potencialmente relacionadas e possivelmente estão sob controle do adversário. Outras vantagens do *framework* UC são: permitir que os protocolos sejam desenvolvidos de forma modular e permitir a análise de protocolos complexos a partir de blocos de construção mais simples.

O *framework* UC define um *modelo real*, que formaliza a execução de um dado protocolo, na presença de um adversário, dentro de um ambiente computacional; e um *modelo ideal*, onde é formalizado o processo que executa a *funcionalidade ideal*. A funcionalidade ideal captura os requisitos que o protocolo deve satisfazer e se comporta como uma terceira parte confiável, isto é, no processo ideal de execução de protocolos as partes enviam suas entradas para a funcionalidade ideal que computa localmente as saídas e entrega para cada parte a saída adequada. Na prática, esses modelos diferem entre si pela forma de comunicação entre as partes e pelo adversário.

O ponto principal do *framework* UC consiste na definição de uma entidade chamada ambiente que representa toda atividade externa à execução do protocolo. Com o ambiente, define-se segurança UC. De uma maneira geral, um protocolo é dito ser UC seguro se para todo adversário no modelo real, existe um adversário no modelo ideal tal que qualquer ambiente não é capaz de distinguir entre a execução do protocolo na presença de um adversário real e a execução do processo ideal na presença de um adversário ideal.

A questão da composibilidade universal é capturada pelo Teorema da Composição que também permite que os protocolos sejam projetados de maneira modular. Com esse teorema, pode-se desenvolver um protocolo usando uma funcionalidade ideal e então

trocar essa funcionalidade por um subprotocolo UC equivalente e tem-se a garantia que a segurança será mantida. Vale a pena ressaltar que o *framework* UC utiliza o teorema da composição como uma ferramenta para garantir que os protocolos UC seguros mantêm sua segurança quando executados em ambientes multiparte arbitrários.

Para se definir protocolos que seguramente realizam uma dada tarefa no *framework* UC, segue três passos:

1. Formaliza-se o processo de execução de um protocolo na presença de um adversário e em um ambiente computacional;
2. Formaliza-se o processo ideal de execução da funcionalidade;
3. Prova-se que o protocolo real (o processo de executar o protocolo) emula o protocolo ideal (o processo ideal de execução daquela funcionalidade).

Este capítulo segue as definições e conceitos de (CANETTI, 2000; CANETTI, 2001). Apresenta-se o modelo básico de execução dos protocolos que serve de base para a formulação do modelo real e do modelo ideal, a definição formal de segurança UC e o Teorema da Composição.

4.1 MODELO BÁSICO DE COMPUTAÇÃO

Para que se possa definir segurança de protocolos, apresenta-se um modelo para representar sistemas distribuídos e protocolos dentro de tais sistemas. O modelo computacional utilizado no *framework* UC estende o modelo de máquinas de Turing Interativas de (GOLDWASSER; MICALI; RACKOFF, 1989; GOLDREICH, 2000a). A definição de ITM é encontrado na Seção 2.2.

Como nos capítulos anteriores, os protocolos, as partes que executam os protocolos e os adversários são também representados por ITMs neste modelo de computação. Um sistema distribuído, na qual o protocolo será executado, é formalizado como um sistema de ITMs.

Definição 18. *Um* sistema (I, C) de ITMs *consiste de uma ITM inicial I e uma função de controle C.*

A função de controle C determina quais fitas de quais ITMs podem ser escritas por cada ITM do sistema. Diz-se que o sistema de ITMs é estendido, se a função de controle pode também modificar as requisições de escrita externa.

Definição 19. *Uma instância de uma ITM $\mu = \{M, id\}$ consiste de um código M e de uma identidade id. A identidade id é formado pelo identificador de sessão SID e pelo identificador da parte PID.*

O acrônimo ITI será utilizado para designar uma instância de uma ITM. Cada ITI pode invocar outras ITIs e escrever mensagens em algumas fitas de outras ITIs, segundo as permissões dadas por C. A ITI inicial é a $\{I, 0\}$, ou seja, uma instância da ITM inicial I. [5] Intuitivamente, uma ITM representa um algoritmo ou um programa e uma ITI representa um processo que executa o código de uma ITM com alguma entrada específica, ou seja, uma instância do programa executando com uma entrada específica.

Uma configuração de uma ITI $\mu = \{M, id\}$ consiste da descrição do código (função de transição) M, o estado de controle, o conteúdo de todas as fitas e as posições de todas as cabeças, sendo que o conteúdo da fita de identidade é id. Uma configuração está ativa se a fita de ativação possui bit 1. De outro modo, está inativa. Uma ativação de uma ITI consiste em uma sequência de configurações, que correspondem à execução do seu código, até uma configuração inativa ser alcançada, caso em que se diz que a ativação está completa e a ITI está esperando por uma próxima ativação. Se o estado de parada é alcançado, a ITI para e não executa nada em ativações futuras.

A execução de um sistema de ITMs é dada por uma sequência de ativação de ITIs. Contudo, em cada ativação apenas uma ITI é ativada. A primeira ITI a ser ativada é a ITI inicial I. Cada vez que uma ITI μ é ativada, ela executa seu código e pode invocar outras ITIs ou pode escrever na fita de uma outra ITI μ' uma única vez, segundo prescrito em C. Quando a ITI μ entra no estado de espera, a ITI que teve alguma de suas fitas escrita, μ', entra em ativação. Se a ITI μ entra no estado de espera e não escreveu em nenhuma fita de outra ITI, a próxima ITI a ser ativada é a ITI inicial. A execução do sistema termina quando a ITI inicial atinge o estado de parada e a saída gerada pela execução é o conteúdo da fita de saída da ITI inicial quando esta atinge o estado de parada.

A instrução de escrita externa de uma ITM especifica os códigos e as identidades das

[5] Abusando da notação, ao longo desse capítulo a mesma notação será usada para designar uma ITM e seu código.

ITIs remetente e destinatária além dos dados a serem escritos. Seja $\mu = \{M, id\}$ a ITI remetente e $\mu' = \{M', id'\}$ a ITI destinatária. A instrução de escrita externa será ignorada se a função de controle C não permitir. Se C permite a operação e não existe ITI com identidade id' no sistema, uma nova ITI com código M' e identidade id' será invocada.

Da forma como foi definido, a invocação de uma nova instância de uma ITM é implícita e só ocorre quando uma ITI existente escreve na fita de uma ITI inexistente no sistema. A identidade de cada ITI é determinada então pela instância que a invoca e é imutável. Esse método também garante que as identidades das ITIs são globalmente únicas dentro do sistema.

Cabe salientar que a comunicação feita através da fita de comunicação de entrada representa a "comunicação não-confiável", onde a identidade e o código da entidade que enviou a mensagem não é necessariamente conhecida pelo remetente. Essa comunicação modela as mensagens recebida pela rede. Já a comunicação feita através das fitas de entrada e de saída de sub-rotina representam a "comunicação confiável", onde o remetente confia na identidade e código da entidade que enviou a mensagem. Essa comunicação modela as mensagens recebidas e enviadas para outros programas confiáveis que geralmente são executados na mesma máquina. [6]

Como dito no início desta seção, um protocolo é modelado como uma única ITM, a qual representa o código a ser executado por cada participante. Uma instância de um protocolo, isto é, uma execução específica do protocolo é definido da seguinte forma:

Definição 20. *Dado um sistema de ITMs, a instância do protocolo multipartes π com SID sid é um conjunto de ITMs no sistema cujo código é π e cujo SID é sid.*

Cada ITI em uma instância do protocolo é chamada de parte e tem um único PID. Então, tem-se que o SID identifica uma instância do protocolo e o PID identifica uma parte dentro de alguma instância do protocolo. Note que cada ITI dentro de uma instância do protocolo possui todo o código do protocolo, mas só executa a parte que lhe cabe. Observe também que o número de partes não é definido, o que permite que os protocolos possuam um número ilimitado de participantes *a priori*. Quando

[6] Observe que uma ITI pode não usar sua fita de saída e, ao contrário, escrever diretamente na fita de saída de sub-rotina do remetente. A única exceção é a ITI inicial, que necessariamente precisa escrever na sua própria fita de saída.

o protocolo π está entendido pelo contexto ou não é especificado, usa-se a notação P_i para representar a i-ésima parte executando o protocolo π de acordo com alguma ordem arbitrária.

Uma última observação que deve ser feita é que as ITMs de interesse são as probabilísticas e de tempo polinomial (PPT). Diz-se que uma ITM M é localmente p-limitada se, a qualquer ponto durante a execução de qualquer ITI μ com código M, o tempo total de execução até então é limitado por um polinômio $p(n)$ tal que $n = k + n_I - n_O - kn_N$, onde k é o parâmetro de segurança; n_I é o número de bits escritos na fita da entrada de μ; n_O é o número total de bits escritos por μ na fita de entrada de outras ITIs; e n_N é o número de diferentes ITIs cujas fitas foram escritas por μ. M é p-limitada se M é localmente p-limitada e cada requisição de escrita externa especifica uma ITM destinatária p-limitada. Se existe um polinômio p tal que M é p-limitada, tem-se que M é PPT. Finalmente, um protocolo multiparte é PPT se a ITM que o formaliza é PPT.

4.2 EXECUÇÃO DO PROTOCOLO NO MODELO REAL

Nesta seção, esboça-se o modelo de execução de um dado protocolo π pelas partes $P_1, \cdots, P_n$ na presença de um adversário $\mathcal{A}$ e dentro de um ambiente de execução $\mathcal{Z}$ com entrada z. Como dito anteriormente, todas as entidades π, $P_1, \cdots, P_n$, $\mathcal{A}$ e $\mathcal{Z}$ são modeladas como ITMs. Para que o protocolo seja realizável em condições realistas, as ITMs são consideradas probabilísticas e executando em tempo polinomial - PPT. [7]

A execução de protocolos multiparte é formalizado no modelo real em termos de um sistema de ITMs. Algumas suposições sobre a comunicação entre ITMs são feitas: a rede é assíncrona e sem garantia de entrega de mensagens; a comunicação é pública e idealmente autenticada, isto é, o adversário não pode modificar as mensagens enviadas pelas partes nem duplicá-las; e as partes não possuem nenhuma informação pré-compartilhada *a priori*. Além disso, tem-se que o adversário pode corromper adaptativamente uma quantidade ilimitada de partes e é ativo no controle das partes corrompidas. Mais precisamente,a execução de um protocolo multiparte é formalizado pelo sistema estendido de ITMs $(\mathcal{Z}, C_{EXEC}^{\pi,\mathcal{A}})$, onde a ITM inicial é o ambiente $\mathcal{Z}$ e a função de controle $C_{EXEC}^{\pi,\mathcal{A}}$ é parametrizado pelo protocolo π e o adversário $\mathcal{A}$.

[7] Como o ambiente possui uma entrada arbitrária, isto equivale dizer que o ambiente é uma ITM PPT não uniforme do ponto de vista da teoria da complexidade.

A entrada z do ambiente $\mathcal{Z}$ representa todas as entradas externas ao sistema, o que inclui as entradas locais de todas as partes. Em especial, a função de controle determina que a primeira ITI a ser chamada pelo ambiente é o adversário $\mathcal{A}$. Em todas as outras ativações, $\mathcal{Z}$ pode chamar um número ilimitado de ITIs, passar entradas para eles e ler suas saídas. Todas as ITIs chamadas por $\mathcal{Z}$ possuem o mesmo SID, que é escolhido por $\mathcal{Z}$, e, exceto por $\mathcal{A}$, a função de controle força essas ITIs a possuírem o mesmo código π. Ou seja, todas as ITIs chamadas por $\mathcal{Z}$ são partes de uma única instância de π. Além dessas, $\mathcal{Z}$ não pode passar entradas para, nem aceitar saídas de, nenhuma outra ITI. Observe que o ambiente não pode acessar nenhuma das fitas de comunicação das partes e do adversário, nem as fitas de entrada e saída que são subpartes de π (uma subparte de π é uma sub-rotina ou de uma parte, ou de outra subparte de π).

Por outro lado, a função de controle permite às partes e às subpartes de π chamar outras ITIs como sub-rotinas e passar entradas e saídas para qualquer outra ITI da mesma instância de π, que não seja o ambiente ou o adversário. As partes de π podem também passar saídas para o ambiente e escrever mensagens na fita de comunicação de entrada do adversário. Essas mensagens podem especificar uma identidade de uma parte de π como destinatário final da mensagem.

Ao adversário $\mathcal{A}$ é permitido, pela função de controle, enviar mensagens para qualquer ITI do sistema. Mais precisamente, as partes não podem se comunicar entre si diretamente através das fitas de comunicação. Todas as mensagens são entregues por $\mathcal{A}$. Contudo, não precisa haver correspondência entre as mensagens enviadas pelas partes e as mensagens entregues pelo adversário. As ITIs nas quais $\mathcal{A}$ entrega mensagens não precisam ser partes da instância de π com SID sid, isto é, $\mathcal{A}$ pode até chamar novas ITIs para entregar mensagens, que podem ser ou não partes da instância de π. Observe que o adversário tem acesso apenas à comunicação entre as partes e não tem acesso à suas entradas e saídas. Além disso, o adversário pode corromper partes de π. Depois de receber uma mensagem (**Corrupt**, id) do ambiente, o adversário corrompe a parte cujo identificador é id enviando uma mensagem (**Corrupt**) para ela. Neste caso, o adversário toma conhecimento de toda informação interna conhecida pela parte e corrompida e passa a controlar esta parte em todas as suas ações futuras. Observe que o ambiente sabe quais partes são corrompidas.

A saída da execução do protocolo é a saída do ambiente. Sem perda de generalidade, assume-se que a saída do ambiente é um bit. Denote $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}(k,z,\vec{r})}$ a saída do ambiente $\mathcal{Z}$ quando interage com adversário $\mathcal{A}$ e as partes P_i executando o protocolo

π com parâmetro de segurança k, entrada z e entrada aleatória $\vec{r} = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, \cdots, r_n$ (z e $r_{\mathcal{Z}}$ para $\mathcal{Z}$, $r_{\mathcal{A}}$ para $\mathcal{A}$, r_i para parte P_i). Denote $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}(k,z)}$ a variável aleatória descrevendo $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}(k,z,\vec{r})}$ quando $\vec{r}$ é uniformemente escolhido. Denote $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ o ensemble de distribuições de probabilidade $\{\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}(k,z)}\}_{k\in\mathcal{N},z\in\{0,1\}^*}$.

A figura 4.1 mostra o esquemático das comunicações entre as entidades durante a execução de um protocolo no modelo real. Lembre-se que é o adversário que controla a comunicação entre as partes e que $\mathcal{A}$ e $\mathcal{Z}$ se comunicam livremente através da fita de entrada de $\mathcal{A}$ e da fita de saída de sub-rotina de $\mathcal{Z}$. As linhas cheias representam a comunicação confiável através das fitas de entrada, de saída e de saída de sub-rotina. Já as linhas tracejadas representam a comunicação não-confiável através da fita de comunicação de entrada.

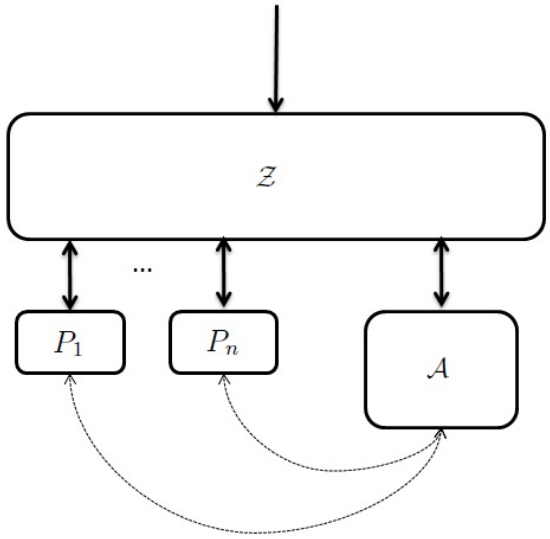

Figura 4.1: Execução do protocolo no modelo real

A execução é uma sequência de ativações, onde a cada ativação um único participante ($\mathcal{Z}$, $\mathcal{A}$ ou algum P_i) é ativado. Com as noções do que cada entidade pode fazer em uma execução do protocolo no modelo real, mostra-se as ativações de cada uma dessas entidades. A ordem de ativação é a seguinte:

1. O ambiente $\mathcal{Z}$ sempre é ativado primeiro e após sua primeira ativação, ele ativa o adversário $\mathcal{A}$;

2. Quando o ambiente é ativado, ele ativa o adversário $\mathcal{A}$ ou alguma parte P_i escrevendo na sua fita de entrada;

3. Se o adversário é ativado, ele pode ativar o ambiente ou alguma parte entregando uma mensagem para esta parte;

4. Se uma parte é ativada, e envia uma mensagem durante a ativação, o adversário é ativado em seguido, de outra forma o ambiente é ativado em seguida.

O ambiente pode escrever apenas uma entrada em cada ativação (para ativar apenas uma outra entidade). De forma similar, o adversário pode entregar apenas uma mensagem por ativação.

A seguir tem-se o que cada entidade faz quando ativado.

- *Quando o ambiente é ativado*: Ele pode ler o conteúdo das fitas de saída de todas as partes e do adversário e pode escrever na fita de entrada de uma das partes ou do adversário. Uma vez que que o ambiente termina sua ativação e entra no estado de espera, a entidade que teve sua fita de entrada escrita é ativada em seguida. A execução do protocolo termina quando o ambiente completa uma ativação sem escrever na fita de entrada de qualquer entidade;

- *Quando o adversário é ativado*: Ele lê suas próprias fitas:

  - Se uma mensagem foi escrita por alguma parte na sua fita de comunicação de entrada, o adversário pode copiar e entregá-la para a parte destinatária escrevendo esta mensagem na fita de comunicação de entrada desta parte. O adversário também pode entregar uma mensagem de corrupção para uma das partes ou escrever informação arbitrária em sua fita de saída;

  - Se o adversário entrega uma mensagem para alguma parte não-corrompida, esta parte é ativada em seguida. Senão, o ambiente será ativado em seguida;

- *Quando uma parte é ativada*: Ela lê suas fitas de entrada e de comunicação de entrada e segue seu código. Ela pode escrever na sua fita de saída ou na fita de comunicação de entrada do adversário (o que modela o envio de uma mensagem a uma parte). Uma vez que a ativação da parte está completa, o adversário é ativado se uma mensagem é enviada ou, de outra forma, o ambiente é ativado.

Observe esta sequência de ativação permite que $\mathcal{Z}$ e $\mathcal{A}$ troquem mensagens usando suas fitas de entrada e saída, entre duas ativações de alguma parte P_i.

4.3 EXECUÇÃO DO PROTOCOLO NO MODELO IDEAL

No modelo ideal, a funcionalidade desejada de uma certa tarefa, ou seja, a descrição do que o protocolo deve fazer e quais informações as partes não podem obter, é capturada pela funcionalidade ideal. Como dito anteriormente, no processo ideal de execução do protocolo as partes enviam suas entradas para a funcionalidade ideal que computa o resultado e entrega a os valores de saída apropriados para cada parte. Além disso, requer-se que a funcionalidade ideal seja uma entidade incorruptível e o adversário nesse processo ideal, chamado de adversário ideal, é limitado a interagir com a funcionalidade ideal em nome das partes corrompidas. De maneira geral, a segurança UC é definida comparando a execução real do protocolo e a execução do processo ideal.

Formalmente, a funcionalidade ideal $\mathcal{F}$ é uma ITM que pode receber entradas e gerar saídas múltiplas vezes durante a execução do processo ideal. Tecnicamente, a fita de entrada de $\mathcal{F}$ pode ser escrita por um número ilimitado de ITIs e $\mathcal{F}$ pode escrever na fita de saída de sub-rotina de um número ilimitado de ITIs. Observa-se que a comunicação de $\mathcal{F}$ com as partes é confiável, pois é feita através das fitas de entrada e de saída de sub-rotina. Entretanto, a comunicação de $\mathcal{F}$ com o adversário ideal $\mathcal{S}$ é inerentemente não-confiável, portanto ela se dá através da fita de comunicação de entrada.

Outras restrições técnicas da funcionalidade ideal são: o PID de $\mathcal{F}$ é configurado para ser $\perp$; todas as entradas esperadas são escritas por ITIs que possuem SID igual ao de $\mathcal{F}$, sendo de outra forma ignoradas; e é responsabilidade de $\mathcal{F}$ determinar os efeitos da corrupção. Em relação à esse último ponto, todas as funcionalidades ideais contidas neste trabalho determinam que quando um adversário corrompe uma parte, ele lê todas as fitas e tem acesso a todos os estados passados da parte corrompida, isto é, ele adquire o conhecimento de todas as computações internas das partes feitas e todas as entradas e saídas que a parte recebeu até então. Além disso, o adversário passa a controlar as ações futuras da parte corrompida.

Para facilitar a exposição, o processo ideal será formulado como um protocolo. O protocolo ideal ϕ que implementa a funcionalidade ideal $\mathcal{F}$ interage com um conjunto de partes *dummy*, um adversário ideal $\mathcal{S}$ e um ambiente $\mathcal{Z}$ com entrada z. Observe que o protocolo ideal ϕ está associado ao protocolo π (a qual será chamado de protocolo real) cuja especificação é capturada por $\mathcal{F}$. As interações de ϕ com as demais entidades são descritas a seguir.

A interação do ambiente $\mathcal{Z}$ com o protocolo ideal é a mesma daquela com o protocolo real. Essa restrição é vital para que o ambiente atue como um distinguidor entre o protocolo ideal e o protocolo real.

Sempre que uma parte *dummy* com identidade (sid, pid) é ativada pela escrita de sua fita de entrada, ela escreve o mesmo valor na fita de entrada de $F(sid, \perp)$, isto é, na ITI de $\mathcal{F}$ cujo SID é *sid*. Sempre a parte *dummy* é ativada pela escrita de sua fita de saída de sub-rotina, ela escreve o mesmo valor na fita de saída de sub-rotina de $\mathcal{Z}$.

O adversário ideal $\mathcal{S}$ não tem acesso ao conteúdo das mensagens entre as partes *dummy* não corrompidas e $\mathcal{F}$. [8] Ao receber uma mensagem do ambiente para corromper uma parte *dummy*, $\mathcal{S}$ envia uma mensagem de corrupção para $\mathcal{F}$ que deve determinar os efeitos da corrupção. Observa-se que não há comunicação direta entre as partes *dummy*.

A funcionalidade ideal $\mathcal{F}$ contém instruções para gerar as saídas das partes a partir das entradas. Como dito anteriormente, $\mathcal{F}$ recebe mensagens das partes *dummy* pela fita de comunicação de entrada e envia mensagem a elas escrevendo em suas fitas de saída de sub-rotina. $\mathcal{F}$ também pode receber mensagens diretamente pelo adversário $\mathcal{S}$ e conter instruções para enviar mensagens para $\mathcal{S}$.

A saída da execução do protocolo ideal é o bit de saída do ambiente. A saída do ambiente é interpretada como se o ambiente acha que está interagindo com o protocolo real ou com o protocolo ideal. De forma semelhante ao modelo real, denote $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}(k,z,\vec{r})}$ a saída do ambiente $\mathcal{Z}$ quando interage com adversário $\mathcal{S}$ e as partes *dummy* $\tilde{P}_i$ executando o protocolo ϕ com parâmetro de segurança k, entrada z e entrada aleatória $\vec{r} = r_{\mathcal{Z}}, r_{\mathcal{S}}, r_1, \cdots, r_n$ (z e $r_{\mathcal{Z}}$ para $\mathcal{Z}$, $r_{\mathcal{S}}$ para $\mathcal{S}$, r_i para parte $\tilde{P}_i$). Denote $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}(k,z)}$ a variável aleatória descrevendo $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}(k,z,\vec{r})}$ quando $\vec{r}$ é uniformemente escolhido. Finalmente, denote $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ o ensemble de distribuições de probabilidade $\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}(k,z)}\}_{k\in\mathcal{N},z\in\{0,1\}^*}$.

A figura 4.2 demonstra as comunicações entre as entidades executando o protocolo no

[8] Pode-se considerar que uma vez que o adversário $\mathcal{S}$ é ativado, além dele poder ler sua própria fita de entrada ele pode ler o destinatário (o cabeçalho público) das mensagens na fita de saída de sub-rotina de $\mathcal{F}$. Isto é, $\mathcal{S}$ pode ver a identidade do beneficiário de cada mensagem enviada por $\mathcal{F}$, mas ele não pode ver o conteúdo desta mensagem, a menos que o destinatário da mensagem seja $\mathcal{S}$ ou uma parte corrompida. Observe que ainda neste caso, a comunicação entre $\mathcal{F}$ e as partes é confiável (idealmente autêntica e secreta). Também pode-se considerar que $\mathcal{S}$ entrega a mensagem de $\mathcal{F}$ para alguma parte, sendo esta mensagem copiada para a fita de comunicação de entrada da parte.

modelo ideal. Observe que o adversário $\mathcal{S}$ se comunica apenas com $\mathcal{F}$.

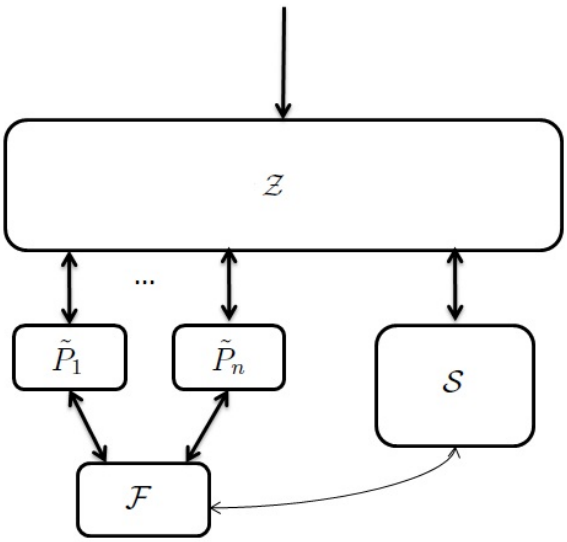

Figura 4.2: Execução do protocolo no modelo ideal

Observe que, como definido anteriormente, a execução do protocolo ideal é uma sequência de ativações e que a cada ativação apenas uma entidade entre $\mathcal{Z}, \mathcal{S}, \mathcal{F}, \tilde{P}_i$ é ativada. A ordem de ativação é definida da seguinte forma:

1. O ambiente $\mathcal{Z}$ sempre é ativado primeiro e após sua primeira ativação, ele ativa o adversário $\mathcal{S}$;
2. Quando o ambiente é ativado, ele ativa o adversário ideal $\mathcal{S}$ ou alguma parte *dummy* $\tilde{P}_i$ escrevendo na sua fita de entrada;
3. Se o adversário ideal $\mathcal{S}$ é ativado, ele ativa funcionalidade ideal $\mathcal{F}$ escrevendo na fita de saída de sub-rotina desta. De outra forma, ele ativa o ambiente;
4. Se uma parte *dummy* é ativada, ela ativa a funcionalidade ideal se escreve na fita de entrada de $\mathcal{F}$, ou ela ativa o ambiente se escreve na fita de saída de sub-rotina de $\mathcal{Z}$;
5. Se a funcionalidade ideal é ativada, ela completa sua ativação e a última entidade que foi ativada antes de $\mathcal{F}$ é ativada novamente.

A seguir, resume-se o que cada entidade faz quando ativado.

- *Quando o ambiente $\mathcal{Z}$ é ativado*: Ele pode ler o conteúdo das fitas de saída de todas as partes *dummy* e do adversário ideal e pode escrever na fita de entrada de uma das partes *dummy* ou do adversário ideal. Uma vez que que o ambiente termina sua ativação e entra no estado de espera, a entidade que teve sua fita de entrada escrita é ativada em seguida. A execução do protocolo termina quando o ambiente completa uma ativação sem escrever na fita de entrada de qualquer entidade;

- *Quando o adversário ideal $\mathcal{S}$ é ativado*: Ele lê suas próprias fitas. Se uma mensagem foi escrita pela funcionalidade ideal na sua fita de saída de sub-rotina ou pelo ambiente na sua fita de entrada, o adversário ideal faz as computações pertinentes e pode escrever na fita de entrada da funcionalidade ideal ou na fita de saída de sub-rotina do ambiente (este último equivale ao adversário escrever na sua própria fita de saída). Mais precisamente, se uma mensagem de corrupção foi escrita pelo ambiente, o adversário ideal envia uma mensagem de corrupção para a funcionalidade ideal;

- *Quando uma parte* dummy $\tilde{P}_i$ *é ativada*: Ela lê suas fitas de entrada e de saída de sub-rotina. Se uma entrada x foi escrita na fita de entrada, ela escreve x na fita de entrada de F. Se o valor x foi escrito na sua fita de saída de sub-rotina, ela escreve x na fita de saída de sub-rotina de $\mathcal{Z}$;

- *Quando a funcionalidade ideal $\mathcal{F}$ é ativada*: Ela lê sua fita de entrada, segue seu código e escreve na fita de saída de sub-rotina de $\mathcal{S}$ ou de uma $\tilde{P}_i$;

Observe esta sequência de ativação ainda permite que $\mathcal{Z}$ e $\mathcal{S}$ troquem mensagens usando suas fitas de entrada e saída, entre duas ativações de alguma parte *dummy* $\tilde{P}_i$.

4.4 DEFINIÇÃO DE SEGURANÇA UC

Com as definições formais de execução do protocolo e execução do protocolo ideal (que representa o processo ideal de execução de uma dada tarefa), pode-se definir formalmente segurança UC. Um protocolo π é uma implementação segura de uma funcionalidade ideal $\mathcal{F}$ se para todo adversário $\mathcal{A}$ executando no modelo real existe um adversário $\mathcal{S}$ executando no modelo ideal, tal que nenhum ambiente $\mathcal{Z}$, para qualquer valor de entrada, pode distinguir com probabilidade não desprezível se está interagindo com $\mathcal{A}$ e as partes que executam π no modelo real, ou se está interagindo com $\mathcal{S}$ e a funcionalidade ideal $\mathcal{F}$ no modelo ideal. Portanto, para o ambiente, executar

o protocolo π equivale a executar um protocolo ideal que interage com a funcionalidade ideal $\mathcal{F}$. Tecnicamente, se tem:

Definição 21. *Sejam $c, n \in \mathcal{N}$, $\mathcal{F}$ uma funcionalidade ideal e π um protocolo executado por n partes. Diz-se que π* UC-realiza *$\mathcal{F}$ se para todo adversário $\mathcal{A}$ existe um adversário ideal $\mathcal{S}$ tal que para todo ambiente $\mathcal{Z}$:*

$$\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}, \tag{4.1}$$

onde $\pi, \mathcal{A}, \mathcal{S}, \mathcal{Z}$ são PPT.

Observe que a Definição 21 pode ser estendida para as noções de indistinguibilidade estatística e perfeita. Quando $\mathcal{A}$ e $\mathcal{Z}$ são limitados em complexidade e $\mathcal{S}$ é polinomial na complexidade de $\mathcal{A}$, diz-se que π *UC-realiza estatisticamente* $\mathcal{F}$. Se além disso, as distribuições de $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ e $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ são idênticas, diz-se que π *UC-realiza perfeitamente* $\mathcal{F}$.

Na prática, para provar que um protocolo π é UC seguro, define-se a funcionalidade ideal e constrói-se um adversário ideal. $\mathcal{S}$ é construído de tal forma que toda a ação que um adversário $\mathcal{A}$ qualquer possa realizar interagindo com o protocolo real pode ser simulado pelo adversário ideal $\mathcal{S}$ que interage com o protocolo ideal. [9] Para não ter que definir um adversário ideal $\mathcal{S}$ (a partir de agora, será chamado de $\mathcal{S}$ também de simulador) para cada adversário $\mathcal{A}$, permite-se que o simulador tenha um acesso "caixa-preta"ao adversário $\mathcal{A}$ e exige-se que o código do simulador permaneça o mesmo para todo $\mathcal{A}$. Em (CANETTI, 2000, Prova da Tese 11) tem-se a prova de que a segurança UC via simulação caixa-preta é equivalente à noção se segurança UC padrão.

Para provar segurança UC via simulação caixa-preta, deve-se impor algumas restrições à operação do simulador. Tecnicamente, na simulação caixa-preta, o adversário $\mathcal{S}$ que interage com o protocolo ideal será chamado de simulador *shell*. $\mathcal{S}$ implementará dentro dele um ITM $\hat{\mathcal{S}}$, chamado de simulador caixa-preta, e um adversário PPT $\mathcal{A}$. [10] A operação de $\mathcal{S}$ é a seguinte:

- Assim que $\mathcal{S}$ é ativado, ele invoca uma instância de $\mathcal{A}$ e de $\hat{\mathcal{S}}$;

[9] Observe que essa noção é equivalente a dizer que a execução do protocolo no modelo real é indistinguível, a menos de uma probabilidade desprezível.

[10] Além da comunicação entre $\hat{\mathcal{S}}$ e $\mathcal{A}$ ser mais restrito que o modelo criptográfico padrão de simulação caixa-preta, outra restrição ao simulador é que $\hat{\mathcal{S}}$ não pode "*resetar*"ou "rebobinar"$\mathcal{A}$.

- Entradas do ambiente para $\mathcal{S}$:
  - As entradas são passados para $\mathcal{A}$ e então para $\hat{\mathcal{S}}$;
  - O resultado da computação geradas por $\mathcal{A}$ são enviadas como entradas para $\hat{\mathcal{S}}$;
  - As mensagens de saída de $\hat{\mathcal{S}}$ são passados como mensagem de saída de $\mathcal{S}$;
- Mensagens recebidas da funcionalidade ideal:
  - As mensagens são enviadas como entrada para $\hat{\mathcal{S}}$;
  - As saídas de $\hat{\mathcal{S}}$ resultantes da computação dessas mensagens são passados como mensagens de entrada para $\mathcal{A}$;
  - A saída de $\mathcal{A}$ proveniente dessa computação é passado como saída de $\mathcal{S}$ para $\mathcal{Z}$.

A figura 4.3 apresenta o esquemático das comunicações quando se define segurança UC através de simulação caixa-preta.

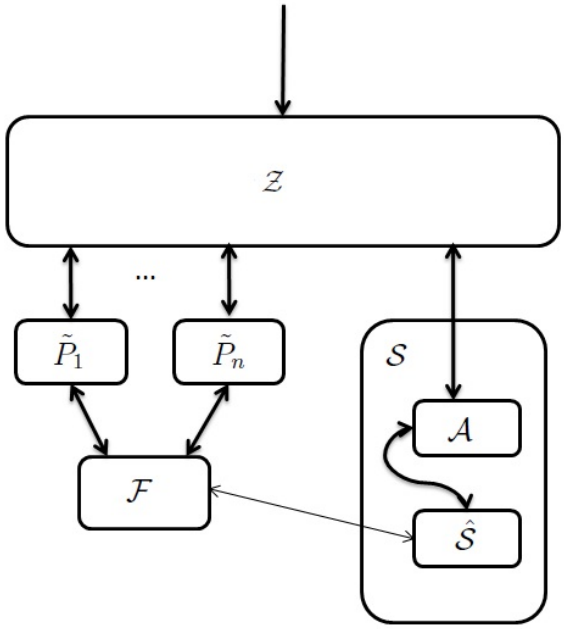

Figura 4.3: Simulação Caixa-Preta

Seja $\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\hat{\mathcal{S}},\mathcal{Z}}$ a saída de $\mathcal{Z}$ depois de interagir com um protocolo ideal que implementa uma funcionalidade ideal $\mathcal{F}$ e com um adversário *shell* que executa um simulador caixa-preta $\hat{\mathcal{S}}$ e um adversário $\mathcal{A}$. Diz-se que um protocolo π *UC-realiza* $\mathcal{F}$ *com simulação caixa-preta* se existe um simulador caixa-preta $\hat{\mathcal{S}}$ tal que para qualquer

adversário PPT $\mathcal{A}$, o adversário *shell* $\mathcal{S}$ resultante é PPT e para todo ambiente PPT $\mathcal{Z}$, tem-se: $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A},\hat{s}},\mathcal{Z}}$.

Nas provas UC dos demais capítulos define-se as funcionalidades ideais adequadas, constrói-se o simulador caixa-preta e então prova-se a indistinguibilidade das distribuições da saída do ambiente interagindo no modelo real e no modelo ideal.

4.5 TEOREMA DA COMPOSIÇÃO

A definição de segurança UC da seção 4.4 é suficiente para garantir a segurança para uma instância de protocolo executando sozinho. É o teorema da composição que dá aos protocolos que são UC seguros a garantia de que eles permanecerão seguros quando executados concorrentemente com protocolos possivelmente desconhecidos. O teorema da composição também permite o desenvolvimento de protocolos de forma modular e facilita a análise de protocolos complexos.

O teorema da composição garante a segurança em ambientes arbitrários porque a operação de composição universal permite trocar um protocolo de sub-rotina por outro UC seguro. Isto é, a operação de composição universal garante que o protocolo composto também será UC seguro. Portanto, independente da tarefa que os protocolos realizam, se cada bloco for UC seguro, a composição de blocos será UC seguro.

Define-se o modelo híbrido para se apresentar o Teorema da Composição. De modo geral, o modelo híbrido é definido para permitir a um protocolo ter acesso à funcionalidade ideal.

4.5.1 Modelo de Computação Híbrida

Neste modelo um protocolo interativo π é executado na presença de alguma funcionalidade ideal $\mathcal{F}$. Isso significa que além das partes $P_1, ..., P_n$ enviarem mensagens umas as outras através do adversário como no modelo real, elas podem usar instâncias da funcionalidade ideal $\mathcal{F}$ como no modelo ideal. Mais precisamente, o modelo de computação híbrida determina a existência de um protocolo $\mathcal{F}$-híbrido π, a qual inclui ilimitadas chamadas de sub-rotina para ϕ (o protocolo ideal que interage com $\mathcal{F}$).

Nota-se que cada instância de $\mathcal{F}$ possui um SID escolhido pela parte do protocolo híbrido que invoca ϕ. Isto é, as instâncias de ϕ, que são chamadas pelas partes que executam π, podem ter SID relacionado com o SID da parte que o chamou e cada

instância de ϕ utiliza uma instância própria de $\mathcal{F}$. Outras observações a serem feitas sobre o modelo de computação híbrida são: apesar do adversário entregar as mensagens, ele não tem acesso ao conteúdo delas a não ser que ele ou uma parte corrompida seja o destinatário da mensagem; a corrupção das partes é feita de forma similar aos demais modelos; o ambiente não tem acesso direto às cópias de $\mathcal{F}$; e os protocolos híbridos são estendidos na maneira usual para o caso onde os protocolos fazem uso de múltiplas funcionalidades ideais diferentes.

4.5.2 Operação de Composição Universal

Com a noção adquirida de protocolos híbridos, pode-se substituir as chamadas aos protocolos ideais que interagem com $\mathcal{F}$ por protocolos "reais"que implementam seguramente $\mathcal{F}$. Precisamente define-se a seguinte operação de composição universal:

Seja π um protocolo no modelo híbrido que faz chamadas de sub-rotina para algumas instâncias de $\mathcal{F}$ e seja ρ um protocolo que UC-realiza $\mathcal{F}$. O protocolo composto $\pi^{\rho|\mathcal{F}}$ é construído substituindo o código de cada ITM de π de forma que a primeira mensagem enviada a cada instância de $\mathcal{F}$ com identidade (sid, pid) é substituída por uma correspondente chamada a uma nova instância de ρ com a mesma identidade (sid, pid). Cada mensagem subsequente para a instância existente de $\mathcal{F}$ é substituído por uma mensagem adequada à instância de ρ. Cada valor de saída gerado pela instância de ρ com identidade (sid, pid) é tratado por π como uma mensagem recebida da instância correspondente de $\mathcal{F}$ com identidade (sid, pid).

Um protocolo ρ é *"subroutine respecting"* se a única interface de entrada/saída entre cada instância de ρ e outras instâncias de protocolos é feita pelas partes de ρ. Isto é, as subpartes de ρ trocam entrada/saída apenas com partes ou subpartes de sua instância.

4.5.3 Enunciado do Teorema

O Teorema da Composição Universal estabelece que se ρ UC-realiza $\mathcal{F}$, então a execução do protocolo $\pi^{\rho|\mathcal{F}}$ no modelo real, com nenhum acesso à $\mathcal{F}$, tem essencialmente o mesmo efeito de executar o protocolo $\mathcal{F}$-híbrido π. Então, pode-se dizer que o Teorema da Composição Universal assegura que substituir chamadas para instâncias de $\mathcal{F}$ por chamadas para instâncias de ρ não afeta o comportamento de π de forma distinguível. Isto garante que para qualquer adversário $\mathcal{A}$ no modelo real existe um adversário $\mathcal{H}$ no modelo híbrido tal que para todo ambiente é indistinguível a interação com $\mathcal{A}$ e as

partes executando π^{ρ} no modelo real e a interação com $\mathcal{H}$ e as partes executando π no modelo híbrido.

Teorema 5 (Teorema da Composição Universal)**.** *Seja π, ρ protocolos multipartes PPT tal que ρ UC-realiza $\mathcal{F}$ e ρ é* "subroutine respecting"*. Então, o protocolo $\pi^{\rho|\mathcal{F}}$ UC-realiza $\mathcal{F}$.*

O teorema acima é provado em (CANETTI, 2000). Um corolário derivado do teorema 5 diz que se π seguramente realiza alguma funcionalidade ideal $\mathcal{G}$ (UC-realiza $\mathcal{G}$), então $\pi^{\rho|\mathcal{F}}$ seguramente realiza $\mathcal{G}$.

Corolário 1. *Seja $\mathcal{F}$, $\mathcal{G}$ funcionalidades ideais tal que $\mathcal{F}$ é PPT. Seja π um protocolo* "subroutine respecting" *que UC-realiza $\mathcal{G}$ e seja ρ um protocolo* "subroutine respecting" *que UC-realiza $\mathcal{F}$. Então, o protocolo composto $\pi^{\rho|\mathcal{F}}$ UC-realiza $\mathcal{G}$.*

Prova: Seja $\mathcal{A}$ o adversário que interage com as partes que executam $\pi^{\rho|\mathcal{F}}$ no modelo real. Do Teorema 5 tem-se que existe um adversário $\mathcal{H}$ no modelo híbrido tal que para todo $\mathcal{Z}$, $\text{REAL}_{\pi^{\rho|\mathcal{F}},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{HÍBRIDO}_{\pi,\mathcal{H},\mathcal{Z}}$. Desde que π UC-realiza $\mathcal{G}$, existe um simulador $\mathcal{S}$ tal que para todo $\mathcal{Z}$, $\text{HÍBRIDO}_{\pi,\mathcal{H},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{G},\mathcal{S},\mathcal{Z}}$. Usando a propriedade de transitividade da indistinguibilidade de ensembles, obtém-se que $\text{REAL}_{\pi^{\rho|\mathcal{F}},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{G},\mathcal{S},\mathcal{Z}}$. □

4.5.4 Teorema da Composição Universal com Estado Conjunto

Observe que o Teorema da Composição é valido quando ρ é *"subroutine respecting"*. Isto é, todas as subpartes de uma instância de ρ devem passar ou receber entradas e saídas apenas de partes e subpartes desta instância. Em outras palavras, o teorema só é válido quando as instâncias de ρ possuem estados locais mutuamente disjuntos e aleatoriedade local. Entretanto, encontra-se muitos casos em que múltiplas instâncias concorrentes de um mesmo protocolo π usam a mesma instância de uma sub-rotina ρ.

O Teorema da Composição Universal com Estado Conjunto (teorema JUC) foi apresentado em (CANETTI; RABIN, 2003). Ele garante a segurança de protocolos que utilizam múltiplas instâncias de outro protocolo UC seguro até mesmo quando as múltiplas instâncias usam algum estado conjunto ou uma sub-rotina conjunta.

Antes de apresentar o teorema JUC define-se $\hat{\mathcal{F}}$, a extensão multissessão de uma funcionalidade ideal $\mathcal{F}$, como a funcionalidade ideal que representa múltiplas instâncias independentes de $\mathcal{F}$ dentro de uma única instância. Tecnicamente, $\hat{\mathcal{F}}$ recebe entradas da forma $(sid, ssid, x)$, onde sid é o identificador de sessão de $\hat{\mathcal{F}}$ e $ssid$ é o seu identificador de subsessão. Quando uma entrada é recebida, $\hat{\mathcal{F}}$ chama uma instância de $\mathcal{F}$ cujo SID é $ssid$ e envia a entrada $(ssid, x)$ para aquela instância. Quando uma instância $\mathcal{F}$ gera uma saída $(ssid, vid)$ para alguma parte, $\hat{\mathcal{F}}$ passa $(sid, ssid, x)$ para esta parte.

A operação de composição universal com estado conjunto é similar à operação de composição universal: dado um protocolo $\mathcal{F}$-híbrido π e um protocolo ρ que UC-realiza $\hat{\mathcal{F}}$, o protocolo composto $\pi^{[\rho|\mathcal{F}]}$ é construído substituindo o código de cada ITM de π de forma que no início da computação $\pi^{[\rho|\mathcal{F}]}$ instrui cada parte a invocar uma instância ρ com algum valor arbitrário e fixo para sid. Então, cada chamada $(ssid, x)$ para a instância $ssid$ de $\mathcal{F}$ é substituída por uma chamada $(sid, ssid, x)$ para a instância sid de ρ. Cada valor de saída $(sid, ssid, x)$ gerado pela instância sid de ρ é tratado como um valor x recebido da instância $ssid$ de $\mathcal{F}$. Com as noções de $\hat{\mathcal{F}}$ e operação de composição universal com estado conjunto, finalmente pode-se definir o teorema JUC.

Teorema 6 (Teorema da Composição Universal com Estado Conjunto)**.** *Sejam $\mathcal{F}$ uma funcionalidade ideal, π um protocolo $\mathcal{F}$-híbrido, ρ um protocolo que UC-realiza $\hat{\mathcal{F}}$ e $\pi^{[\rho|\mathcal{F}]}$ o protocolo composto. Então, $\pi^{\rho|\mathcal{F}}$ UC-realiza $\mathcal{F}$.*

O capítulo a seguir apresentará uma das principais contribuições deste trabalho que é um protocolo de *oblivious transfer* seguro contra adversários completamente maliciosos que atinge a capacidade de um canal com apagamentos generalizados e é seguro no *framework* UC.

5 OBLIVIOUS TRANSFER BASEADO EM UM CANAL COM APAGAMENTOS GENERALIZADO

Oblivious Transfer (OT) é uma primitiva criptográfica de duas partes introduzida por Wiesner e Even, Goldreich e Lempel (WIESNER, 1983; EVEN; GOLDREICH; LEMPEL, 1985). Uma primitiva similar foi introduzida por Rabin em (RABIN, 1981) e posteriormente diversas outras variações de OT apareceram.

Na versão de (RABIN, 1981), conhecida como *Rabin OT*, o emissor transmite um bit b para o receptor que recebe o bit b com probabilidade $\frac{1}{2}$ ou um símbolo que indica que o bit foi apagado com probabilidade $\frac{1}{2}$. Além disso, o emissor não sabe se o receptor recebeu o bit ou o símbolo de apagamento. Por outro lado, quando o receptor recebe um símbolo de apagamento, ele pode adivinhar corretamente o bit que foi enviado pelo receptor com uma probabilidade menor ou igual à $\frac{1}{2}$ se b é escolhido independente e uniformemente aleatório. Ou seja, o receptor não sabe qual foi a entrada do emissor neste caso.

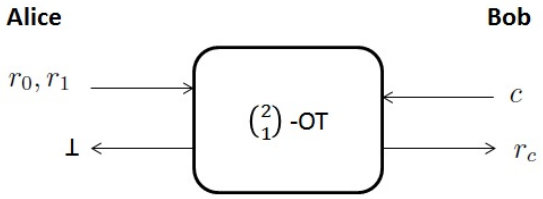

Figura 5.1: One-out-of-Two Oblivious Transfer

Uma das variações principais de OT é o *one-out-of-two oblivious transfer* $(1-2$ OT) introduzido por (EVEN; GOLDREICH; LEMPEL, 1985). [11] Nessa versão, o emissor possui dois bits de entrada b_0, b_1 e o receptor possui um bit c chamado de bit de escolha. No final, o receptor recebe o bit escolhido b_c e não têm nenhuma informação sobre b_{1-c} além do que ele pode inferir das mensagens trocadas com o emissor. Além disso, o

[11] De fato, em (EVEN; GOLDREICH; LEMPEL, 1985), o receptor não tem um bit de escolha. Ele recebe um bit com probabilidade 1/2. O que é chamado neste trabalho de *one-out-of-two oblivious transfer* é precisamente um *one-out-of-two* chosen *oblivious transfer*, porque aqui o receptor escolhe qual bit ele quer receber.

emissor não sabe qual o bit escolhido pelo receptor. A figura 5.1 mostra as entradas e saídas de um protocolo $1-2$ OT.

As versões de Rabin OT e $1-2$ OT foram provadas equivalentes por Crépeau em (CRÉPEAU, 1987). Entretanto, a versão de interesse neste capítulo é o *one-out-of-two string oblivious transfer*(String OT). Essa versão é idêntica ao $1-2$ OT, com exceção de que o emissor possui duas strings $b_0, b_1 \in \{0,1\}^k$ como entrada. Em (BRASSARD; CREPEAU; ROBERT, 1986) tem-se que um String OT pode ser implementado a partir de um $1-2$ OT.

Para facilitar a exposição, a menos que indicado explicitamente, um *one-out-of-two string oblivious transfer* será chamado simplesmente de OT neste capítulo. Outra consideração feita para facilitar a exposição é a suposição de que as entradas são uniformemente aleatórias. Esta suposição pode ser tomada sem perda de generalidade, pois tem-se em (BEAVER, 1995, Seção 3.2) uma redução muito eficiente de um *randomized oblivious transfer* para um OT.

Neste capítulo será apresentado um protocolo de *oblivious transfer* baseado em um Canal com Apagamentos Generalizado (GEC - *Generalized Erasure Channel*). Este protocolo foi inicialmente apresentado em (PINTO et al., 2011) e agora demonstra-se que ele é seguro quando composto universalmente. As propriedades relevantes deste protocolo são:

- Ele é seguro quando executado na presença de adversários completamente maliciosos;
- Ele atinge a capacidade de *oblivious transfer* em um canal com apagamentos generalizado;
- Ele é UC seguro, ou seja, ele permanece seguro quando executado simultaneamente com protocolos diversos que também sejam UC seguros.

Para apresentar o protocolo e suas propriedades, na Seção 5.1 apresenta-se as definições necessárias, na Seção 5.2 apresenta-se o protocolo em si, na Seção 5.2.2 demonstra-se que o protocolo atinge a capacidade de OT para um canal GEC. Finalmente a prova de segurança baseado na teoria da informação e a prova de segurança UC nas últimas duas seções.

5.1 DEFINIÇÕES

As definições relacionadas à análise do protocolo proposto são: a definição de segurança baseada na na teoria da informação para um protocolo de String OT e a definição de capacidade de OT. As definições necessárias na construção do protocolo são as de *Hashing* Interativo e do esquema de codificação de subconjuntos.

5.1.1 Canal com Apagamentos Generalizado

O Canal com Apagamentos Generalizado (GEC - *Generalized Erasure Channel*) pode ser visto como a combinação de um Canal com Apagamentos (*Erasure Channel*) e um Canal Discreto Sem Memória (*Discrete Memoryless Channel*). A seguir apresenta-se as definições formais de canais discretos, canais discretos sem memória e canais discretos com apagamentos.

Definição 22. *Um* Canal Discreto Sem Memória *é um sistema consistindo de um alfabeto de entrada $\mathcal{X}$, um alfabeto de saída $\mathcal{Y}$ e uma matriz de transição de probabilidade $W = [p(y|x)]$, onde $x \in \mathcal{X}$ e $y \in \mathcal{Y}$. A saída não depende do estado inicial do canal e $p(y|x)$ é condicionalmente independente das entradas ou saídas anteriores do canal, i.e., $p(y_1, ..., y_n|x_1, ..., x_n) = p(y_1|x_1)p(y_2|x_2)...p(y_n|x_n)$, $x_i \in \mathcal{X}$ e $y_i \in \mathcal{Y}$.*

Definição 23. *Um* Canal com Apagamentos *é um canal $\{W : \mathcal{X} \to \mathcal{Y}\}$ tal que a entrada é apagada (perdida) comprobabilidade α ou a saída é igual à entrada com uma probabilidade $1 - \alpha$.*

Com essas definições, pode-se definir um Canal Com Apagamentos Generalizado.

Definição 24 ((AHLSWEDE; CSISZAR, 2007)). *Um canal discreto sem memória $\{W : \mathcal{X} \to \mathcal{Y}\}$ vai ser chamado de* Canal Com Apagamentos Generalizado (GEC) *se o alfabeto de saída $\mathcal{Y}$ pode ser decomposto como $\mathcal{Y}_0 \cup \mathcal{Y}^*$ tal que $W(y|x)$ não depende de $x \in \mathcal{X}$, se $y \in \mathcal{Y}^*$. Para um GEC, será denotado $W_0(y|x) = \frac{1}{1-p^*}W(y|x)$, $x \in \mathcal{X}$, $y \in \mathcal{Y}_0$, onde p^* é a soma de $W(y|x)$ para $y \in \mathcal{Y}^*$ (não dependente de x).*

Como dito inicialmente, um GEC é uma combinação de um canal discreto sem memória e um canal com apagamentos: com probabilidade p^* a saída vai ser tal que sua probabilidade é independente da entrada $x \in \mathcal{X}$, isso é como se a entrada fosse apagada, e a saída foi modificada para um $y \in \mathcal{Y}^*$; e com probabilidade $1 - p^*$ a entrada vai ser transmitida em um canal discreto sem memória com matriz de canal W_0. A figura 5.2 demonstra esta visão.

Observe que o Rabin OT pode ser visto como um canal com apagamentos.

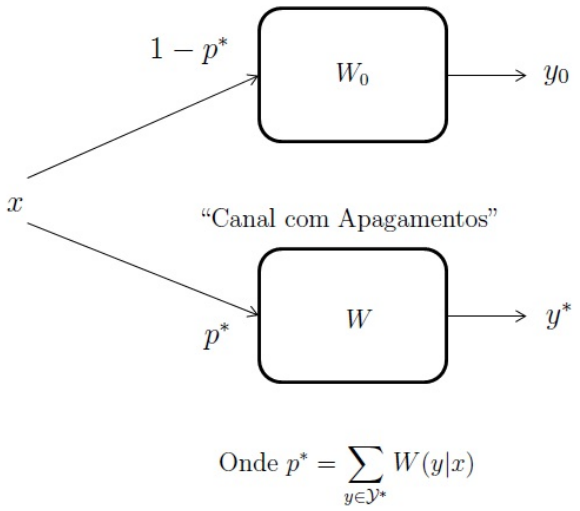

Figura 5.2: Canal com Apagamentos Generalizado

5.1.2 Segurança baseada na Teoria da Informação para OT

Neste trabalho será usada a definição de segurança para o *String* OT de Crépeau e Wullschleger em (CRÉPEAU; WULLSCHLEGER, 2008). Essa definição implica que o protocolo de *String* OT que a satisfaz é composto sequencialmente. Lembre-se de que composição sequencial se refere a execução do mesmo protocolo um atrás do outro, isto é, uma instância do protocolo é começa a ser executada apenas quando a outra instância do protocolo termina. Observe que a máxima segurança possível de se atingir utilizando definições específicas para uma dada tarefa, sem considerar o ambiente de execução, é garantida até composição sequencial. Outra observação que deve ser feita é que o protocolo que será apresentado também é seguro através de composição universal, como será visto na Seção 5.3.

A definição apresentada aqui segue as linhas do Capítulo 3. Lembre-se que um protocolo $\mathcal{F}$-híbrido consiste de um par de algoritmos $P = (A_1, A_2)$ que podem interagir por troca de mensagens entre si e tem acesso a uma funcionalidade $\mathcal{F}$. Então, um par de algoritmos $\widetilde{A} = (\widetilde{A_1}, \widetilde{A_2})$ é admissível para o protocolo P se ao menos uma das partes é honesta, isto é, se $\widetilde{A_1} = A_1$ ou $\widetilde{A_2} = A_2$.

As variáveis aleatórias $B = (B_0, B_1)$ e C representam as entradas do emissor e do receptor, respectivamente, distribuídas de acordo com uma distribuição P_{BC} desconhecida pelas partes. As variáveis aleatórias U e V representam as saídas do emissor e

do receptor, respectivamente. Em (CRÉPEAU; WULLSCHLEGER, 2008) tem-se que a Definição 17 aplicado ao caso da funcionalidade de *Oblivious Transfer* gera a seguinte definição:

Definição 25. *(CRÉPEAU; WULLSCHLEGER, 2008, Teorema 3) Um protocolo P realiza seguramente* String *OT (para strings de comprimento k) com erro de no máximo 6ϵ se, para todo par de algoritmos $\widetilde{A} = (\widetilde{A_1}, \widetilde{A_2})$ que são admissíveis para o protocolo P e para todas entradas (B, C), $\widetilde{A}$ produz as saídas (U, V) tal que as seguintes condições são satisfeitas:*

- *(Correção) Se ambas as partes são honestas, então $U = \perp$ e*

$$\Pr[V = B_C] \geq 1 - \epsilon$$

- *(Segurança para o Emissor) Se o emissor é honesto, então tem-se que $U = \perp$ e existe uma variável aleatória C' distribuída de acordo com $P_{C'|B,C,V}$, tal que*

$$I_S(B; C'|C) \leq \epsilon$$

  e

$$I_S(B; V|C, C', B_{C'}) \leq \epsilon.$$

- *(Segurança para o Receptor) Se o receptor é honesto, tem-se que $V \in \{0,1\}^k$ e*

$$I_S(C; U|B) \leq \epsilon.$$

O protocolo é seguro se ϵ é uma função negligível.

5.1.3 Capacidade de *Oblivious Transfer*

A *capacidade de oblivious transfer* foi implementada por Nascimento e Winter em (NASCIMENTO; WINTER, 2008) e é usada para protocolos de OT construídos com base em um canal ou distribuição ruidosa. Simplificadamente, a capacidade de OT mede quão eficientemente um protocolo de OT usa um recurso ruidoso. Foi demonstrado em (NASCIMENTO; WINTER, 2008) que os recursos ruidosos sobre os quais um protocolo de OT pode ser implementado são canais ou correlações [12] que se tornam não perfeitos após suas redundâncias serem retiradas. A intuição é que em um canal não perfeito

[12] Dado um par de variáveis aleatórias X, Y com distribuição conjunta $P(X, Y)$, a correlação é dada por $\rho(X, Y) = H(X|Y) + H(Y|X)$, onde $H(\cdot)$ é a entropia de Shannon.

não é possível inferir com certeza a entrada a partir da saída. Similarmente, uma distribuição possui uma correlação não-perfeita se $H(X|Y) \neq 0$ (ou equivalentemente, $H(Y|X) \neq 0$), isto é, conhecendo Y não se sabe ao certo X. Observe que o Canal com Apagamentos Generalizados (GEC) (Definição 24) é um canal não-perfeito.

Para definir capacidade de *oblivious transfer*, define-se primeiramente a taxa de *oblivious transfer* e taxa alcançável.

Definição 26. *Se o canal ruidoso é usado n vezes e k é o comprimento da string sendo transferida pelo esquema de OT, a* taxa de oblivious transfer *do protocolo é dado por:*

$$R_{OT} = \frac{k}{n}. \tag{5.1}$$

Se existe um protocolo implementando oblivious transfer *seguramente e com uma taxa de* oblivious transfer *R, diz-se que R é uma* taxa alcançável.

Definição 27. *A* Capacidade de Oblivious Transfer *de um protocolo é dado por:*

$$C_{OT} = \sup\frac{k}{n}. \tag{5.2}$$

onde o supremo é tomado sobre todas as taxas alcançáveis.

A capacidade de *oblivious transfer* do GEC para adversários passivos foi encontrado por Ahlswede and Csiszár em (AHLSWEDE; CSISZAR, 2007). Demonstra-se com o protocolo deste capítulo que a capacidade de *oblivious transfer* do GEC para adversários maliciosos é a mesma daquela para adversários passivos.

Apresenta-se o protocolo de OT baseado em GEC na Seção 5.2, prova-se sua segurança segundo a Definição 25 na Seção 5.2.1 e, finalmente, demonstra-se a capacidade de OT do GEC para adversários maliciosos e que o protocolo em questão atinge essa capacidade na Seção 5.2.2.

Para apresentar o protocolo de OT ainda falta apresentar o esquema de codificação de subconjuntos e a principal ferramenta que é o *Hashing* Interativo.

5.1.4 Esquema de Codificação de Subconjuntos

O protocolo de OT que será apresentado neste capítulo utiliza um esquema de codificação de subconjuntos para resolver a seguinte questão: dado uma *string* de comprimento n, escolhe-se l posições da *string* e deseja-se representar o conjunto k de l posições como uma *string* binária.

O esquema de codificação usado neste trabalho é o de Savvides (SAVVIDES, 2007, Seção 4.2.1) que modifica o esquema de Cachin, Crépeau e Marcil em (CACHIN; CREPEAU; MARCIL, 1998). A codificação associa um inteiro no conjunto $\{0, ..., \binom{n}{l} - 1\}$ com o subconjunto k de $[n]$, $|k| = l$. Tecnicamente, a codificação do subconjunto k de $[n]$ é dado por:

$$\sigma(k) = \sum_{i=1}^{l} \sum_{j=e_{i-1}+1}^{e_1-1} \binom{n-i}{l-i} \tag{5.3}$$

onde $k = \{e_1, e_2, \ldots, e_v, \ldots, e_l\}$. Dessa forma, codifica-se o conjunto um conjunto da forma $\binom{[n]}{l}$ em *strings* binárias de comprimento $m = \lceil log\binom{n}{l} \rceil$(veja (CACHIN; CREPEAU; MARCIL, 1998, Section 3.1) para maiores detalhes).

O processo de decodificação de uma *string* binária m é feito calculando cada e_v como $\max_r\{\sum_{e_1+1}^{r-1} \binom{n-j}{l-i}\} \leq m$ e fazendo $m = m - \sum_{e_1+1}^{j=e_i-1} \binom{n-j}{l-i}$ para todo $1 \leq i \leq l$. Entretanto, uma desvantagem deste esquema é que as strings que correspondem a codificações validas são apenas um pouco mais de metade de todas as *strings*. Savvides propôs que cada *string* $w \in \{0,1\}^m$ codifica o mesmo subconjunto como $w \pmod{log\binom{n}{l}}$, que é sempre uma codificação válida do esquema original. Desde que cada subconjunto corresponde a uma ou duas *strings* em $\{0,1\}^m$, este esquema pode no máximo dobrar a fração de *strings* mapeadas para subconjuntos com a propriedade desejada.

5.1.5 *Hashing* Interativo

O *Hashing Interativo(IH)* foi introduzido em (NAOR et al., 1998). Ela é uma primitiva criptográfica entre dois jogadores, o emissor e o receptor . O emissor tem como entrada uma *string* $w \in \{0,1\}^m$ e o receptor não possui nenhuma entrada. Ao final da execução do protocolo tanto o emissor quanto o receptor recebem duas strings de m bits como saída: uma delas é w e a outra é $w' \neq w$. Seja as duas strings de saída w_0 e w_1, de acordo com uma ordem lexicográfica. Então existe um $d \in \{0,1\}$ tal que $w_d = w$. Além disso, um receptor desonesto não pode dizer qual das strings (w, w') foi a entrada de emissor, e ao menos uma das strings w, w' está efetivamente fora do controle de um emissor desonesto. A figura 5.3 mostra a entrada e saída esperadas em um *hashing* interativo.

Hashing Interativo recebeu esse nome pela implementação inicial que usava funções de *hash* aleatórias. Naquele cenário, um emissor queria enviar uma *string* w para um receptor. O emissor então transferia um *hash* aleatório $y = h(w)$ para o receptor com

a propriedade de que duas entradas diferentes possuem o mesmo *hash* (um *two-to-one hash*). A propriedade desse *hash* garante que o receptor não aprende qual das duas pré-imagens $\{w, w'\} = h^{-1}(y)$ era a entrada do emissor. Além disso, o emissor tem a garantia de que o receptor não aprende qualquer informação adicional de w. Observe que o *hashing* interativo pode ser implementado sem funções de *hash*. Por exemplo, em (SAVVIDES, 2007) tem-se um protocolo de IH onde o emissor quer enviar w com m bits para o receptor. Para tal, o receptor escolhe uniformemente uma matriz $(m-1) \times m$ de posto $m-1$. O receptor envia as linhas da matriz para o emissor uma de cada vez. A cada envio da linha da matriz, o emissor responde com o produto interno entre a linha e w. Depois de $m-1$ linhas serem transmitidas, o emissor e o receptor computam os dois valores de w consistentes com o sistema linear $Q.w = c$, onde Q é a matriz escolhida pela receptor e c é o vetor de resposta do emissor.

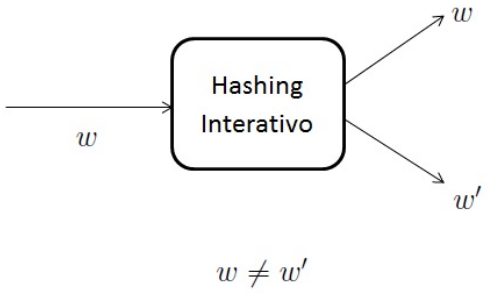

Figura 5.3: Hashing Interativo

Entretanto, neste trabalho o protocolo de *hashing* interativo será parecido com o primeiro, isto é, ele usa funções de *hash* para implementar um *hashing* interativo seguro. Este trabalho irá focar na variante de IH baseado na teoria de informação. Esta linha de pesquisa foi iniciado por (CACHIN; CREPEAU; MARCIL, 1998).

5.1.5.1 Segurança do *Hashing* Interativo

Um protocolo de IH é seguro para o emissor se, mesmo se o receptor trapaceia, ele não sabe qual das duas saídas do protocolo era a entrada do emissor. Por outro lado, será dito que o protocolo de IH é seguro para o receptor se, para qualquer estratégia que o emissor siga, ele não pode forçar uma propriedade particular em ambas as saídas. Isto significa que w e w' não podem simultaneamente pertencer à um conjunto específico S.

Formalmente tem-se a seguinte definição:

Definição 28 (Segurança do *Hashing* Interativo (DING et al., 2007))**.** *Um protocolo de* hashing *interativo é* seguro para o emissor *se para qualquer estratégia ilimitada do receptor (A') e todo W, se W_0, W_1 são as saídas do protocolo entre um emissor honesto com entrada W e A', então as distribuições $\{View_{A'}^{\langle A',B\rangle}(W)|W = W_0\}$ e $\{View_{A'}^{\langle A',B\rangle}(W)|W = W_1\}$ são idênticas, onde $View_{A'}^{\langle A',B\rangle}(W)$ é a visão do receptor do protocolo quando a entrada do emissor é W. Um protocolo de* hashing *interativo é (s, ρ)-*seguro para o receptor *se, para todo $S \subseteq \{0,1\}^m$ de tamanho no máximo 2^s e toda estratégia ilimitada do emissor (B'), se W_0, W_1 são as saídas do protocolo, então*

$$\Pr[W_0, W_1 \in S] < \rho. \tag{5.4}$$

Onde as probabilidades são tomadas sobre os lançamentos de moeda do emissor e do receptor. Um protocolo de hashing *interativo é (s, ρ)-*seguro *se é seguro para o emissor e (s, ρ)-*seguro para o receptor.

5.1.5.2 Protocolo de *Hashing* Interativo com Rodadas Constantes

O protocolo de IH apresentado aqui foi retirado de (DING et al., 2007, Seção 5.4). Uma das principais ferramentas usadas no protocolo de *hashing* interativo com rodadas constantes é uma permutação *η-almost t-wise* independente. Uma permutação *t-wise* independente π é tal que quando aplicado em qualquer t pontos em $\{0,1\}^m$, a permutação π se comporta como uma verdadeira permutação aleatória. [13] Em uma permutação *η-almost t-wise* independente π' , a distribuição em qualquer dos t pontos tem distância estatística no máximo η de uma distribuição induzida nestes pontos por uma permutação verdadeiramente aleatória. Uma permutação *2-wise* independente pode ser visto como uma permutação 0-*almost* 2-*wise* independente. Pode-se construí-la escolhendo $a, b \in_R GF(2^m)$, $a \neq 0$ (enquanto as strings em $\{0,1\}^m$ são identificadas com o corpo $GF(2^m)$) e definindo a permutação por $g(x) = ax + b$.

Outra ferramenta usada neste protocolo de IH é a *2-1 hash function. Seja h:* $\{0,1\}^m \rightarrow \{0,1\}^{m-1}$ *uma* 2-1 hash function. Então, para cada saída de h existem exatamente 2 pré-imagens. Observe que para construir uma *2-wise* independente *2-1 hash function*, pode-se tomar uma permutação *2-wise* independente e omitir o último bit de sua saída.

[13] A razão para não se usar uma permutação verdadeiramente aleatória é de sua descrição ser exponencial em m.

Então, tem-se uma forma de computar uma permutação *2-wise* independente *2-1 hash function* permutando polinômios sobre corpos finitos.

Para implementar o *hashing* interativo com rodadas constantes, configura-se o conjunto S ter cardinalidade $|S| = 2^s$. Lembre-se que S é o subconjunto de $\{0,1\}^m$ cujas strings têm alguma propriedade particular e a entrada do emissor é $w \in \{0,1\}^m$. Então, os parâmetros e ferramentas do protocolo de IH são:

- Os parâmetros são m e s, tais que m é o comprimento da string w e s é o parâmetro que define o tamanho do conjunto S;
- $t = m$ e $\eta = (\frac{1}{2^v})^t$, onde $v = s - \log m$;
- Uma família Π de permutações η-quase t-*wise* independentes π: $\{0,1\}^m \rightarrow \{0,1\}^m$;
- Uma família G de *2-wise* independentes *2-1 hash functions* g: $\{0,1\}^{m-v} \rightarrow \{0,1\}^{m-v-1}$;
- Uma família H (induzida por Π e G) de *2-1 hash functions* h: $\{0,1\}^m \rightarrow \{0,1\}^{m-1}$ definida como:

  $$h(x) \triangleq \pi(x)_1, ..., \pi(x)_v, g(\pi(x)_{v+1}, ..., \pi(x)_m);$$

  onde $\pi(x)_i$ denota o i-ésimo bit de $\pi(x)$.

Apresenta-se a seguir o protocolo que implementa *hashing* interativo com apenas quatro trocas de mensagens, independentes do comprimento da *string* que o remetente quer enviar para o destinatário. Seja Bob o emissor e Alice o receptor. Bob, neste caso tem como entrada a *string* $w \in \{0,1\}^m$.

Protocolo 5.1 Protocolo de *Hashing* Interativo

1: Alice escolhe $\pi \in_R \Pi$ e envia a descrição de π para Bob.
2: Bob calcula $\pi(w) = z_1...z_m$, onde z_i é o i-ésimo bit de $\pi(w)$, e envia os bits $z_1...z_v$ para Alice.
3: Alice escolhe $g \in_R G$ e envia a descrição de g para Bob.
4: Bob calcula e envia $g(z_{v+1}...z_m)$ para Alice.
5: Ambos calculam e tem como resultado (w_0, w_1) tal que $h(w) = h(w_0) = h(w_1)$.

Em (DING et al., 2007, Section 5.4) tem-se o seguinte teorema que determina os limites dos parâmetros de segurança para alcançar um *hashing* interativo (s, ρ)-*seguro*:

Teorema 7. *Para todo* s, m *tal que* $s \geq \log m + 2$*, o protocolo acima é um protocolo de* hashing *interativo* η'*-uniforme* $(s, 2^{-(m-s)+O(\log m)})$*-seguro para* $\eta' = (2^{s-\log m-1})^{-m} < 2^{-m}$.

5.2 O PROTOCOLO

Finalmente, apresenta-se nesta seção o protocolo para *String* OT baseado em um GEC que atinge a capacidade de OT no modelo malicioso. Ele é baseado no protocolo para *String* OT baseado em um canal com apagamentos de Savvides (SAVVIDES, 2007, Protocolo 5.1). Assume-se um adversário malicioso (ou ativo) que pode ter um comportamento arbitrário. Os jogadores são conectados através de um canal sem ruído e por um Canal com Apagamentos Generalizado (Definição 24).

Seja Alice o emissor e Bob o receptor. O objetivo do protocolo é Alice enviar strings aleatórias r_0 e r_1 para Bob que vai recuperar no máximo uma delas. As entradas de Alice são uma string aleatória escolhida aleatoriamente x^n e quatro funções de hash 2-universal g_0, g_1, h_0 e h_1 escolhidos aleatoriamente. As entradas de Bob são um bit de escolha c e uma string $w \in_R \{0,1\}^m$, onde w será decodificada em um conjunto S com cardinalidade $|S| = \alpha n$. No início do protocolo, ambos os jogadores concordam com os parâmetros de segurança ϵ e γ, onde αn é a porção de bits sacrificado para teste. Bob sabe a probabilidade p^* do GEC.

O protocolo funciona da seguinte maneira: Alice envia x^n através do GEC, cuja saída é y^n. Bob monta dois conjuntos disjuntos, um conjunto $\mathcal{B}$ contendo as posições de y^n em que ele recebeu um símbolo de apagamento (o bit foi transmitido por um canal com apagamentos) e um conjunto $\mathcal{G}$ contendo as posições de y^n em que ele recebeu um bit (o bit foi transmitido por um canal discreto sem memória). Bob aborta se ele não tem uma quantidade suficiente de posições em $\mathcal{G}$ para continuar o protocolo.

Bob também monta outros dois conjuntos $\mathcal{R}_c$ e $\mathcal{R}_{\bar{c}}$ com cardinalidade βn. O conjunto $\mathcal{R}_c$ possui elementos apenas em $\mathcal{G}$ e $\mathcal{R}_{\bar{c}}$ possui αn elementos em $\mathcal{G}$ e outros $(\beta - \alpha)n$ elementos, fora aqueles já escolhidos, em $\mathcal{G} \cup \mathcal{B}$. Observe que os bits de y^n que estão nas posições de $\mathcal{G}$ foram transmitidos por um canal discreto sem memória e, portanto, Bob sabe o valor destes bits da Alice com exceção daqueles que foram trocados pelo

canal de acordo com a matriz do canal W_0. Portanto, pode-se dizer que os αn bits que Bob conhece serão sacrificados para o teste de desvio de Bob. Em seguida, Bob envia a descrição de $\mathcal{R}_0$ e $\mathcal{R}_1$ para Alice que então pode confirmar que todas as posições foram escolhidas apenas uma vez. Note que Alice não sabe qual conjunto, ou $\mathcal{R}_0$ ou $\mathcal{R}_1$, é o conjunto formado apenas com as posições em $\mathcal{G}$ e qual é o conjunto com os bits que serão usados para teste.

Observe que para fazer o teste de desvio de Bob, ele tem que enviar para Alice a informação de quais posições de y^n serão usados no teste. Para tal, Bob utiliza o protocolo de *hashing* interativo, pois ele dá ao Bob a garantia de que Alice não aprenderá a bit de escolha c que está associada aos conjuntos $\mathcal{R}_c$ e $\mathcal{R}_{\overline{c}}$. Além disso, o IH auxilia na construção do próprio teste de desvio, uma vez que garante que Bob não pode controlar todas as posições que serão usadas no teste. Então, seja w a *string* que descreve quais as posições de $\mathcal{R}_{\overline{c}}$ serão usados para o teste. Bob envia w através de um protocolo seguro de *hashing* interativo. Lembre-se que ao final da execução de um IH, tanto Alice quanto Bob recebem duas strings w_0 e w_1, tal que $w_0 \neq w_1$, e existe um $b \in \{0,1\}$ tal que $w_b = w$ que é conhecido pelo Bob. Portanto, pelas propriedades de segurança do IH, Alice recebe as posições escolhidas por Bob para teste e outro conjunto com αn posições de teste que não está sobre o controle de Bob.

Até então Bob enviou apenas as posições para Alice. A partir deste ponto, ele envia os bits recebidos que estão nas posições selecionadas para teste: os bits que Bob escolheu $y^{\mathcal{R}_1^{\mathcal{S}_a}} = y^{\mathcal{R}_{\overline{c}}^{\mathcal{S}_b}}$ e os bits que estão nas posições determinadas pela outra saída do IH $y^{\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}} = y^{\mathcal{R}_c^{\mathcal{S}_{\overline{b}}}}$. Observe que para "esconder"o bit de escolha, Bob faz $a = b \oplus c$ que é aleatório do ponto de vista da Alice, desde que o protocolo de IH é seguro.

Desde que o canal discreto sem memória associado ao GEC introduz erros, se faz necessário saber se a string recebida por Bob y^n possui uma quantidade de erros de forma a incapacitar a recuperação de x^n a partir de y^n. Para tal, Alice testa se $y^{\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}}$ e $y^{\mathcal{R}_1^{\mathcal{S}_a}}$ são conjuntamente típicas com a entrada dela nestas posições. Se elas não são típicas, então y^n contém muitos erros e o protocolo é abortado.

Para Alice corrigir os erros em y^n, ela calcula e envia $g_0(x^{\mathcal{R}_0})$, $g_1(x^{\mathcal{R}_1})$ juntamente com as descrições de g_0 e g_1 para Bob. Ele então pode computar todos as possíveis sequências $\widetilde{x}^{\mathcal{R}_c}$ que são conjuntamente típicas com $y^{\mathcal{R}_c}$ e satisfaz $g_c(\widetilde{x}^{\mathcal{R}_c}) = g_c(x^{\mathcal{R}_c})$. Se existir apenas uma sequência obedecendo essas restrições então diz-se que Bob recuperou a *string* enviada por Alice no início do protocolo $\widetilde{x}^{\mathcal{R}_c}$. Observe que como

Bob utiliza a saída de uma função de *hash* 2-universal para corrigir erros, o protocolo em questão não é computacionalmente eficiente. Contudo, ele é suficiente para mostrar que a capacidade de OT para GEC no modelo malicioso.

Depois de descartar os αn bits usados no teste de desvio de Bob, os $\beta n[H(X|Y \in \mathcal{Y}_0)+\epsilon]$ bits usados para corrigir os erros e outros $\beta n + H(X)[\beta n - 5\alpha n] + \gamma n$ bits necessários para atingir a segurança do protocolo, Alice pode extrair as strings aleatórias r_0 e r_1 que são de fato o que é transmitido no protocolo de OT. Ela calcula $r_0 = h_0(x^{\mathcal{R}_0})$ e $r_1 = h_1(x^{\mathcal{R}_1})$ e os envia juntamente com a descrição de h_0, h_1 para Bob. Finalmente, Bob recebe a *string* aleatória desejada $r_c = h_c(\widetilde{x}^{\mathcal{R}_c})$. A figura 5.4 resume os passos de execução do protocolo em estudo.

A seguir analisa-se a segurança do protocolo, de forma a mostrar que o protocolo é correto, seguro para Alice e seguro para Bob. Dito de outra forma, será provado o seguinte teorema:

Teorema 8. *O Protocolo 5.2 é seguro segundo a Definição 25.*

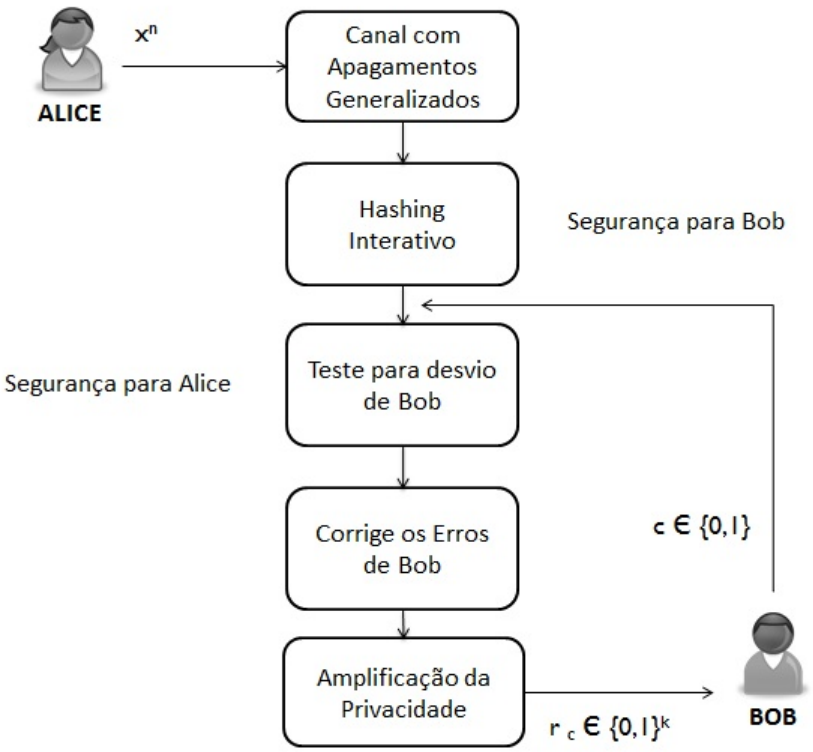

Figura 5.4: Resumo dos passos seguidos no protocolo para *String* OT

Protocolo 5.2 Protocolo de *Oblivious Transfer*

1: Alice e Bob selecionam uma (tipicamente muito pequena) constante positiva $\alpha < \frac{1-p^*}{7}$ e fazem $\beta = 1 - p^* - 2\alpha$.

2: Alice aleatoriamente escolhe x^n de acordo com a distribuição de probabilidade que atinge a capacidade de Shannon do canal W_0 e envia x^n através do GEC.

3: Bob recebe a *string* y^n e coleciona as *boas* (aquelas correspondentes à $y \in \mathcal{Y}_0$) e as *más* (aquelas correspondentes à $y \in \mathcal{Y}^*$) posições nos conjuntos $\mathcal{G}$ e $\mathcal{B}$, respectivamente. Ele aborta se $|\mathcal{G}| < (1 - p^* - \alpha)n = \beta n + \alpha n$.

4: Bob escolhe $c \in_R \{0,1\}$ e $w \in_R \{0,1\}^m$, onde $m = \lceil \log \binom{\beta n}{\alpha n} \rceil$. Ele decodifica w em um subconjunto $\mathcal{S}$ de cardinalidade αn (fora dos βn) usando o esquema de codificação da Seção 5.1.4. Bob então define dois conjuntos disjuntos $\mathcal{R}_c$ e $\mathcal{R}_{\overline{c}}$ de cardinalidade βn. $\mathcal{R}_c$ consiste apenas de posições de $\mathcal{G}$, escolhidos aleatoriamente e sem repetição. $\mathcal{R}_{\overline{c}}$ tem αn posições de $\mathcal{G}$ (definindo o subconjunto $\mathcal{R}_{\overline{c}}^{\mathcal{S}}$) e as posições restantes são escolhidas de $\mathcal{G} \cup \mathcal{B}$, aleatoriamente e sem repetição. Bob envia as descrições de $\mathcal{R}_0$ e $\mathcal{R}_1$ para Alice.

5: Alice confirma que nenhuma posição está repetida nos conjuntos $\mathcal{R}_0$ e $\mathcal{R}_1$, de outra forma ela aborta.

6: Bob envia w para Alice usando o protocolo de *Hashing* Interativo (Protocolo 5.1). Seja w_0, w_1 as strings de saída, seja $\mathcal{S}_0$, $\mathcal{S}_1$ os correspondentes subconjuntos de cardinalidade αn e deixe $b \in \{0,1\}$ ser tal que $w_b = w$.

7: Bob anuncia $a = b \oplus c$ assim como $y^{\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}}$ e $y^{\mathcal{R}_1^{\mathcal{S}_a}}$.

8: Alice confirma se $y^{\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}}$ e $y^{\mathcal{R}_1^{\mathcal{S}_a}}$ são 2ϵ conjuntamente típicos para o canal discreto sem memória $\{W_0 : \mathcal{X} \to \mathcal{Y}_0\}$ com a entrada nela nessas posições. Se eles não são conjuntamente típicos, Alice aborta.

9: Alice escolhe aleatoriamente funções de *hash* 2-universal $g_0, g_1 : \mathcal{X}^{\beta n} \to \{0,1\}^{\beta n[H(X|Y \in \mathcal{Y}_0)+\epsilon]}$ (com $\epsilon > 0$ tal que o comprimento de saída é inteiro). Ela computa $g_0(x^{\mathcal{R}_0})$ e $g_1(x^{\mathcal{R}_1})$. Ela também aleatoriamente escolhe funções de *hash* 2-universal $h_0, h_1 : \mathcal{X}^{\beta n} \to \{0,1\}^{\delta n}$, onde $\delta = (\beta - 5\alpha)H(X) - \beta(H(X|Y \in \mathcal{Y}_0) + \epsilon) - \gamma$ e $\gamma > 0$ tal que o comprimento de saída é inteiro. Ela envia $g_0(x^{\mathcal{R}_0})$, $g_1(x^{\mathcal{R}_1})$ e as descrições de g_0, g_1, h_0, h_1 para Bob. Alice tem como resposta $r_0 = h_0(x^{\mathcal{R}_0})$ e $r_1 = h_1(x^{\mathcal{R}_1})$.

10: Bob computa todas as possíveis $\widetilde{x}^{\mathcal{R}_c}$ que são conjuntamente típicas com $y^{\mathcal{R}_c}$ e satisfazem $g_c(\widetilde{x}^{\mathcal{R}_c}) = g_c(x^{\mathcal{R}_c})$. Se existe exatamente um tal que $\widetilde{x}^{\mathcal{R}_c}$, Bob tem como resultado $r_c = h_c(\widetilde{x}^{\mathcal{R}_c})$. De outra forma, ele tem como resultador$_c = 0^{\delta n}$.

5.2.1 Prova de Segurança Baseada em Teoria da Informação

Será provado a seguir o Teorema 8. Segundo a Definição 25, o protocolo para *String* OT será seguro se as propriedades de correção, segurança para o emissor (Alice) e segurança para o receptor (Bob) forem satisfeitas.

Correção. Quando Alice e Bob são honestos, Bob só não irá receber a saída correta se ele abortar no passo 3 ou se ele não obter exatamente um $\widetilde{x}^{\mathcal{R}_c} = x^{\mathcal{R}_c}$ no passo 10.

Primeiro será analisado o passo 3. Bob aborta se $|\mathcal{G}| < (1 - p^* - \alpha)n$. Lembre-se que $y^n = \{y_1, y_2, \ldots, y_n\}, \quad y_i \in \{0, 1\}$ e $Pr[y \in \mathcal{Y}_0] = 1 - p^*$. Então, o valor esperado de $y_i \in \mathcal{Y}_0$ é $E(y_i | y_i \in \mathcal{Y}_0) = (1 - p^*)p$, onde $p = Pr[y_i = 1]$. Fazendo $\eta = \left\{1 - \frac{1}{p} + \frac{\alpha}{(1-p^*)p}\right\}$ tem-se pelo limite de Chernoff (Lema 2 Capítulo 2) que a probabilidade de Bob abortar no passo 3 é dado por:

$$Pr\left\{\frac{1}{n}\sum_{i=1}^{n} y_i \leq 1 - p^* - \alpha\right\} \leq e^{-(1-p^*)n}. \tag{5.5}$$

Dessa forma, o evento $|\mathcal{G}| < (1 - p^* - \alpha)n$ ocorre com probabilidade $\epsilon < e^{-(1-p^*)n}$, que é uma função negligível de n.

Falta analisar agora o passo 10. Bob não encontrará exatamente $\widetilde{x}^{\mathcal{R}_c} = x^{\mathcal{R}_c}$ se $x^{\mathcal{R}_c}$ não é conjuntamente típica com $y^{\mathcal{R}_c}$ ou se existe outra sequência $\overline{x}^{\mathcal{R}_c}$ que é também conjuntamente típica com $y^{\mathcal{R}_c}$ e tem $g_c(\overline{x}^{\mathcal{R}_c}) = g_c(x^{\mathcal{R}_c})$. Entretanto, o item 1 do Teorema 2 assegura que $Pr((x^{\mathcal{R}_c}, y^{\mathcal{R}_c}) \in A_\epsilon^n) \leq \varepsilon$ quando $n \to \infty$. Então, a probabilidade de $x^{\mathcal{R}_c}$ não ser conjuntamente típica com $y^{\mathcal{R}_c}$ é negligível com n. Por outro lado, o item 2 do Teorema 2 diz qie o número de sequências $\overline{x}^{\mathcal{R}_c}$ que são conjuntamente típicas com $y^{\mathcal{R}_c}$ é $|A_\epsilon^n| \leq 2^{\beta n[H(X|Y \in \mathcal{Y}_0) + \epsilon']}$ para $0 < \epsilon' < \epsilon$ e n suficientemente grande. Finalmente, pelo *Leftover-Hash Lemma* tem-se que a probabilidade de uma sequência $\overline{x}^{\mathcal{R}_c}$ ser conjuntamente típica com $y^{\mathcal{R}_c}$ e ter $g_c(\overline{x}^{\mathcal{R}_c}) = g_c(x^{\mathcal{R}_c})$ é dada por:

$$\frac{2^{\beta n[H(x|y \in \mathcal{Y}_0) + \epsilon']}}{2^{\beta n[H(x|y \in \mathcal{Y}_0) + \epsilon]}} = 2^{\beta n(\epsilon' - \epsilon)} \to 0 \quad \text{quando} \quad n \to \infty. \tag{5.6}$$

Como as probabilidades de Bob abortar nos passos 3 e 10 são negligíveis em n, tem-se para o Protocolo 5.2 que:

$$Pr[V = B_C] \geq 1 - \epsilon. \tag{5.7}$$

Segurança para o Receptor. Lembre-se que a visão de um jogador é dado por toda a informação em posse de um jogador ao final do protocolo incluindo os resultados de todas computações locais, amostragens de aleatoriedade local, entradas locais e mensagens trocadas. A propriedade de segurança do receptor (Bob) requer que se prove que ao fim do protocolo a probabilidade de Alice ter qualquer informação, além da sua visão, sobre o bit de escolha c de Bob dado que a entrada B de Alice no protocolo é negligível. Observe que Alice apenas pode obter informação sobre c quando Bob envia à ela as descrições de $\mathcal{R}_0$ e $\mathcal{R}_1$ no passo 4 e quando Bob envia $a = b \oplus c$, $y^{\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}}$ e $y^{\mathcal{R}_1^{\mathcal{S}_a}}$ no passo 7.

Para analisar o passo 4, observe que no GEC todo símbolo de entrada x é apagado (isto é, o símbolo de saída está em $\mathcal{Y}^*$) com probabilidade p^* independente de x. Então, Alice não sabe qual dos símbolos de entrada foram apagados pelo canal e a distribuição de $(\mathcal{R}_0, \mathcal{R}_1)$ é independente de c a partir da visão da Alice. Portanto, Alice não pode dizer se $\mathcal{R}_c = \mathcal{R}_0$ ou $\mathcal{R}_c = \mathcal{R}_1$.

No passo 7, Alice só pode extrair o valor de c através de a se ela sabe o valor de b. Contudo, se o protocolo de *Hashing* Interativo é seguro, Alice não sabe o valor de b, pois o ponto de vista de Alice é o mesmo para $b = 0$ e $b = 1$.

Observe que não importa o que uma Alice maliciosa de fato envia no passo 9, Bob não abortará. Em particular, isso previne ataques de reação, isto é, ao receber os bits de Bob no passo anterior, uma Alice maliciosa não terá nada a ganhar se ela tenta trapacear enviando alguma função diferente que pode depender dos bits recebidos no lugar de g_0, g_1, h_0, h_1.

Portanto, a distribuição da visão de Alice não depende de c. Então, dado uma pequena constante positiva $\varepsilon = \varepsilon(p^*, \eta')$, onde η' provém do *Hashing* Interativo (veja Seção 5.1.5), tem-se:

$$I_S(C; U|B) = H(C|B) - H(C|U, B) \leq \varepsilon. \tag{5.8}$$

Segurança para o Emissor. Essa prova segue a de Savvides (SAVVIDES, 2007, Seção 5.1). Para fornecer segurança para Alice (emissor), tem-se que provar que um Bob malicioso não pode completar o protocolo sem ser pego com grande probabilidade e se ela desvia do protocolo um pouco, ele terá informações sobre ambas as entradas de Alice com uma probabilidade negligível.

Observe que Bob tem cerca de $(1 - p^*)n$ posições boas a menos de uma probabilidade negligível em n (veja a prova de correção acima). Portanto, um Bob malicioso pode seguir duas estratégias:

1. Bob usa posições boas em ambos conjuntos R_0 e R_1 tanto quanto possível. Isso implica que ambos conjuntos terão muitas posições más;
2. Bob usa poucas posições boas em um dos conjuntos R_0, R_1 de forma a completar o protocolo e tentar reunir informações sobre ambas as entradas de Alice.

Será visto a seguir que ambas as estratégias não dão informações para Bob sobre as duas entradas de Alice. Antes, será apresentado algumas definições.

Definição 29. *Seja $u(\mathcal{R})$ o número de posições contidas em $\mathcal{R}$ tal que a saída correspondente naquelas posições eram símbolos de apagamento.*

Definição 30. *$\mathcal{S}$ é chamada de* boa *para $\mathcal{R}$ se $u(\mathcal{R}^{\mathcal{S}}) < \alpha^2 n$, de outra forma ela é chamada de* má *para $\mathcal{R}$.*

Com as definições expostas acima, pode-se reduzir as duas estratégias de um Bob malicioso para: (i) tanto $u(\mathcal{R}_0)$ quanto $u(\mathcal{R}_1)$ são maiores ou iguais a $2\alpha n$, (ii) ou $u(\mathcal{R}_0)$ ou $u(\mathcal{R}_1)$ é menor que $2\alpha n$.

Para provar o primeiro caso será necessário os dois lemas seguintes provenientes de (SAVVIDES, 2007, Seção 5.1). O lema 3 segue do limite de Chernoff (Lema 2 do Capítulo 2) e das propriedades de GEC. Já o lema 4 segue do lema 3, do *union bound* e das propriedades do esquema de codificação usado.

Lema 3. *Seja $\mathcal{R}$ um conjunto de cardinalidade βn tal que $u(\mathcal{R}) \geq 2\alpha n$. Então, a fração f de subconjuntos $\mathcal{S}$ de cardinalidade αn que são bons para $\mathcal{R}$ satisfaz $f < e^{-\alpha^2 n/4}$.*

Lema 4. *Sejam $\mathcal{R}_0, \mathcal{R}_1$ conjuntos de cardinalidade βn tal que $u(\mathcal{R}_0) \geq 2\alpha n$ e $u(\mathcal{R}_1) \geq 2\alpha n$. Então a fração de strings w que decodifica para subconjuntos $\mathcal{S}$ que são bons para ou $\mathcal{R}_0$ ou $\mathcal{R}_1$ não é maior que $4e^{-\alpha^2 n/4}$.*

Estratégia 1: Desde que a fração de strings $w \in_R \{0,1\}^m$ que são boas para ou $\mathcal{R}_0$ ou $\mathcal{R}_1$ não é maior do que $4e^{-\alpha^2 n/4}$, pode-se configurar o parâmetro de segurança s do protocolo de *hashing* interativo para $\log(4e^{-\alpha^2 n/4}2^m) = m - \frac{\alpha^2 n}{4}\log e + 2$.

Tem-se também $m = \lceil \log \binom{\beta n}{\alpha n} \rceil \leq \lceil \log 2^{\beta n} \rceil = \lceil \beta n \rceil = O(n)$. Portanto,

$$\rho = 2^{-(m-s)+O(\log m)} = 2^{-\alpha^2 \log(e) n/4 + O(\log n)} \tag{5.9}$$

e então, pela segurança do protocolo de *hashing* interativo, a probabilidade de que Bob tenha tanto w_0 quanto w_1 bons para ou $\mathcal{R}_0$ ou $\mathcal{R}_1$ é a seguinte:

$$Pr[w_0, w_1 \in S] < \rho = 2^{-\alpha^2 \log(e) n/4 + O(\log n)} \tag{5.10}$$

que é uma função negligível de n.

Então, com grande probabilidade um dos conjuntos, sem perda de generalidade $\mathcal{R}_0$, terá $u(\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}) \geq \alpha^2 n$, i.e., $\mathcal{R}_c^{\mathcal{S}_{\overline{b}}}$ terá mais que $\alpha^2 n$ posições que correspondem a um símbolo de apagamento.

Apresenta-se a seguir um lema que prova que se as sequências de n bits da entrada e da saída de um canal são ϵ-conjuntamente típicos, então essas sequências de entrada e saída restritas à posições de um determinado conjunto finito de cardinalidade $\delta n, 0 < \delta < 1$ também são ϵ'-conjuntamente típicos. Para a demonstração do lema utiliza-se principalmente a Definição 10 do Capítulo 2 de sequências conjuntamente típicas.

Lema 5. *Sejam $W : \mathcal{X} \to \mathcal{Y}$ um canal e $x^n \in \mathcal{X}^n$ e $y^n \in \mathcal{Y}^n$ as strings de entrada e saída, respectivamente, deste canal. Seja $\mathcal{C}$ um subconjunto aleatório de $[n]$ tal que $|\mathcal{C}| = \delta n$, $0 < \delta < 1$ e $x^{\mathcal{C}}$ e $y^{\mathcal{C}}$ as restrições de x^n e y^n para as posições no conjunto $\mathcal{C}$. Se x^n e y^n são ϵ-condicionalmente típicos, então $x^{\mathcal{C}}$ e $y^{\mathcal{C}}$ são 2ϵ-condicionalmente típicos para qualquer $\epsilon > 0$ e n suficientemente grande.*

Demonstração. Por hipótese, x^n e y^n são ϵ-condicionalmente típicos. Então, para todos os símbolos x e y tem-se que

$$|N(xy|x^n y^n) - np_{x^n}(x)W(y|x)| \leq \epsilon n, \tag{5.11}$$

para um n suficientemente grande.

Dado as *strings* ϵ-típicas condicionais x^n e y^n, a probabilidade de selecionar um par com valores específicos x e y para as *substrings* $x^{\mathcal{C}}$ e $y^{\mathcal{C}}$ é dado por $\frac{N(xy|x^n y^n)}{n}$. Então,

tem-se:

$$p_{x^n}(x)W(y|x) - \epsilon \leq \frac{N(xy|x^n y^n)}{n} \leq p_{x^n}(x)W(y|x) + \epsilon. \tag{5.12}$$

Portanto, pelo limite de Chernoff (Lema 2 do Capítulo 2), para n grande o suficiente com alta probabilidade o número de pares de x e y nas *substrings* $x^{\mathcal{C}}$ e $y^{\mathcal{C}}$, $N(xy|x^{\mathcal{C}}y^{\mathcal{C}})$, é limitada da seguinte forma:

$$\delta n(p_{x^n}(x)W(y|x) - \epsilon - \epsilon') \leq \frac{N(xy|x^{\mathcal{C}}y^{\mathcal{C}})}{n} \leq \delta n(p_{x^n}(x)W(y|x) + \epsilon + \epsilon'), \tag{5.13}$$

para $\epsilon' > 0$.

Fazendo $\epsilon' = \epsilon$ tem-se que as *substrings* $x^{\mathcal{C}}$ e $y^{\mathcal{C}}$ são 2ϵ-condicionalmente típicas. □

Com o Lema 5 tem-se que se $y^{\mathcal{R}_0^{S_{\bar{a}}}}$ e $x^{\mathcal{R}_0^{S_{\bar{a}}}}$ não são ϵ'-conjuntamente típicos, então y^n e x^n também não são ϵ-conjuntamente típicos para n suficientemente grande. Portanto, Bob tem sucesso no teste do passo 8 apenas se ele advinha corretamente os valores de y para estas posições que são conjuntamente típicas com a entrada da Alice.

Para n suficiente grande, o Teorema 3 garante que existem no máximo $2^{\alpha^2 n[H(Y\in\mathcal{Y}_0|X)+\epsilon]}$ ($\epsilon > 0$) sequências de valores de y que são conjuntamente típicas com a entrada de Alice. Entretanto, pelo item 2 do Teorema 1, existem no mínimo $2^{\alpha^2 n[H(Y\in\mathcal{Y}_0)-\epsilon]}$ sequências típicas para valores de y. Então, a probabilidade de Bob ter sucesso no teste é menor do que $2^{\alpha^2 n[H(Y\in\mathcal{Y}_0|X)-H(Y\in\mathcal{Y}_0)+2\epsilon]} = 2^{-\alpha^2 n[C(W_0)-2\epsilon]}$, que é uma função negligível de n.

Em outras palavras, um Bob malicioso deve adivinhar corretamente toda a informação que um Bob honesto teria recebido se essas posições fossem boas. Então, a probabilidade de Bob trapacear com sucesso, ζ, quando ambos $u(\mathcal{R}_0)$ e $u(\mathcal{R}_1)$ são no mínimo $2\alpha n$ é uma função negligível de n:

$$\zeta = 2^{-\alpha^2 n[H(Y\in\mathcal{Y}_0)+H(Y\in\mathcal{Y}_0|X)+2\epsilon+\frac{log(\epsilon)}{4}]+O(n)} \tag{5.14}$$

Estratégia 2: Para o caso em que ou $u(\mathcal{R}_0)$ ou $u(\mathcal{R}_1)$ é menor do que $2\alpha n$, assume-se, sem perda de generalidade, que $u(\mathcal{R}_0) < 2\alpha n$.

Dado o limite de Chernoff (lema 2 do Capítulo 2) e fazendo $\eta = \left\{\frac{(p^*-\alpha)}{(1-p^*)p} - 1\right\}$, tem-se que a probabilidade de $|\mathcal{B}| > (p^* - \alpha)n$ é dado da seguinte forma:

$$Pr\left\{\frac{1}{n}\sum_{i=1}^{n} y_i \leq p^* - \alpha\right\} \leq \exp\left(-\frac{n}{2\ln 2}\frac{p^*}{(1-p^*)}\frac{(p^*-\alpha)}{p}\right) = \vartheta. \tag{5.15}$$

Desde que apenas $(1-2\beta)n$ posições não foram usadas em $\mathcal{R}_0, \mathcal{R}_1$, então $u(\mathcal{R}_0) + u(\mathcal{R}_1) + (1-2\beta)n > (p^* - \alpha)n$. Portanto, $u(\mathcal{R}_1) > (1 - p^* - 7\alpha)n = \beta n - 5\alpha n$.

Observe que mais do que $\beta n - 5\alpha n$ posições de $\mathcal{R}_1$ correspondem à símbolos de apagamentos e Alice envia apenas $\beta n[H(X|Y \in \mathcal{Y}_0) + \epsilon]$ bits de informação sobre $x^{\mathcal{R}_1}$ no passo 9. Portanto, tem-se que $H_\infty(X^{\mathcal{R}_1}|\text{View}_{\text{Bob}}) > n[(\beta - 5\alpha)H(X) - \beta H(X|Y \in \mathcal{Y}_0) - \beta\epsilon]$, onde View_{Bob} denota a visão de Bob. Então a propriedade da função de *hash* 2-univeral h_1 para extrair $n[(\beta - 5\alpha)H(X) - \beta H(X|Y \in \mathcal{Y}_0) - \beta\epsilon - \gamma], \gamma > 0$ bits de informação segue do *Leftover-Hash Lemma*. Logo, a probabilidade de Bob obter alguma informação sobre r_1 é dada por:

$$\xi < \frac{1}{2} 2^{-\frac{\gamma n}{2}} \tag{5.16}$$

que é negligível em n. Portanto, tem-se que:

$$I_S(B; C'|C) \leq \zeta \tag{5.17}$$

e

$$I_S(B; V|C, C', B_{C'}) \leq \xi. \tag{5.18}$$

Dessa forma, tem-se que o Protocolo 5.2 satisfaz as condições de correção, segurança para o emissor e segurança da Definição 25.

5.2.2 Atingindo a Capacidade de *Oblivious Transfer*

Para definir a capacidade de *oblivious transfer* do GEC no modelo malicioso usa-se a noção de *Capacidade de Canal*. A capacidade de Shannon de um canal é o logaritmo do número máximo de sinais distinguíveis para n usos do canal de comunicação e, operacionalmente, a capacidade de um canal é a maior taxa de bits por uso do canal na qual a informação pode ser enviada com probabilidade de erro arbitrariamente baixo. Precisamente, a capacidade de Shannon de um canal discreto sem memória é definido como $C = \max_{p(x)} I(x; y)$, onde o máximo é tomado sobre todas as distribuições de entrada $p(x)$ possíveis e $I(.;.)$ é a informação mútua de Shannon.

Com a definição de capacidade de canal discreto sem memória acima, apresenta-se o seguinte teorema:

Teorema 9. *Para um Canal com Apagamentos Generalizado com* $p^* \geq \frac{1}{2}$*, a capacidade de* oblivious transfer *no caso de adversários maliciosos é* $(1 - p^*)C(W_0)$ *onde* $C(W_0)$ *é a capacidade de Shannon do canal discreto sem memória* $\{W_0 : \mathcal{X} \to \mathcal{Y}_0\}$.

De acordo com a Definição 27, o Teorema acima será provado em duas partes. Na primeira parte (parte direta), prova-se que o Protocolo 5.2 possui a taxa de *oblivious transfer* de $(1-p^*)C(W_0)$. Logo, essa é uma taxa alcançável uma vez que o protocolo foi provado seguro no modelo malicioso (Seção 5.2.1). Na segunda parte (parte inversa), prova-se que a maior taxa de OT quando o esquema de *oblivious transfer* é implementado baseado em um GEC é $(1-p^*)C(W_0)$. Logo, a taxa de $(1-p^*)C(W_0)$ é a capacidade de OT desde que existe um protocolo de OT seguro que a alcança.

5.2.2.1 Parte direta:

Deseja-se demonstrar o seguinte teorema:

Teorema 10. *Existem constantes positivas arbitrariamente pequenas ϵ, α e γ tais que usando um Canal com Apagamentos Generalizado com probabilidade $p^* \geq \frac{1}{2}$, para n suficientemente grande, o Protocolo 5.2 atinge a taxa de OT de $(1-p^*)C(W_0)$, onde $C(W_0)$ é a capacidade de Shannon para o canal discreto sem memória $\{W_0 = \mathcal{X} \to \mathcal{Y}_0\}$.*

Prova do Teorema 10 . Verifica-se primeiramente os parâmetros do Protocolo 5.2. α indica a proporção dos bits sacrificados no passo 8. Desde que Alice e Bob selecionam $\alpha < \frac{1-p^*}{3}$ no início do protocolo, α pode ser escolhido arbitrariamente pequeno. Similarmente, os parâmetros ϵ e γ podem ser escolhidos arbitrariamente pequenos desde que ϵ é negligível em n e γ é a proporção dos bits que são desconsiderados (além dos $\beta[H(X|Y \in \mathcal{Y}_0) + \epsilon]$ bits enviados pela Alice para corrigir os erros na *string* recebida por Bob através do GEC) por segurança quando se extrai as strings aleatórias r_0, r_1. Observe que γ pode ser escolhido arbitrariamente pequena sem que isso comprometa a segurança do protocolo.

Desde que no passo 2 Alice pode escolher x^n de acordo com a distribuição de probabilidade que atinge a capacidade de Shannon de W_0, tem-se que o comprimento das strings r_0, r_1 transmitidas através do protocolo de OT quando $n \to \infty$ é

$$\begin{aligned} l &= n[(\beta - 5\alpha)H(X) - \beta(H(X|Y \in \mathcal{Y}_0) + \epsilon') - \gamma] \\ &= n\beta[H(X) - H(Y \in \mathcal{Y}_0)] \\ &= n(1-p^*)[H(X) - H(X|Y \in \mathcal{Y}_0)] \\ &= n(1-p^*)I(X; Y \in \mathcal{Y}_0) \\ &= n(1-p^*)C(W_0) \end{aligned}$$

Como a taxa de OT é definida como a razão entre o comprimento da *string* transmitida pelo protocolo de OT (l) e o número de vezes que o canal ruidoso foi usado para tal (n), tem-se que:

$$R_{OT} = (1 - p^*)C(W_0). \tag{5.19}$$

□

5.2.2.2 Parte inversa:

Como na parte direta, deseja-se demonstrar o seguinte teorema:

Teorema 11. *A capacidade OT de um protocolo de OT seguro implementado com base em um GEC com probabilidade* $p^* \geq \frac{1}{2}$ *é* $C_{OT} \leq (1 - p^*)C(W_0)$.

A prova desse teorema segue a prova do Teorema 1 de Ahlswede and Csiszár (AHLSWEDE; CSISZAR, 2007). Além disso, será utilizado o lema 1 de (AHLSWEDE; CSISZAR, 2007).

Lema 6 ((AHLSWEDE; CSISZAR, 2007))**.** *Para variáveis aleatórias* U, V, Z *com valores nos conjuntos finitos* $\mathcal{U}, \mathcal{V}, \mathcal{Z}$*, e qualquer* z_0, z_1 *em* $\mathcal{Z}$ *com* $Pr\{Z = z_0\} = p > 0$*,* $Pr\{Z = z_1\} = q > 0$*,*

$$|H(U|V, Z = z_0) - H(U|V, Z = z_1)| \leq c\sqrt{I(UV;Z)}\log|\mathcal{U}| + h\left(\min\left[c\sqrt{I(UV;Z)}, \frac{1}{2}\right]\right)$$

onde $h(t) = -t\log t - (1 - t)\log(1 - t)$ *e* c *é uma constante que depende de* p *e* q.

Prova do Teorema 11 . A ideia é provar que $\frac{k}{n} \leq C(W_0)(1 - p^*)$, onde o GEC é usado n vezes para que seja transmitido uma string de tamanho k em um protocolo de OT seguro. Serão usadas as seguintes variáveis aleatórias nesta prova:

- $K_0, K_1 \in \{0, 1\}^k$ são as strings transmitidas por um protocolo de OT;
- X^n é a entrada do emissor no GEC;
- Y^n é a saída do receptor no GEC;
- Z é o bit de escolha do receptor;

- $\hat{K}_Z$ é a saída estimada do receptor no protocolo de OT; e
- F é a comunicação total no canal sem ruído.

Considere sem perda de generalidade que o bit de escolha do receptor é $Z = 0$. Para esta prova pode-se usar as seguintes condições de segurança para um protocolo de OT:

$$Pr\{\hat{K}_0 \neq K_0 | Z = 0\} \to 0 \tag{5.20}$$

$$\frac{1}{k} I(Y^n F, K_0 | Z = 1) \to 0 \tag{5.21}$$

$$I(K_0 X^n F; Z) \to 0 \tag{5.22}$$

Observe que essas condições são mais fracas do que aquelas da Definição 25. Fazendo uma comparação entre elas, a equação (5.20) é a correção, a equação (5.21) é a segurança para o emissor e equação (5.22) é a segurança para o receptor.

Tese 1.

$$\frac{k}{n} \leq \frac{1}{n} \sum_{t=1}^{n} I(X_t; Y_t) + \varepsilon, \quad \varepsilon \to 0.$$

Demonstração da Tese 1. A partir do Lema 6 e fazendo $I = I(K_0 X^n F; Z)$ tem-se que:

$$|H(K_0 | X^n F, Z = z_0) - H(K_0 | X^n F, Z = z_1)| \leq c\sqrt{I} \log k + h\left(\min\left[c\sqrt{I}, \frac{1}{2}\right]\right) \tag{5.23}$$

Entretanto, pela equação (5.22) tem-se que $I = I(K_0 X^n F; Z) = \delta$, então $h\left(\min\left[c\sqrt{I}, \frac{1}{2}\right]\right) = h(\delta') = h(\delta")$ e $c\sqrt{I}\log k = o(k)$. Desde que condicionar reduz a entropia, segue que:

$$\begin{aligned} & H(K_0 | X^n F, Z = 0) - H(K_0 | X^n F, Z = 1) = o(k) \\ \Rightarrow \quad & H(K_0 | F, Z = 0) - H(K_0 | F, Z = 1) = o(k) \end{aligned} \tag{5.24}$$

Por outro lado, tem-se que $H(K_0 | Z = 0) = H(K_0 | Z = 1) = H(K_0) = k$, pois K_0 é independente de Z. Juntamente com a equação 5.24, tem-se:

$$\begin{aligned} I(K_0; F | Z = 0) &= H(K_0 | Z = 0) - H(K_0 | FZ = 0) \\ &= k + o(k) + H(K_0 | F, Z = 1) \end{aligned}$$

Para encontrar $H(K_0 | F, Z = 1)$, desenvolve-se a condição (5.21):

$$\begin{aligned} I(Y^n F; K_0 | Z = 1) &= k\vartheta \\ I(Y^n F; K_0 | Z = 1) &= I(F; K_0 | Z = 1) + I(Y^n; K_0 | FZ = 1) \\ I(F; K_0 | Z = 1) &= I(Y^n F; K_0 | Z = 1) - I(Y^n; K_0 | FZ = 1) \\ I(F; K_0 | Z = 1) &= o(k) \end{aligned}$$

desde que, se o protocolo é seguro, o receptor não consegue informação da outra entrada do emissor além daquela obtida pelo bit de escolha ($I(Y^n; K_0|FZ = 1) \to 0$). Então, $H(K_0|F, Z = 1) = H(K_0) - I(K_0, F|Z = 1) = k - o(k) \Rightarrow H(K_0|F, Z = 1) = o(k)$

Logo,

$$\begin{aligned} I(K_0; F|Z = 0) &= k + o(k) + o(k) \\ I(K_0; F|Z = 0) &= o(k) \end{aligned} \tag{5.25}$$

Se as equações (5.20) e (5.25) se mantêm sem o condicionamento em $Z = 0$, então K_0 pode ser visto como uma chave secreta entre o emissor e o receptor dentro de um protocolo de acordo de chave secreta seguro quando um adversário observa a comunicação pública F. Então, pelo Teorema 3 de Maurer em (MAURER, 1993) a taxa k/n de tal chave secreta é assintoticamente limitada por

$$\frac{k}{n} \leq \frac{1}{n} \sum_{t=1}^{n} I(X_t; Y_t) + \varepsilon, \quad \varepsilon \to 0.$$

De fato, pode-se considerar $I(X_t; Y_t)$ no lugar de $I(X_t; Y_t|Z = 0)$ desde que $I(K_0; F|Z = 0) \leq I(K_0; F)$, $\max_t I(X_t, Z) \to 0$ por (5.22) e a distribuição condicional de X_t em relação à $Z = 0$ difere negligivelmente da distribuição incondicional. □

Tese 2.

$$\frac{k}{n} \leq \frac{1}{n} \sum_{t=1}^{n} H(X_t; Y_t) + \varepsilon, \quad \varepsilon \to 0.$$

Demonstração da Tese 2. Observe que $K_0 \to X^n F \to Y^n F Z$ é uma cadeia de Markov desde que $P_{K_0|X^n F, Y^n F Z} = P_{K_0|X^n F}$. Então, pela desigualdade de Fano (Teorema 4)

$$H(K_0|X^n F, Z = 0) \leq H(K_0|Y^n F, Z = 0) \quad = \quad o(k) \tag{5.26}$$

Pela condição (5.21), tem-se:

$$\begin{aligned} I(Y^n F; K_0|Z = 1) &= o(k) \\ H(K_0|Z = 1) - H(K_0|Y^n F, Z = 1) &= o(k) \end{aligned} \tag{5.27}$$

Por (5.24) e a equação acima (5.27) tem-se:

$$\begin{aligned} H(K_0|X^n F, Z = 1) &= H(K_0|X^n F, Z = 0) - o(k) \\ H(K_0|X^n F, Z = 1) &\leq H(K_0|X^n F, Z = 0) \\ &\leq H(K_0|Y^n F, Z = 0) = o(k) \\ \Rightarrow \quad H(K_0|X^n Y^n F, Z = 1) &\leq H(K_0|X^n F, Z = 1) \\ &\leq o(k) \end{aligned} \tag{5.28}$$

Logo,

$$\begin{aligned} k &= H(K_0|Z=1) = H(K_0|Y^nF, Z=1) + o(k) && (5.29)\\ &\leq H(K_0|X^nY^nF, Z=1) + H(X^n|Y^nF, Z=1) + o(k) && (5.30)\\ &\leq H(X^n|Y^n, Z=1) + o(k) && (5.31)\\ &\leq \sum_{t=1}^{n} H(X_t|Y_t, Z=1) + o(k) \text{pela regra da cadeia} && (5.32)\\ &\leq n\sum_{t=1}^{n} H(X_t|Y_t) + o(k) && (5.33) \end{aligned}$$

Onde a igualdade (5.29) segue de (5.27), a desigualdade (5.31) segue de (5.28) e de $H(X^n|Y^nF, Z = 1) \leq H(X^n|Y^n, Z = 1)$, a desigualdade (5.32) segue da regra da cadeia e (5.33) segue da regra da cadeia e desde que condicionar reduz a entropia. □

Finalmente, de posse das Teses (1) e (2), tem-se:

$$\frac{k}{n} \leq \frac{1}{n}\sum_{t=1}^{n} I(X_t;Y_t) + \varepsilon \leq I(X_T;Y_T)$$
$$\frac{k}{n} \leq \frac{1}{n}\sum_{t=1}^{n} H(X_t|Y_t) + \varepsilon \leq H(X_T|Y_T)$$

onde T é uma variável aleatória uniformemente distribuída em $[n]$ e independente das variáveis aleatórias X_t, Y_t.

Fazendo $X_T = X$ e $Y_T = Y$, tem-se:

$$\begin{aligned} \frac{k}{n} &\leq \max[I(X;Y), H(X|Y)]\\ \frac{k}{n} &\leq \max[I(X;Y), H(X) - I(X;Y)] \end{aligned}$$

Observe que $H(X|Y)$ mede o quanto da incerteza de X diminui quando se sabe Y e que $I(X|Y)$ mede o quanto da informação é compartilhada entre X e Y. Considerando que o GEC é usado k vezes, tem-se que $H(X) = k$. Para que a entrada e saída do GEC sejam usadas em um protocolo seguro, $I(X;Y) \geq \frac{1}{2}H(X) = \frac{1}{2}k$, pois de outro modo poderia ser utilizado uma string completamente aleatória como saída do canal. Então, dado que $H(X|Y) = H(X) - I(X;Y) \Rightarrow H(X|Y) \leq \frac{1}{2}k$. Portanto, neste caso, $\max[I(X;Y), H(X|Y)] = I(X;Y)$.

Por outro lado, lembre-se que todas as vezes que a saída do GEC é mapeado para $\mathcal{Y}^*$, essa saída é independente da entrada X. Então, potencialmente, o máximo de informação que a saída possui sobre a entrada é $(1 - p^*)H(X)$. Contudo, o GEC também acrescenta erros. Logo, a informação que a saída possui sobre a entrada é no máximo $(1 - p^*)I(X;Y)$.

Finalmente, observe que $I(X;Y) \leq \max[I(X;Y)] \Rightarrow (1-p^*)I(X;Y) \leq (1-p^*)\max[I(X;Y)]$. Desse modo, segue que

$$\frac{k}{n} \leq C(W_0)(1 - p^*).$$

Observe que este resultado é consistente desde que em (AHLSWEDE; CSISZAR, 2007) tem-se que a capacidade de OT para GEC na presença de adversários passivos é igual a $C(W_0)(1 - p^*)$. □

Prova do Teorema 9. Segue dos Teoremas 10 e 11. □

5.3 PROVA DE SEGURANÇA UNIVERSALMENTE COMPOSTA

Nesta seção será mostrado que o Protocolo 5.2 além de ser seguro quando composto sequencialmente é seguro quando composto concorrentemente com cópias ilimitadas dele mesmo e/ou com outros protocolos desconhecidos. Formalmente, será provado o seguinte teorema:

Teorema 12. *O Protocolo 5.2 UC-realiza* $\hat{\mathcal{F}}_{OT}$.

Tem-se que $\hat{\mathcal{F}}_{OT}$ é a extensão multissessão da funcionalidade ideal $\mathcal{F}_{OT}$ de OT. $\mathcal{F}_{OT}$ e $\hat{\mathcal{F}}_{OT}$ serão explicados e definidos adiante.

A partir do Capítulo 4 tem-se que para demonstrar que um protocolo é seguro no *framework* UC, define-se uma funcionalidade ideal para o protocolo OT, constrói-se um adversário ideal $\mathcal{S}$ e finalmente demonstra-se que as visões do ambiente quando está interagindo com o Protocolo 5.2 e na presença de um adversário real $\mathcal{A}$ é indistinguível daquela quando ele está interagindo com o protocolo ideal, o qual tem acesso a funcionalidade ideal, na presença do adversário ideal $\mathcal{S}$.

Primeiramente, define-se as funcionalidades ideais para o canal com apagamentos generalizado e para o String OT.

5.3.1 Funcionalidades ideais

Considera-se adversários estáticos e maliciosos. Supõe-se que quando o adversário malicioso corrompe uma parte, ele passa a ter controle de todas as fitas da parte e controlando assim as ações futuras dela. Então, não será explicitado nas funcionalidades ideais o tratamento para a corrupção das partes. Nas funcionalidades ideais seguintes, considere A o emissor, B o receptor e $\mathcal{S}$ o adversário ideal.

Funcionalidade Ideal para $\mathcal{F}_{GEC}$

- Ao receber uma mensagem (AenviaEntrada, x^n, *sid*) de A:
  - Desde que $x^n \in \{0,1\}^n$, construa a saída y^n da seguinte forma: para cada $x \in x^n$ com probabilidade $\frac{p^*-1}{p^*}$, faça $y \in \mathcal{Y}_0$ de acordo com $W_0(y|x)$ e com probabilidade $\frac{1}{p^*}$ faça $y \in \mathcal{Y}^*$ de acordo com $W(y|x)$. Envie a mensagem (BrecebeSaída, y^n, *sid*) para B, (BrecebeSaída, *sid*) para $\mathcal{S}$ e pare.

Funcionalidade Ideal para $\mathcal{F}_{OT}$

- Ao receber uma mensagem (AenviaEntrada, r_0, r_1, *sid*) de A:
  - Se $r_0 = r_1$ e $r_i \notin \{0,1\}^{\delta n}$, então envie uma mensagem ProtocoloAbortou para A e B e pare;
  - De outra forma, guarde (r_0, r_1) e *sid* e envie a mensagem(EntradaArecebida, *sid*) para $\mathcal{S}$ e B;
- Ao receber uma entrada (BenviaEntrada, c, *sid*) de B:
  - Se $c \notin \{0,1\}$, então envie uma mensagem ProtocoloAbortou para A e B e pare;
  - Se uma mensagem AenviaEntrada com o mesmo *sid* foi recebida anteriormente de A, envie uma mensagem (BrecebeSaída, r_c, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$ e pare;
  - De outra forma, ignore a mensagem e continue executando.

A seguinte funcionalidade ideal para $\mathcal{F}_{OT}$ é baseada em (CANETTI et al., 2002).

Observe que depois que o emissor envia x^n bits através do GEC, vários protocolos de *String* OT podem ser executados usando como base o mesmo par x^n, y^n. Portanto, tem-se que considerar que a invocação entre $\mathcal{F}_{GEC}$ e $\mathcal{F}_{OT}$ possuem estados compartilhados. Assim, como discutido no Capítulo 4, pode-se usar o mesmo par x^n, y^n para implementar várias instâncias do OT projetando um protocolo para a extensão multissessão $\hat{\mathcal{F}}_{OT}$ da funcionalidade ideal $\mathcal{F}_OT$. Resumidamente, $\hat{\mathcal{F}}_{OT}$ coordena as interações de $\mathcal{F}_OT$ com as partes através de subsessões de uma mesma sessão, onde a sessão é especificado por seu identificador de sessão SID e as subsessões são especificados por seus identificadores de subsessões SSID (derivado de SID). Na prática, a descrição de $\hat{\mathcal{F}}_{OT}$ é a mesma de $\mathcal{F}_{OT}$ acrescentando-se o termo *ssid* nas trocas de mensagens.

Lembre-se que o Teorema UC com estado conjunto (Teorema 6) garante que qualquer protocolo híbrido π que tem acesso à funcionalidade ideal $\mathcal{F}_{OT}$, quando composto com um protocolo ρ que UC-realiza $\hat{\mathcal{F}}_{OT}$, UC-realiza $\mathcal{F}_{OT}$. Então, basta provar que o protocolo ρ que UC-realiza $\hat{\mathcal{F}}_{OT}$. De fato, o protocolo real 5.2 será considerado como ρ.

5.3.2 Construindo o Simulador

Para simplificar a construção do simulador, nomear-se-á as mensagens trocadas entre Alice e Bob:

- No passo 4, $\mu_1 := (\mathcal{R}_0, \mathcal{R}_1, ssid, sid)$;
- No passo 6, considere $\mu_2 := (w, ssid, sid)$ a mensagem que Bob envia no início do protocolo de IH, $\mu_3 := (w_0, w_1, ssid, sid)$ a mensagem que Alice recebe ao fim do protocolo de IH e $\mu_4 := (w_0, w_1, b, ssid, sid)$ a mensagem que Bob recebe ao fim do protocolo de IH;
- No passo 7, $\mu_5 := (a, y^{\mathcal{R}_0^{\mathcal{S}_{\overline{a}}}}, y^{\mathcal{R}_1^{\mathcal{S}_a}}, ssid, sid)$;
- No passo 9, $\mu_6 := (g_0(x^{\mathcal{R}_0}), g_1(x^{\mathcal{R}_1}), g_0, g_1, h_0, h_1, ssid, sid)$.

De acordo com o Capítulo 4, será construído um simulador $\mathcal{S}$ com acesso caixa-preta à $\mathcal{A}$. O objetivo é construir um adversário $\mathcal{S}$ de forma que o ambiente $\mathcal{Z}$ não pode dizer se está interagindo com $\mathcal{A}$ na execução do Protocolo 5.2 ou se está interagindo com $\mathcal{S}$

na execução do protocolo ideal a menos de uma probabilidade negligível. Lembre-se que o protocolo ideal é aquele em que as partes *dummy* enviam suas entradas para uma funcionalidade ideal que produz e envia as saídas correspondentes à cada parte *dummy*. Observe que a partir das entradas das partes *dummy*, o protocolo ideal produz as mesmas saídas do protocolo real(Protocolo 5.2)

Lembre-se também que $\mathcal{S}$, chamado de simulador *shell*, implementa dentro dele, além de uma cópia A' do adversário $\mathcal{A}$, um simulador caixa-preta $\hat{\mathcal{S}}$. De fato, será descrito a seguir o simulador caixa-preta $\hat{\mathcal{S}}$. As interações de $\mathcal{S}$ com $\hat{\mathcal{S}}$ foram descritas na Seção 4.4 do Capítulo 4.

Supõe-se que a probabilidade p^* e a distribuição do canal W_0 é fixado e conhecido por todas as partes. Similarmente, considera-se que o parâmetro de segurança $\alpha < \frac{1-p^*}{7}$ e, consequentemente, $\beta = 1 - p^* - 2\alpha$ é fixado e conhecido por todas as partes antes da simulação começar.

Como dito no Capítulo 2, será usado apóstrofo ($'$) em todas as variáveis no ambiente simulado. Lembre-se que toda entrada que $\mathcal{S}$ recebe de $\mathcal{Z}$ é escrito na fita de entrada do adversário A' sem alteração e todo valor de saída de A' é copiado por $\mathcal{S}$, possivelmente modificado, e enviado para a fita de saída de $\mathcal{S}$ para ser lido pelo ambiente.

Será descrito a seguir as ações do simulador caixa-preta $\hat{\mathcal{S}}$ para os casos em que apenas Bob é honesto, apenas Alice é honesta, ambos Alice e Bob são corruptos e ambos Alice e Bob são honestos.

5.3.2.1 Alice Corrupta e Bob Honesto

As interações de A' com $\hat{\mathcal{S}}$ são aquelas de Alice no protocolo real com $\mathcal{Z}$ e Bob. Segue as ações de $\hat{\mathcal{S}}$ em cada interação com A'.

1. **Simulando GEC e mensagem μ_1':**
    - Assim que A' escreve que quer transmitir x'^n pelo GEC, $\hat{\mathcal{S}}$ envia a mensagem (AenviaEntrada, x'^n, *sid*) para $\mathcal{F}_{GEC}$;
    - Ao receber a mensagem (BrecebeSaída, *sid*) de $\mathcal{F}_{GEC}$, $\hat{\mathcal{S}}$ calcula para cada $x' \in x'^n$ a saída y' da seguinte forma: com probabilidade $\frac{p^*-1}{p^*}$, $y' \in \mathcal{Y}_0$

de acordo com $W_0(y|x)$ e com probabilidade $\frac{1}{p^*}$, $y' \in \mathcal{Y}^*$ de acordo com $W(y|x)$. Assim, $\hat{\mathcal{S}}$ simula a saída y'^n do GEC;

- Em seguida, $\hat{\mathcal{S}}$ cria os conjuntos $\mathcal{G}'$ e $\mathcal{B}'$, de forma que $\mathcal{G}'$ possui as posições de y'^n nas quais $y' \in \mathcal{Y}_0$ e $\mathcal{B}'$ as posições nas quais $y' \in \mathcal{Y}^*$;
- Se $|\mathcal{G}'| < (1 - p^* - \alpha)n = \beta n + \alpha n$, $\hat{\mathcal{S}}$ faz $r'_0 = r'_1$ tal que $r'_0 \notin \{0,1\}^{\delta n}$ e envia a mensagem (AENVIAENTRADA, r'_0, r'_1, $ssid, sid$) para $\hat{\mathcal{F}}_{OT}$;
- De outra forma, $\hat{\mathcal{S}}$ escolhe aleatoriamente $c' \in \{0,1\}$, $w' \in \{0,1\}^m$ e cria os conjuntos $\mathcal{R}'_0$ e $\mathcal{R}'_1$ conforme prescrito no protocolo real. Isto é, tanto $\mathcal{R}'_0$ quanto $\mathcal{R}'_1$ possuem comprimento βn; da *string* w', $\hat{\mathcal{S}}$ usa a mesma codificação de subconjuntos usada no protocolo real para obter um subconjunto de cardinalidade αn; $\hat{\mathcal{S}}$ constrói $\mathcal{R}'_{c'}$ apenas com posições de $\mathcal{G}'$, escolhidos aleatoriamente e sem repetição, enquanto $\mathcal{R}'_{\overline{c}'}$ é construído de αn posições de $\mathcal{G}'$, sem repetir as posições usadas em $\mathcal{R}''_c$, e as posições restantes são escolhidas de $\mathcal{G}' \cup \mathcal{B}'$, aleatoriamente e sem repetição;
- Em seguida, $\hat{\mathcal{S}}$ envia a mensagem $\mu'_1 := (\mathcal{R}'_0, \mathcal{R}'_1, ssid, sid)$ para A';

2. **Simulando mensagem μ'_3:**

   - Para simular que um Bob honesto envia w através de um protocolo de hasing interativo, $\hat{\mathcal{S}}$ escolhe aleatoriamente $w" \in \{0,1\}^m$ e $b' \in \{0,1\}$. $\hat{\mathcal{S}}$ faz $w'_{b'} = w'$, $w'_{1-b'} = w"$ e envia a mensagem $\mu'_3 := (w'_0, w'_1, ssid, sid)$ para A';

3. **Simulando mensagem μ'_5:**

   - $\hat{\mathcal{S}}$ calcula $a' = b' \oplus c'$, $\mathcal{S}'_{a'}$ e $\mathcal{S}'_{\overline{a}'}$ a partir de w'_0, w'_1;
   - Desde que $\hat{\mathcal{S}}$ conhece y'^n, $\mathcal{R}'_0$, $\mathcal{R}'_1$, $\mathcal{S}'_{a'}$ e $\mathcal{S}'_{\overline{a}'}$, ele pode calcular $y'^{\mathcal{R}'^{\mathcal{S}'_{\overline{a}'}}_0}$ e $y'^{\mathcal{R}'^{\mathcal{S}'_{a'}}_1}$ e enviar a mensagem $\mu'_5 := (a', y'^{\mathcal{R}'^{\mathcal{S}'_{\overline{a}'}}_0}, y'^{\mathcal{R}'^{\mathcal{S}'_{a'}}_1}, ssid, sid)$ para A';

4. **Extraindo r'_0, r'_1 e finalizando o protocolo:**

   - Ao receber a mensagem $\mu'_6 := (g'_0(x'^{\mathcal{R}'_0}), g'_1(x'^{\mathcal{R}'_1}), g'_0, g'_1, h'_0, h'_1, ssid, sid)$ de A', $\hat{\mathcal{S}}$ calcula $y'^{R'_{c'}}$, $x'^{\mathcal{R}'_0}$, $x'^{\mathcal{R}'_1}$ e todas as possíveis sequências $\widetilde{x'}^{\mathcal{R}'_{c'}}$ que são conjuntamente típicas com $y'^{R'_{c'}}$. Dessas sequências, $\hat{\mathcal{S}}$ procura pela sequência que satisfaz $g'_{c'}(\widetilde{x'}^{\mathcal{R}'_{c'}}) = g'_{c'}(x'^{\mathcal{R}'_{c'}})$;
   - Se existir apenas uma sequência $\widetilde{x'}^{\mathcal{R}'_{c'}}$ que satisfaz essas condições, $\hat{\mathcal{S}}$ calcula $r'_0 = h'_0(x'^{\mathcal{R}'_0})$, $r'_1 = h'_1(x'^{\mathcal{R}'_1})$ e envia a mensagem (AENVIAENTRADA, r'_0, r'_1, $ssid, sid$) para $\hat{\mathcal{F}}_{OT}$;
   - Senão, $\hat{\mathcal{S}}$ faz $r'_0 = r'_1 = 0^{\delta n}$ e envia a mensagem (AENVIAENTRADA, r'_0, r'_1, $ssid, sid$) para $\hat{\mathcal{F}}_{OT}$;

5. **Ao receber mensagem** ProtocoloAbortou:

   - $\hat{\mathcal{S}}$ envia a mensagem para A' que a entrega para o ambiente.

5.3.2.2 Alice Honesta e Bob Corrupto

Similarmente ao caso anterior, as interações de A' com $\hat{\mathcal{S}}$ são aquelas de Bob no protocolo real com $\mathcal{Z}$ e Alice. Segue as ações de $\hat{\mathcal{S}}$ em cada interação com A'.

1. **Simulando GEC:**

   - Ao receber a mensagem (BrecebeSaída, y'^n, sid) de $\mathcal{F}_{GEC}$, $\hat{\mathcal{S}}$ guarda o valor de y'^n e sid;
   - $\hat{\mathcal{S}}$ cria os conjuntos $\mathcal{G}'$ e $\mathcal{B}'$, de forma que $\mathcal{G}'$ possui as posições de y'^n nas quais $y' \in \mathcal{Y}_0$ e $\mathcal{B}'$ as posições nas quais $y' \in \mathcal{Y}^*$;
   - Se $|\mathcal{G}'| < (1 - p^* - \alpha)n = \beta n + \alpha n$, $\hat{\mathcal{S}}$ faz $c' \notin \{0, 1\}$ e envia a mensagem (BenviaEntrada, c', $ssid, sid$) para $\hat{\mathcal{F}}_{OT}$;

2. **Envio da mensagem μ'_1:**

   - Ao receber a mensagem $\mu'_1 := (\mathcal{R}'_0, \mathcal{R}'_1, sid, ssid)$, $\hat{\mathcal{S}}$ guarda o valor de $\mathcal{R}'_0$ e $\mathcal{R}'_1$;

3. **Simulando mensagem μ'_4:**

   - Quando o adversário A' quer enviar w' pelo protocolo de *hashing* interativo, então, supomos que A' envia uma mensagem como $\mu'_2 := (w', sid, ssid)$. Ao receber essa mensagem , $\hat{\mathcal{S}}$ escolhe aleatoriamente $w" \in \{0,1\}^m$ e $b' \in \{0,1\}$; $\hat{\mathcal{S}}$ faz $w'_{b'} = w'$, $w'_{1-b'} = w"$ e envia uma mensagem $\mu'_4 := (w'_0, w'_1, b', sid, ssid)$ como a saída do protocolo de IH para A';

4. **Extraindo c:**

   - Ao receber a mensagem $\mu'_5 := (a', y'^{\mathcal{R}'^{\mathcal{S}'}_{0\,\bar{a}'}}, y'^{\mathcal{R}'^{\mathcal{S}'}_{1\,a'}}, ssid, sid)$, $\hat{\mathcal{S}}$ calcula $c' = a' \oplus b'$;
   - $\hat{\mathcal{S}}$ verifica se $y'^{\mathcal{R}'^{\mathcal{S}'}_{0\,\bar{a}'}}$ e $y'^{\mathcal{R}'^{\mathcal{S}'}_{1\,a'}}$) são 2ϵ-conjuntamente típicas. Se não forem, $\hat{\mathcal{S}}$ faz $c' \notin \{0, 1\}$ e envia a mensagem (BenviaEntrada, c', $ssid, sid$) para $\hat{\mathcal{F}}_{OT}$. Se forem, $\hat{\mathcal{S}}$ espera o recebimento da mensagem (EntradaArecebida,

$ssid, sid$) de $\hat{\mathcal{F}}_{OT}$ e envia em seguida a mensagem (BENVIAENTRADA, c', $ssid, sid$) para $\hat{\mathcal{F}}_{OT}$;

5. **Simulando mensagem μ'_6 e terminando o protocolo:**

   - Ao receber a mensagem (BRECEBESAÍDA, $r'_{c'}$, $ssid, sid$) de $\hat{\mathcal{F}}_{OT}$, $\hat{\mathcal{S}}$ guarda $r'_{c'}$;
   - $\hat{\mathcal{S}}$ escolhe aleatoriamente funções de *hash* 2-universal $g'_0, g'_1 : \mathcal{X}^{\beta n} \rightarrow \{0,1\}^{\beta n[H(X|Y\in\mathcal{Y}_0)+\epsilon]}$, $h'_0, h'_1 : \mathcal{X}^{\beta n} \rightarrow \{0,1\}^{\delta n}$, onde $\delta = (\beta - 5\alpha)H(X) - \beta(H(X|Y \in \mathcal{Y}_0) + \epsilon) - \gamma$ e $\gamma > 0$, $g'_0(x^{\mathcal{R}_0})$, $g'_1(x^{\mathcal{R}_1}) \in \{0,1\}^{\beta n[H(X|Y\in\mathcal{Y}_0)+\epsilon]}$ e envia $\mu'_6 := (g'_0(x^{\mathcal{R}_0}), g'_1(x^{\mathcal{R}_1}), g'_0, g'_1, h'_0, h'_1, ssid, sid)$ para A';
   - $\hat{\mathcal{S}}$ intercepta o valor calculado por A' para o r_c;
   - $\hat{\mathcal{S}}$ calcula as sequências típicas de y'^n e verifica se existe apenas uma sequência $\widetilde{x'}^{\mathcal{R}'_{c'}}$ tal que $g'_{c'}(\widetilde{x'}^{\mathcal{R}'_{c'}}) = g'_{c'}(x^{\mathcal{R}_c})$;
   - Se não existe tal sequência e o valor de r_c calculado por A' é igual à $0^{\delta n}$, então $\hat{\mathcal{S}}$ envia o $r'_{c'}$ recebido por $\hat{\mathcal{F}}_{OT}$ como a saída da Bob corrupto para A'. Similarmente, se existe tal sequência e o valor de r_c calculado por A' é igual à $h'_{c'}(\widetilde{x'}^{\mathcal{R}'_{c'}}$, então $\hat{\mathcal{S}}$ envia o $r'_{c'}$ recebido por $\hat{\mathcal{F}}_{OT}$ como a saída da Bob corrupto para A';
   - Se as condições do item anterior não são satisfeitas, A' está considerando como saída r_c um valor diferente do prescrito pelo protocolo, assim $\hat{\mathcal{S}}$ envia o mesmo r_c recebido por A' como a saída da Bob corrupto de volta para A';

6. **Ao receber mensagem** PROTOCOLOABORTOU:

   - $\hat{\mathcal{S}}$ envia a mensagem para A' que a entrega para o ambiente.

5.3.2.3 Alice e Bob Corruptos

Neste caso, $\mathcal{A}$ irá gerar todas as trocas de mensagens entre Alice e Bob na execução do protocolo real e $\hat{\mathcal{S}}$ apenas entrega todas as mensagens entre as partes *dummy* e as funcionalidades ideais $\mathcal{F}_{GEC}$ e $\hat{\mathcal{F}}_{OT}$.

5.3.2.4 Alice e Bob Honestos

Quando nenhuma parte está corrompida, $\hat{\mathcal{S}}$ recebe a mensagem (BrecebeSaída, *sid*) de $\mathcal{F}_{GEC}$ e as mensagens (EntradaArecebida, *sid*) e (BrecebeSaída, *sid*) de $\hat{\mathcal{F}}_{OT}$, além da transcrição de todas as trocas de mensagens entre Alice e Bob.

5.3.3 Provando a Indistinguibilidade

Será apresentado agora o passo principal para provar que um protocolo é seguro no *framework* UC. Isto é, será provado que para todo adversário real $\mathcal{A}$ existe um adversário ideal $\mathcal{S}$ tal que para todo ambiente $\mathcal{Z}$ é computacionalmente indistinguível a saída de $\mathcal{Z}$ quando ele interage com $\mathcal{A}$ e as partes que executam o protocolo real (Protocolo 5.2) e a saída de $\mathcal{Z}$ quando ele interage com $\mathcal{S}$ e as partes *dummy* que executam o protocolo ideal com acesso à $\hat{\mathcal{F}}_{OT}$ e $\mathcal{F}_{GEC}$. Para provar o Teorema 12, será utilizado as três teses a seguir, as quais analisam a indistinguibilidade de cada caso considerado nas seções (5.3.2.1), (5.3.2.2), (5.3.2.3) e (5.3.2.4). Considere $\rho = Protocolo$ 5.2 nas três teses a seguir.

Tese 3. *Quando um adversário PPT $\mathcal{A}$ corrompe a emissora, Alice:*

$$Para\ todo\ \mathcal{Z}, \mathrm{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S}^{\mathcal{A}},\hat{\mathcal{S}},\mathcal{Z}}.$$

Demonstração da Tese 3. A ideia desta prova é demonstrar que as mensagens trocadas durante a execução do protocolo real possuem a mesma distribuição das mensagens simuladas e trocadas durante a execução do protocolo ideal. O ponto chave da demonstração está no fato de que se Bob é honesto, ele sempre executará o mesmo que está prescrito no protocolo. Logo, o simulador pode simular as mensagens de Bob apenas agindo de acordo com o protocolo.

Lembre-se que o ambiente $\mathcal{Z}$, tanto no modelo real quanto no modelo ideal, envia entradas para o adversário e para cada uma das partes e lê a saída de cada um deles. Como o adversário é estático, supõe-se que Alice está sobre o controle do adversário $\mathcal{A}$ desde o começo da execução do protocolo.

Observe que toda máquina de Turing polinomial probabilística (PPT) pode ser vista como uma máquina de Turing polinomial determinística, onde as escolhas internas da função de transição da máquina são dadas como uma entrada auxiliar. Isto é, como nas definições de ITM na Seção 2.2 do Capítulo 2, uma máquina PPT possui

duas entradas, onde uma delas é a entrada usual (na fita de entrada) e a outra é uma sequência aleatória representando as escolhas feitas pela máquina (na fita de aleatoriedade). Dessa forma o comportamento do adversário no modelo real $\mathcal{A}$, e portanto da sua cópia A', é completamente determinado pelos bits aleatórios de $\mathcal{A}$, $r_{\mathcal{A}}$, pela entrada x'^n do GEC e pelas mensagens μ'_1, μ'_3 e μ'_5 (pois x'^n, μ'_1, μ'_3 e μ'_5 são os dados que $\mathcal{A}$ recebe do simulador $\mathcal{S}$).

Independentemente da estratégia do adversário $\mathcal{A}$, ele pode desviar do protocolo quando envia a *string* x'^n pelo GEC e nos dados enviados na mensagem μ'_6. Contudo, nestes dois casos a simulação captura igualmente o comportamento das partes e, portanto, de $\mathcal{Z}$ na execução do protocolo real.

Se x'^n é diferente daquela prescrita no protocolo real ($x'^n \notin \{0,1\}^n$ ou x'^n com distribuição que não atinge a capacidade de Shannon para W_0), a simulação copia este comportamento perfeitamente, até porque esta mesma entrada será transmitida para a funcionalidade $\mathcal{F}_{GEC}$, a qual se comporta igualmente ao canal GEC. Logo, a simulador pode abortar o protocolo com a mesma probabilidade do Bob honesto abortar na execução do protocolo real. Além disso, tanto $\mathcal{A}$ quanto $\mathcal{S}$ não vão abortar os respectivos protocolos nos passos 5 e 8 porque, sendo Bob honesto, ele segue o protocolo.

Se os dados enviados na mensagem μ'_6 diferem daqueles prescrito pelo protocolo real, Bob pode ter como saída $r_c = 0^{\delta n}$ com a mesma probabilidade que a versão *dummy* de Bob tem essa saída, pois o simulador $\mathcal{S}$ simula a reação de Bob antes de enviar as entradas r'_0, r'_1 para a funcionalidade ideal. Como Bob é honesto, ele sempre irá executar o mesmo procedimento prescrito pelo protocolo e como o simulador realiza este mesmo procedimento ao enviar as entradas para $\hat{\mathcal{S}}$, pode-se afirmar que a saída de Bob e da versão *dummy* de Bob é a mesma.

Agora, lembre-se que é o ambiente $\mathcal{Z}$ que envia a aleatoriedade $r_{\mathcal{A}}$ do adversário $\mathcal{A}$. Então, essa aleatoriedade possui a mesma distribuição tanto na execução do protocolo real quanto na execução do protocolo ideal. Por outro lado, observe que o simulador simula as mensagens μ'_1, μ'_3 e μ'_5 com dados que possuem a mesma distribuição daqueles prescritos no protocolo real. Logo, pode-se concluir que as saídas do adversário $\mathcal{A}$, de Alice, de Bob, das versões *dummy* de Alice e Bob e do simulador $\mathcal{S}$ possuem a mesma distribuição. Logo é perfeitamente indistinguível a saída de um ambiente quando interage com $\mathcal{A}$ e as partes executando o protocolo real da saída deste mesmo ambiente quando interage com $\mathcal{S}$ e as partes *dummy* executando o protocolo ideal.

Portanto, como $\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}$, segue a tese. □

Tese 4. *Quando um adversário PPT $\mathcal{A}$ corrompe o receptor, Bob:*

$$\textit{Para todo } \mathcal{Z}, \text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \text{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

Demonstração da Tese 4. A demonstração desta tese é extremamente similar à da tese anterior. Será focado aqui as diferenças.

O comportamento do adversário $\mathcal{A}$ é determinado, neste caso, por $r_{\mathcal{A}}$, pela saída y'^n do $\mathcal{F}_{GEC}$ e pelas mensagens simuladas μ'_4 e μ'_6. Além disso, $\mathcal{A}$ pode desviar do protocolo nos dados enviados em μ'_1, μ'_2 e μ'_5. Contudo, como Alice é honesta, ela segue o protocolo e com isso o simulador age da mesma forma que Alice ao receber os dados do Bob malicioso.

Outra diferença é que neste caso, o simulador não conhece o valor das entradas r_0, r_1 da Alice nem sabe x^n. Entretanto, tem-se que o simulador testa se a saída do adversário é consistente com os valores fornecidos por $\mathcal{F}_{GEC}$ e os valores das mensagens μ'_4, mu'_6 simuladas por $\mathcal{S}$, onde os valores de μ'_4 e μ'_6 possuem a mesma distribuição daqueles prescritos pelo protocolo real. Portanto $\mathcal{S}$ sabe se o adversário tentou seguir o protocolo ou não. Se o adversário real seguiu, o simulador troca o valor calculado por $\mathcal{A}$ pelo valor recebido por $\hat{\mathcal{F}}_{OT}$. Senão, o simulador tem como saída a mesma saída arbitrária de $\mathcal{A}$. Esse comportamento garante que as saídas de $\mathcal{A}$, de $\mathcal{S}$, de Alice e Bob e suas versões *dummy* possuam distribuições idênticas. Logo, a visão do simulador no modelo real e no modelo ideal é o mesmo, ou seja, $\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}$ e segue a tese. □

Tese 5. *Quando um adversário PPT $\mathcal{A}$ corrompe tanto o emissor quanto o receptor ou quando não corrompe nem o emissor, nem o receptor, tem-se:*

$$\textit{Para todo } \mathcal{Z}, \text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \text{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

Demonstração da Tese 5. Analisando o caso em que tanto o emissor (Alice) quanto o receptor (Bob) são corrompidas, tem-se que o adversário $\mathcal{A}$ (e portanto $\mathcal{S}$) tem total controle das mensagens enviadas entre as partes. Logo, as saídas de $\mathcal{A}$, de $\mathcal{S}$, de Alice e Bob e suas versões *dummy* possuam distribuições idênticas.

No caso em que nenhuma das partes é corrompida, tanto Alice quanto Bob, quanto as versões *dummy* deles se comportarão da mesma forma, produzindo saídas com a

mesma distribuição, pois as entradas foram enviadas igualmente pelo ambiente e como em ambos os casos eles seguem o que foi prescrito pelo protocolo real, eles terão como saída a mesma coisa.

Dessas duas observações acima, segue o resultado. □

Prova do Teorema 12. Segue diretamente das Teses 3, 4 e 5, do Teorema 6 e da Tese 11 de (CANETTI, 2000). □

O Capítulo a seguir apresentará o outro grupo de contribuições que são protocolos para álgebra linear seguros no *framework* UC. A solução para *oblivious transfer* apresentada neste capítulo utilizou um canal ruidoso como hipótese inicial para obter um protocolo de duas partes segura. Nas soluções apresentadas no capítulo a seguir será utilizado como hipótese inicial a presença de um inicializador confiável para obter protocolos de duas partes segura.

6 PROTOCOLOS PARA ÁLGEBRA LINEAR DISTRIBUÍDA SEGURA

Neste capítulo será apresentado protocolos para se calcular determinante, autovalor e autovetor de forma distribuída e segura. Esses protocolos possuem rodadas constantes, são incondicionalmente seguros, isto é sua segurança não depende do poder computacional do adversário, e são seguros quando universalmente compostos. Juntamente com o protocolo para produto interno, eles fornecem as principais ferramentas de álgebra linear.

O principal uso imediato de protocolos de álgebra linear distribuídos é na mineração de dados. Neste problema, cada parte possui o seu próprio banco de dados e deseja-se fazer uma análise estatística entre eles ou, de forma geral, deseja-se computar uma função dos bancos de dados. Contudo, sendo as partes "desconfiadas", deseja-se que a partir da saída da função e das mensagens trocadas não se possa inferir as entradas das partes.

Lembre-se que uma computação de duas partes segura só é possível com uma suposição adicional (veja mais no Capítulo 3). O Protocolo 5.2 para a computação de *String* OT utilizou como hipótese adicional a presença de um canal ruidoso (GEC) entre as duas partes. Os protocolos deste capítulo, entretanto, utilizarão como hipótese adicional a presença de um inicializador confiável (TI). Até onde se sabe, esses são os primeiros protocolos para se calcular determinante, autovalores e autovetores baseados na presença de um TI. Como foi citado na seção 2.9 do Capítulo 2, a principal vantagem dos protocolos com a presença de um inicializador confiável é que eles são mais simples e eficientes. Por exemplo, para fornecer privacidade para as partes não é necessário o uso de provas de conhecimento nulo quando se possui um TI.

Os protocolos apresentados neste capítulo, utilizam a propriedade de se calcular produto interno entre vetores. Mais precisamente, precisa-se calcular produto de matrizes, que nada mais são do que produto interno entre linhas e colunas. Existem diversas propostas de implementações de protocolos distribuídos para o produto interno. Contudo, grande parte desses protocolos se mostraram inseguros. Para construir os protocolos para determinante, autovalor e autovetor, será usado como subprotocolo a solução de

(DOWSLEY et al., 2010) levemente modificada para produto interno. A modificação consiste de ao invés de Alice escolher aleatoriamente um valor r em $\mathbb{F}_q$, esse valor será pré-distribuído pelo TI. Essa solução é preferível por ser segura quando universalmente composta. Então, pelo Teorema da Composição (Teorema 5) pode-se afirmar que os protocolos apresentados são também seguros quando universalmente compostos.

Todos os cálculos serão realizados em um corpo finito $\mathbb{F}_q$ de ordem q. Lembre-se que as matrizes inversíveis são aquelas que possuem determinante não nulos e também são chamadas de não singulares. O conjunto de toda as matrizes $n \times n$ não-singulares com elementos em $\mathbb{F}_q$ será denotado por $SL(\mathbb{F}_q)$.

Em especial, considera-se que cada parte que executa os protocolos a seguir possuem a habilidade de calcular o determinante isoladamente. Observe que isto não interfere no objetivo dos protocolos, uma vez que se pretende calcular o determinante, autovalor e autovetor da entrada de ambas as partes. Dito de outra maneira, no caso do determinante, por exemplo, deseja-se calcular o determinante de $X + Y$, sendo X a entrada privada de uma parte e Y a entrada privada de outra parte.

Considera-se que as duas partes que executam os protocolos sejam Alice e Bob. Além disso, ao final da execução dos protocolos Alice não recebe nada e Bob recebe o resultado da computação. A exceção desse comportamento são os protocolos de produto interno e de multiplicação de matrizes. Ao final dos protocolos Alice recebe um valor aleatório e Bob recebe outro de forma que a soma das saídas de Alice e Bob é igual ao produto interno de suas entradas ou o produto das matrizes de entrada. Ter o protocolo de produto interno definido desta maneira será importante para garantir que a entrada da Alice permanece oculta para Bob.

Outra consideração importante a ser feita é que se supõe a presença de apenas um inicializador confiável para o protocolo principal e para os subprotocolos. Além disso, todos eles são inicializados simultaneamente.

Em todas as provas no *framework* UC considerado neste capítulo, considera-se adversários estáticos e maliciosos. Supõe-se que quando o adversário malicioso corrompe uma parte, ele passa a ter controle de todas as fitas da parte, controlando assim as ações futuras dela. Então, não será explicitado nas funcionalidades ideais apresentadas o tratamento para a corrupção das partes. Vale ressaltar que sendo o adversário ideal estático, ele corrompe uma parte antes da execução do protocolo. Portanto, o

adversário ideal conhece a entrada da parte corrompida.

6.1 PRODUTO INTERNO

Nesta seção considera-se a definição usual de produto interno: dado dois vetores $\vec{a} = (a_1, a_2, ..., a_n)$, $\vec{b} = (b_1, b_2, ..., b_n)$, onde a_i e b_i são elementos de $\mathbb{F}_q$ e $i \in \{1, ..., n\}$, o produto interno entre $\vec{a}$ e $\vec{b}$ é dado por:

$$\langle \overrightarrow{a} \cdot \overrightarrow{b} \rangle = \sum_{i=1}^{n} a_i b_i.$$

O protocolo de produto interno que será apresentado a seguir é um protocolo para produto interno compartilhado. Isto é, as entradas de Alice e Bob são vetores em $\mathbb{F}_q$ e as saídas de Alice e Bob são dois valores aleatórios em $\mathbb{F}_q$ tal que a soma dos dois valores aleatórios é igual ao produto interno dos vetores de entradas. Diz-se que um protocolo para produto interno compartilhado é privado se ao final da execução do protocolo Alice e Bob não extraem nenhuma informação sobre a entrada do outro jogador além das informações contidas em suas respectivas saídas e trocas de mensagem. Esquematicamente, tem-se:

Tabela 6.1: Funcionalidade do Produto Interno

	Alice	Bob
Entradas	$\vec{x} \in \mathbb{F}_q$	$\vec{y} \in \mathbb{F}_q$
Saídas	$\vec{r} \in_R \mathbb{F}_q$	$\langle \overrightarrow{x} \cdot \overrightarrow{y} \rangle - r$

O protocolo apresentado a seguir é baseado em (DOWSLEY et al., 2010, Seção 3.4).

Protocolo 6.1 Protocolo para Produto Interno

1: O Inicializador Confiável escolhe $\overrightarrow{x_0} \in_R \mathbb{F}_q^n$, $\overrightarrow{y_0} \in_R \mathbb{F}_q^n$, $r \in_R \mathbb{F}_q$, calcula $s_0 := \langle \overrightarrow{x_0} \cdot \overrightarrow{y_0} \rangle$ e envia $\mu_A := (\overrightarrow{x_0}, r)$ para Alice e $\mu_B := (\overrightarrow{y_0}, s_0)$ para Bob.
2: Bob calcula $\overrightarrow{y_1} := \overrightarrow{y} - \overrightarrow{y_0}$ e envia $\mu_1 := (\overrightarrow{y_1})$ para Alice.
3: Alice confere se $\mu_1 \in \mathbb{F}_q^n$ e aborta se não for o caso.
4: Alice calcula $\overrightarrow{x_1} := \overrightarrow{x} + \overrightarrow{x_0}$, faz $r_1 := \langle \overrightarrow{x} \cdot \overrightarrow{y_1} \rangle - r$ e envia $\mu_2 := (\overrightarrow{x_1}, r_1)$ para Bob.
5: Bob aborta se $\overrightarrow{x_1} \notin \mathbb{F}_q^n$ ou $r_1 \notin \mathbb{F}_q$. Caso contrário, ele tem como resultado $v := \langle \overrightarrow{x_1} \cdot \overrightarrow{y_0} \rangle + r_1 - s_0$

Correção: Deseja-se mostrar que a saída do Protocolo 6.1 é igual à especificada, $\langle \overrightarrow{x} \cdot \overrightarrow{y} \rangle - r$.

Observe que

$$\begin{aligned} v &= \langle \overrightarrow{x_1} \cdot \overrightarrow{y_0} \rangle + r_1 - s_0 \\ &= \langle (\overrightarrow{x} + \overrightarrow{x_0}) \cdot \overrightarrow{y_0} \rangle + \langle \overrightarrow{x} \cdot (\overrightarrow{y} - \overrightarrow{y_0}) \rangle - r - \langle \overrightarrow{x_0} \cdot \overrightarrow{y_0} \rangle \\ &= \langle \overrightarrow{x} \cdot \overrightarrow{y_0} \rangle + \langle \overrightarrow{x_0} \cdot \overrightarrow{y_0} \rangle + \langle \overrightarrow{x} \cdot \overrightarrow{y} \rangle - \langle \overrightarrow{x} \cdot \overrightarrow{y_0} \rangle - r - \langle \overrightarrow{x_0} \cdot \overrightarrow{y_0} \rangle \\ &= \langle \overrightarrow{x} \cdot \overrightarrow{y} \rangle - r \end{aligned}$$

Alice não consegue extrair $\vec{y}$ através de μ_1 porque $\overrightarrow{y_0}$ é escolhido uniformemente pelo TI. Similarmente, Bob não extrai $\vec{x}$ através de μ_2. Além disso, a saída de Bob se comporta de forma aleatória para ele uma vez que r é escolhido aleatoriamente pelo TI.

A prova de que esse protocolo é seguro no *framework* UC é extremamente similar à prova dos protocolos que serão apresentados a seguir e se encontra em (DOWSLEY et al., 2010).

Esse protocolo pode ser adaptado para calcular multiplicação de matrizes da seguinte forma:

Tabela 6.2: Funcionalidade de Multiplicação de Matrizes

	Alice	Bob
Entradas	$A \in \mathbb{F}_q^n$	$B \in \mathbb{F}_q^n$
Saídas	$R \in_R \mathbb{F}_q^n$	$C = AB - R \in \mathbb{F}_q^n$

Protocolo 6.2 Protocolo para Produto de Matrizes

1: Para $1 \leq i \leq n$, Alice seleciona a i-ésima linha de A ($\overrightarrow{a_i}$) e executa n vezes o Protocolo 6.1, onde a entrada de Bob no Protocolo 6.1 é a j-ésima coluna de B ($\overrightarrow{b_j}$), com j variando de 1 a n. Alice constrói a matriz R a partir das saídas r_{ij} de cada uma das n^2 execuções do Protocolo 6.1 e Bob constrói C a partir das suas saídas do Protocolo 6.1, ou seja, $c_{ij} = \langle \overrightarrow{a_i} \cdot \overrightarrow{b_j} \rangle - r_{ij}$

Do Teorema da Composição Universal com Estados Conjunto, Teorema 6, tem-se que o Protocolo 6.2 é seguro no *framework* UC dado que ele é constituído somente de n^2 chamadas do Protocolo 6.1 que é UC seguro.

A figura 6.1 deixa claro que todos os n^2 subprotocolos de produto interno necessários para calcular o produto entre duas matrizes quadradas de dimensões $n \times n$ são inicializados simultaneamente pelo mesmo inicializador confiável, mas são executados apenas quando são chamados pelo protocolo principal (protocolo de produto de matrizes).

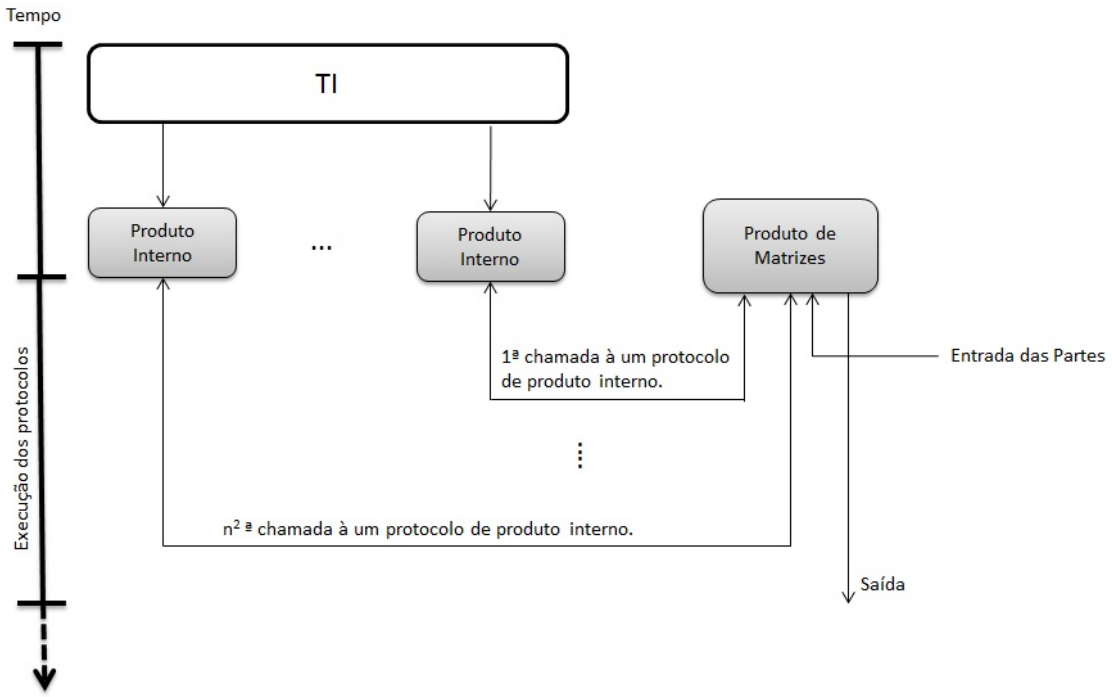

Figura 6.1: Fase de Inicialização e Fase de Execução do Protocolo de Produto de Matrizes e Seus Subprotocolos

Observe que o Protocolo 6.2 manipula matrizes quadradas por estas serem as matrizes de interesse nos próximos protocolos. Contudo, ele funciona para as matrizes tal que a quantidade de linhas da matriz de entrada de Alice seja igual a quantidade de colunas da matriz de Bob.

É interessante notar que o Protocolo 6.1 permite a solução de sistema lineares, como foi demonstrado em (DOWSLEY et al., 2010). A solução apresentada naquele artigo é demonstrado ser seguro no *framework* UC.

6.2 DETERMINANTE

Historicamente, o determinante foi definido como uma propriedade de um sistema de equações lineares, isto é, o determinante "determinava"se o sistema tinha uma única, várias ou nenhuma solução. O determinante $\det(A)$ é um número associado a uma matriz quadrada $A = [a_{ij}]$ cujas entradas a_{ij} podem ser tomadas em qualquer anel

comutativo. Existem várias formas de se definir o determinante, como através da fórmula de Leibniz, fórmula de Laplace, ou expansões de série de Taylor. Opta-se por definir determinante pela fórmula de Leibniz pelo seu tratamento usual e propriedades que a seguem. Para a definição de determinante precisa-se das noções de permutação e número de inversões.

Uma permutação é uma função que dispõe em uma certa ordem um determinado conjunto que tem a propriedade de ser totalmente ordenado. Para o uso na definição de determinante considera-se subconjuntos dos números naturais e a relação de ordem estrita "<". Então, eles são conjuntos totalmente ordenados. Por exemplo, para $n = 3$, o conjunto a ser considerado é $\{1, 2, 3\}$ e tem-se que [123] é uma permutação assim como [231]. O conjunto de todas tais permutações, S_n, é um grupo simétrico com ordem $n!$, isto é, existem $n!$ permutações em um conjunto de n elementos.

Definição 31. *Dado uma permutação $\sigma = [\sigma_1\sigma_2...\sigma_n]$ no conjunto $[n] = \{1, 2, ..., n\}$, diz-se que exite uma* inversão *quando um número natural precede outro menor que ele, isto é, quando $\sigma_i > \sigma_j, i < j$.*

Agora pode-se definir a paridade de uma permutação. Uma permutação será par ou ímpar segundo o número de inversões presentes. Para expressar a paridade usa-se a função sinal, denotado sgnσ, tendo valor $+1$ se σ for par e -1 se σ for ímpar. [14]

Definição 32. *O sinal de uma permutação σ é expresso como:*

$$sgn(\sigma) = (-1)^{N(\sigma)} \tag{6.1}$$

onde $N(\sigma)$ é o número de inversões na permutação σ.

Com as definições e noções estabelecidas acima, pode-se definir determinante pela fórmula de Leibniz:

Definição 33. *O determinante de uma matriz quadrada $A = [a_{ij}]_{n\times n}, n \in \mathbb{N}$ é um número calculado da seguinte forma:*

$$\det(A) = \sum_{\sigma \in S_n} sgn(\sigma)\Pi_{i=1}^{n} a_{i\sigma_i} \tag{6.2}$$

Segundo essa definição de determinante, pode-se afirmar as seguintes propriedades:

[14] Observe que uma permutação não pode ser simultaneamente par e ímpar.

1. Se todos os elementos de uma linha, ou de uma coluna, de uma matriz são nulos, então o determinante será nulo;
2. Se for multiplicado uma linha da matriz por uma constante, o determinante fica multiplicado por esta constante;
3. Se for trocado a posição de duas linhas, ou duas colunas, o determinante troca de sinal;
4. Se a matriz possui duas linhas, ou duas colunas, iguais, então o determinante será nulo;
5. Dada uma matriz quadrada A e sua transposta A', tem-se que $\det(A) = \det(A')$;
6. Dada uma matriz quadrada inversa I, $\det(I) = 1$;
7. Dados duas matrizes quadradas A e B, $\det(AB) = \det(A)\det(B)$;
8. Uma matriz quadrada sobre um anel comutativo F é inversível se, e apenas se, seu determinante é um elemento com inverso multiplicativo;
9. Se F for um corpo, a propriedade anterior é equivalente ao determinante ser não nulo.

O protocolo para cálculo do determinante que será apresentado a seguir utiliza especialmente a propriedade de número 7.

6.2.1 Implementação

O protocolo para calcular determinantes de forma distribuída possui a seguinte relação entre entradas e saídas: dadas as matrizes de entrada X e Y de Alice e Bob, respectivamente, ao término da execução do protocolo, Alice nada recebe e Bob recebe o determinante de $X + Y$.

Tabela 6.3: Funcionalidade de Determinante

	Alice	Bob
Entradas	$X \in \mathbb{F}_q^{n \times n}$	$Y \in \mathbb{F}_q^{n \times n}$
Saídas	$\perp$	$\det(X + Y)$

Como todo protocolo de computação segura de duas partes, deseja-se que Alice não receba nenhuma informação sobre a entrada do Bob além daquela obtida pela troca

de mensagens e que, de forma similar, Bob não receba nenhuma informação sobre a entrada da Alice além daquela obtida pela saída e troca de mensagens.

O Protocolo 6.3 funciona da seguinte maneira: O inicializador confiável (TI) distribui para Bob matrizes escolhidas aleatoriamente em $\mathbb{F}_q^{n\times n}$ e distribui para Alice uma matriz escolhida aleatoriamente e uma relação entre as matrizes distribuídas para Bob. Precisamente o TI envia as matrizes $P \in_R SL(\mathbb{F}_q), Q \in_R SL(\mathbb{F}_q)$ e $U \in_R \mathbb{F}_q^{n\times n}$ para Bob e as matrizes $R \in_R \mathbb{F}_q^{n\times n}$ e $V = RQ + U$ para Alice. É importante frisar que as matrizes P e Q devem ser não-singulares ($\det \neq 0$) porque o protocolo requer que seja usado o determinante delas como denominadores de uma divisão.

É importante frisar que o inicializador confiável do protocolo em questão e do Protocolo 6.2 de produto interno é o mesmo. Além disso, a fase de inicialização do protocolo e dos subprotocolos ocorrem simultaneamente. Veja a figura 6.2 a seguir.

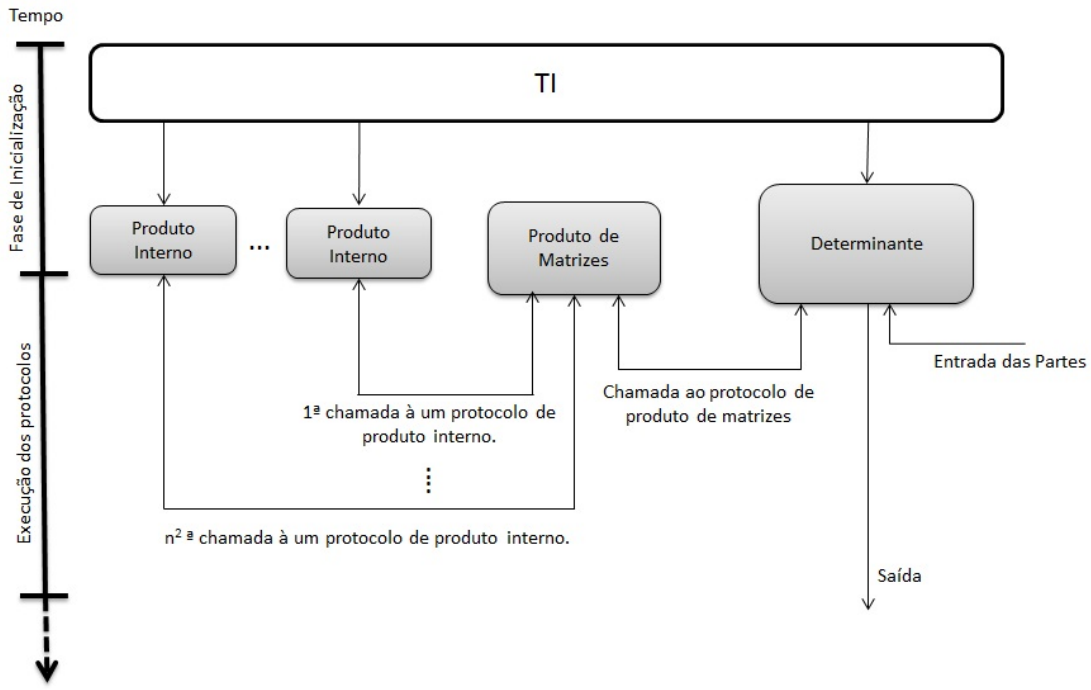

Figura 6.2: Fases de inicialização e de Execução do Protocolo e dos Subprotocolos necessários para Calcular Determinante

Portanto, além de escolher as matrizes P, Q, V, U, R, o TI escolhe aleatoriamente duas matrizes quadradas $n^2 \times n^2$ com elementos em $\mathbb{F}_q$, X_0 e Y_0, e calcula o vetor $\overrightarrow{s_0} \in \mathbb{F}_q^n$, onde cada elemento de $\overrightarrow{s_0}$ é da forma $s_{0l} = \langle \overrightarrow{x_{0l}} \cdot \overrightarrow{y_{0l}} \rangle$, com $\overrightarrow{x_{0l}}$, $\overrightarrow{y_{0l}}$ sendo as l-ésimas linhas de X_0 e Y_0 respectivamente e $1 \leq l \leq n^2$. Então, o TI envia em cada uma das n^2 execuções do Protocolo 6.2 (necessárias para o Protocolo 6.2) $\mu_A := (\overrightarrow{x_{0l}}, R)$ para Alice e $\mu_B := (\overrightarrow{y_{0l}}, s_{0l})$ para Bob.

Após receber as informações vindas de TI, Alice e Bob executam o Protocolo 6.2 com a entrada de Alice sendo X e a entrada de Bob sendo P. Assim, ao término da execução do Protocolo 6.2, Alice recebe o mesmo R que foi recebido inicialmente pelo TI e Bob recebe a matriz $C = PX - R$. Observe que não há problema algum de Alice receber a mesma matriz R duas vezes. O mesmo efeito é conseguido se o TI não entregar R no início do protocolo em questão e deixar para Alice o receber ao fim da execução do Protocolo 6.2. Optou-se por apresentar a entrega de R duas vezes para Alice para facilitar a análise da prova de segurança UC e para deixar claro que a matriz V recebida por Alice é uma função de R. De fato, deseja-se apresentar uma funcionalidade para o inicializador confiável contendo apenas os itens utilizados no desenvolvimento do protocolo em questão, ignorando os itens utilizados no subprotocolo.

Assim que Bob recebe a matriz C, ele calcula e envia para Alice a matriz $M := CQ+PYQ-U$. Se a matriz não possui a distribuição esperada, Alice aborta. De outra forma, Alice retira de M a parte que não é necessária para o cálculo do determinante , que foi acrescentada à M para garantir o "ocultamento"da entrada de Bob, somando à M a matriz V recebida pelo TI e, assim, obtendo N. Observe que ao calcular N, Alice não extrai a entrada de Bob porque ela não conhece nem P nem Q. Em seguida, Alice calcula e envia para Bob o determinante de N. Bob então, pode calcular os determinantes de P e de Q, que são garantidos pelo TI serem diferentes de zero. Ao dividir o determinante de N recebido pela Alice pelo produto dos determinantes de P e de Q, Bob consegue o determinante da soma de X e Y.

Protocolo 6.3 Protocolo para Determinante

1: O Inicializador Confiável escolhe no início do protocolo $R \in_R \mathbb{F}_q^{n\times n}, P \in_R SL(\mathbb{F}_q), V = RQ + U, Q \in_R SL(\mathbb{F}_q), U \in_R \mathbb{F}_q^{n\times n}$ e envia $\mu_A := (R, V)$ para Alice e $\mu_B := (P, Q, U)$ para Bob.

2: Alice e Bob executam o Protocolo 6.2 sendo X a entrada da Alice, onde $X \in \mathbb{F}_q^{n\times n}$, e P como entrada de Bob. Ao final da execução do Protocolo 6.2 Bob tem como resultado a matriz $C = PX - R$.

3: Bob calcula $M := CQ + PYQ - U$, onde $Y \in \mathbb{F}_q^{n\times n}$, e envia $\mu_1 := (M)$ para Alice.

4: Alice aborta se $\mu_1 \notin \mathbb{F}_q^{n\times n}$. Caso contrário, Alice faz $N := M + V$, calcula $t = \det(N)$ e envia $\mu_2 := (t)$ para Bob.

5: Bob aborta se $\mu_2 \notin \mathbb{F}_q^{n\times n}$. Caso contrário, Bob faz $var := \det(P)\det(Q)$ e calcula $z = \frac{t}{var} = \det(X + Y)$

A figura 6.3 mostra a troca de mensagem entre Alice, Bob e TI, além das entradas

das partes. A saída de Alice, R, do Protocolo 6.2 não é mostrada, pois possui o mesmo valor do R pré-distribuído pelo TI na fase de inicialização.

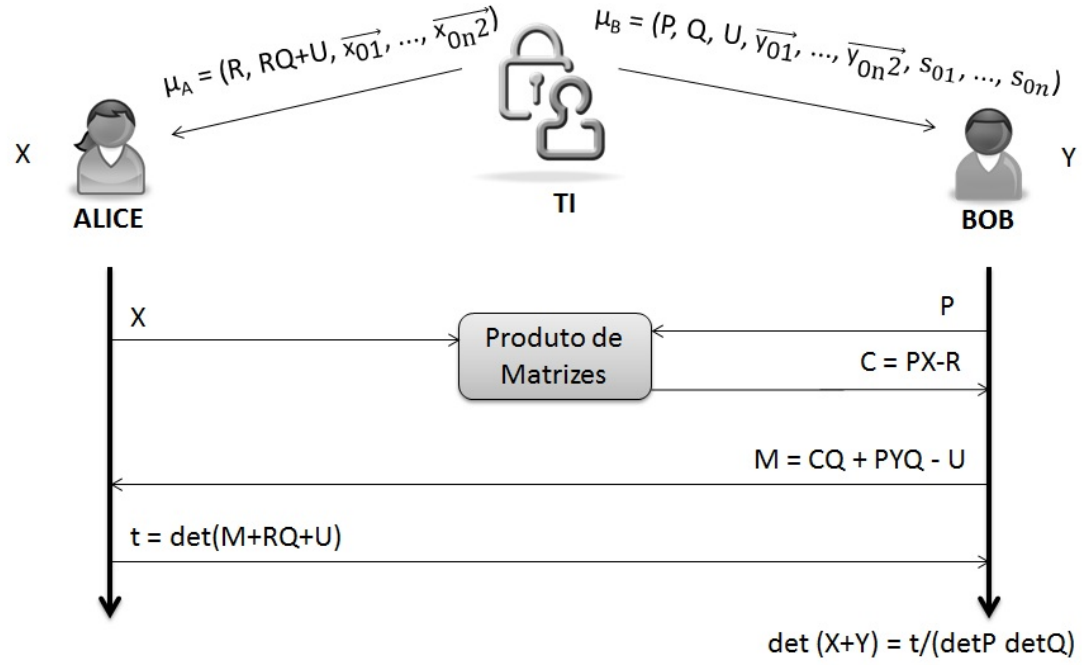

Figura 6.3: Troca de Mensagens no Protocolo para Cálculo de Determinante

Correção: Deseja-se mostrar que a saída do Protocolo 6.3 é de fato o $\det(X+Y)$. Observe que $N := M+V = CQ+PYQ-U+RQ+U = P(X+Y)Q$. Da propriedade 7, tem-se que dado duas matrizes quadradas A e B, $\det(AB) = \det(A)\det(B)$. Então, desde que $\det(P) \neq 0$ e $\det(Q) \neq 0$, tem-se que

$$z = \frac{t}{var} = \frac{\det(N)}{var} = \frac{\det(P(X+Y)Q)}{\det(P)\det(Q)} = \det(X+Y). \tag{6.3}$$

Alice não consegue extrair Y através de μ_1 porque Y está "escondido"pela multiplicação entre duas matrizes escolhidas aleatoriamente uniforme P e Q desconhecidas por ela. Além disso, os termos CQ e U são também desconhecidos por Alice. Desde que o determinante é um número em $\mathbb{F}_q$ e que, em geral, $\det(A+B) \neq \det(A) + \det(B)$, tem-se que Bob não extrai X através de sua saída e, portanto, de μ_2.

A prova de que o Protocolo 6.3 é seguro no *framework* UC será apresentado a seguir. Como foi dito no Capítulo 4 e feito na prova de UC do Capítulo 5, para provar que um protocolo é seguro no *framework* UC, segue-se três passos: primeiramente,

determina-se as funcionalidades ideais; em seguida, constrói-se um adversário ideal $\mathcal{S}$; e, finalmente, demonstra-se que as visões do ambiente quando está interagindo com o Protocolo 6.3 na presença de um adversário real $\mathcal{A}$ é indistinguível daquela quando ele está interagindo com o protocolo ideal, o qual tem acesso à funcionalidade ideal, na presença do adversário ideal $\mathcal{S}$.

6.2.2 Funcionalidades Ideais

Define-se a seguir as funcionalidades ideias para o inicializador confiável e para um protocolo que calcula o determinante conforme definido no início desta seção. Nas funcionalidades ideais seguintes, considere A a Alice, B o Bob e $\mathcal{S}$ o adversário ideal.

Funcionalidade Ideal para o Cálculo do Determinante $\mathcal{F}_{\det(X,Y)}$

- Ao receber uma mensagem (AenviaEntrada, X, *sid*) de A:
  - Ignore qualquer mensagem AenviaEntrada subsequente;
  - Se $X \notin \mathbb{F}_q^{n\times n}$, então envie uma mensagem EntradaInválida para A e B e pare;
  - Se nenhuma mensagem BenviaEntrada foi recebida anteriormente de B, guarde X e *sid* e envie uma mensagem (EntradaDeArecebida, *sid*) para B e $\mathcal{S}$. De outra forma, calcule $z = \det(X + Y)$ e envie uma mensagem (BrecebeSaída, z, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$;
- Ao receber uma mensagem (BenviaEntrada, Y, *sid*) de B:
  - Ignore qualquer mensagem BenviaEntrada subsequente;
  - Se $Y \notin \mathbb{F}_q^{n\times n}$, então envie uma mensagem EntradaInválida para A e B e pare;
  - Se nenhuma mensagem AenviaEntrada foi recebida anteriormente de A, guarde Y e *sid*, e envie uma mensagem (EntradaDeBrecebida, *sid*) para A e $\mathcal{S}$. De outra forma, calcule $z = \det(X + Y)$ e envie uma mensagem (BrecebeSaída, z, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$.

Como foi discutido anteriormente, não é explicitado nesta funcionalidade para TI as

matrizes usadas no subprotocolo que calcula o produto entre matrizes. Contudo, isso pode ser feito sem perda de generalidade, pois o subprotocolo já foi provado ser UC seguro.

Funcionalidade Ideal para o Inicializador Confiável $\mathcal{F}_{TI}$

- Quando ativado, escolha $P \in_R SL(\mathbb{F}_q)$, $Q \in_R SL(\mathbb{F}_q)$, $R \in_R \mathbb{F}_q^{n\times n}$ e $U \in_R \mathbb{F}_q^{n\times n}$ e calcule $V = RQ + U$. Envie R e V para Alice e P, Q e U para Bob.

6.2.3 Prova de Segurança Universalmente Composta

A seguir, será construído um adversário ideal $\mathcal{S}$ chamado de simulador *shell* o qual possui dentro dele uma cópia A' do adversário real $\mathcal{A}$ e um simulador caixa-preta $\hat{\mathcal{S}}$. As interações de $\mathcal{S}$ com $\hat{\mathcal{S}}$ e A' foram descritas na Seção 4.4 do Capítulo 4. Observe que, de fato, o que muda para cada protocolo é o simulador caixa-preta $\hat{\mathcal{S}}$.

A construção de $\mathcal{S}$ deve objetivar que todo ataque que o adversário real possa realizar no modelo real seja simulado pelo adversário ideal no modelo ideal. Lembre-se que a comunicação de $\mathcal{S}$ com o ambiente $\mathcal{Z}$ ocorre da seguinte forma: toda entrada que $\mathcal{S}$ recebe de $\mathcal{Z}$ é escrito na fita de entrada do adversário A', sem alteração, e todo valor de saída de A' é copiado por $\mathcal{S}$, possivelmente modificado, e enviado para a fita de saída de $\mathcal{S}$ para ser lido pelo ambiente.

O simulador caixa-preta $\hat{\mathcal{S}}$ será construído demonstrando as ações que ele toma para os casos em que apenas Bob é honesto, apenas Alice é honesta, ambos Alice e Bob são corruptos e ambos Alice e Bob são honestos.

6.2.3.1 Alice Corrupta e Bob Honesto

Primeiramente descreve-se a simulação de $\hat{\mathcal{S}}$ e em seguida demonstra-se a indistinguibilidade da visão do ambiente durante a execução do protocolo real (Protocolo 6.3) pelas partes e na presença do adversário $\mathcal{A}$ e da visão durante a execução do protocolo ideal (protocolo que interage com $\mathcal{F}_{\det(X,Y)}$) pelas partes *dummy* e na presença do adversário $\mathcal{S}$.

As interações de A' com $\hat{\mathcal{S}}$ são aquelas de Alice no protocolo real com $\mathcal{Z}$, TI e Bob. Lembre-se de como o adversário é estático, $\hat{\mathcal{S}}$ corrompe Alice antes da execução do protocolo. Assim, $\hat{\mathcal{S}}$ conhece a matriz de entrada da Alice X. Segue as ações de $\hat{\mathcal{S}}$ em cada interação com A' e $\mathcal{F}_{\det(X,Y)}$.

1. **Na fase de inicialização:**
   - $\hat{\mathcal{S}}$ escolhe $R' \in_R \mathbb{F}_q^{n\times n}$, $Q' \in_R SL(\mathbb{F}_q)$, $U' \in \mathbb{F}_q^{n\times n}$, calcula $V' = R'Q' + U'$, e envia $\mu'_A := (R', V')$ para A';
2. **Ao receber a entrada X de A':**
   - $\hat{\mathcal{S}}$ faz $X' = X$ e escolhe $Y' \in_R \mathbb{F}_q^{n\times n}, P' \in_R SL(\mathbb{F}_q)$;
3. **Ao receber mensagem** EntradaDeBrecebida **de** $\mathcal{F}_{\det(X,Y)}$:
   - $\hat{\mathcal{S}}$ calcula $M' = (P'X' - R')Q' + P'Y'Q' - U'$ e envia $\mu'_1 := (M')$ para A';
4. **Ao receber mensagem** $\mu'_2 := (t)$ **de** A':
   - $\hat{\mathcal{S}}$ verifica se t é consistente com M' e V'. Se esse for o caso, $\hat{\mathcal{S}}$ envia a mensagem (AenviaEntrada, X', $ssid$, sid) para $\mathcal{F}_{\det(X,Y)}$. Se esse não for o caso, $\hat{\mathcal{S}}$ escolhe um $X'' \notin \mathbb{F}_q^{n\times n}$ e envia a mensagem (AenviaEntrada, X'', $ssid$, sid) para $\mathcal{F}_{\det(X,Y)}$;
5. **Ao receber mensagem** EntradaInválida:
   - $\hat{\mathcal{S}}$ entrega essa mensagem para $\mathcal{Z}$ através da interface de A';
6. **Ao receber mensagem** EntradaDeArecebida**ou**BrecebeSaída **de** $\mathcal{F}_{\det(X,Y)}$:
   - $\hat{\mathcal{S}}$ ignora.

Agora se quer provar que a visão do ambiente $\mathcal{Z}$ quando este interage com as partes que executam o protocolo real na presença do adversário real $\mathcal{A}$ é computacionalmente indistinguível da visão de $\mathcal{Z}$ quando este interage com o adversário ideal $\mathcal{S}$ e as partes *dummy* que executam o protocolo ideal com acesso à $\mathcal{F}_{\det(X,Y)}$ e $\mathcal{F}_{IT}$. Mais precisamente, será demonstrado que essas visões são perfeitamente indistinguíveis.

Para tal, observe que o comportamento do adversário no modelo real $\mathcal{A}$, e portanto da sua cópia A', é completamente determinado pelos bits aleatórios de $\mathcal{A}$, $r_{\mathcal{A}}$, pela

entrada X' e pelas mensagens μ'_A e μ'_1. Além disso, independentemente da escolha da estratégia que o adversário $\mathcal{A}$ tome, ele pode desviar do protocolo enviando uma entrada X' diferente daquela prescrita no protocolo ou envia na mensagem μ'_2 um valor de t' que pode não ser um elemento de $\mathbb{F}_q$ ou um valor de t' que não seja consistente com X', μ'_A e μ'_1. Entretanto, em ambos os casos, a simulação captura o mesmo comportamento do Bob honesto. Isto é, ele aborta se $t' \notin \mathbb{F}_q$ e continua a tentar calcular o determinante se t' não for consistente.

Além das estratégias de A' serem copiadas na simulação, observe que as mensagens mu'_A e μ'_1 foram construídas usando a mesma distribuição das mensagens μ_A e μ_1 recebidas pela Alice no protocolo real, pois tanto o TI é incorruptível quanto Bob é honesto, isto é, ele segue as especificações do protocolo. Tem-se também que a a aleatoriedade $r_{\mathcal{A}}$ do adversário $\mathcal{A}$ possui a mesma distribuição daquela recebida no modelo real, porque é $\mathcal{Z}$ quem a distribui. Lembre-se de que $\mathcal{Z}$ de forma igual no modelo real e no modelo ideal a fim de que ele possa ser um distinguidor.

Portanto, sendo as distribuições das mensagens trocadas na execução do protocolo real iguais à execução do protocolo ideal e sendo as estratégias do adversário real simuladas pelo simulador, tem-se que as saídas de $\mathcal{A}$, $\mathcal{S}$, Alice, Bob e suas versões *dummy* possuem a mesma distribuição. Fazendo $\rho =$ Protocolo 6.3 e lembrando que o ambiente $\mathcal{Z}$ só lê a saída das partes e adversários no modelo real e no modelo ideal, segue que para todo $\mathcal{Z}$:

$$\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{\det(X,Y)},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

6.2.3.2 Alice Honesta e Bob Corrupto

Segue-se as mesmas linhas do caso anterior, ou seja, descreve-se a simulação de $\hat{\mathcal{S}}$ e em seguida demonstra-se a indistinguibilidade da visão do ambiente.

As interações de A' com $\hat{\mathcal{S}}$ são aquelas de Bob no protocolo real com $\mathcal{Z}$, TI e Alice. Lembre-se de como o adversário é estático, $\hat{\mathcal{S}}$ corrompe Bob antes da execução do protocolo. Assim, $\hat{\mathcal{S}}$ conhece a matriz de entrada do Bob Y. Segue as ações de $\hat{\mathcal{S}}$ em cada interação com A' e $\mathcal{F}_{\det(X,Y)}$.

1. **Na fase de inicialização:**

   - $\hat{\mathcal{S}}$ escolhe $P' \in_R SL(\mathbb{F}_q)$,$Q' \in_R SL(\mathbb{F}_q)$, $U' \in \mathbb{F}_q^{n\times n}$ e envia $\mu'_B := (P', Q', U')$ para A';

2. **Ao receber a entrada Y de A':**

   - $\hat{\mathcal{S}}$ faz $Y' = Y$, escolhe $X' \in_R \mathbb{F}_q^{n\times n}$, $R' \in_R \mathbb{F}_q^{n\times n}$ e calcula $V' = R'Q' + U'$;

3. **Ao receber mensagem $\mu'_1 := (M)$ de A':**

   - $\hat{\mathcal{S}}$ verifica se M é consistente com P', X', R', Q', Y' e U'. Se esse for o caso, $\hat{\mathcal{S}}$ envia a mensagem (BenviaEntrada, Y', $ssid$, sid) para $\mathcal{F}_{\det(X,Y)}$. Se esse não for o caso, $\hat{\mathcal{S}}$ escolhe um $Y'' \notin \mathbb{F}_q^{n\times n}$ e envia a mensagem (BenviaEntrada, Y'', $ssid$, sid) para $\mathcal{F}_{\det(X,Y)}$;

4. **Ao receber mensagem** (BrecebeSaída, z, $ssid$, sid) **de $\mathcal{F}_{\det(X,Y)}$:**

   - $\hat{\mathcal{S}}$ guarda z, calcula N' e t' e envia $\mu'_2 := (t')$ para A'. $\hat{\mathcal{S}}$ intercepta o valor de z' calculado por A' e verifica se z' é consistente com t', P', Q'. Se esse for o caso, $\hat{\mathcal{S}}$ substitui z' pelo valor z recebido por $\mathcal{F}_{\det(X,Y)}$. Senão, não será feita a substituição;
   - $\hat{\mathcal{S}}$ entrega a saída z' para o ambiente $\mathcal{Z}$ através da interface de A';

5. **Ao receber mensagem** EntradaInválida:

   - $\hat{\mathcal{S}}$ entrega essa mensagem para o ambiente $\mathcal{Z}$ através da interface de A';

6. **Ao receber mensagem** EntradaDeArecebida**ou**EntradaDeBrecebida **de $\mathcal{F}_{\det(X,Y)}$:**

   - $\hat{\mathcal{S}}$ ignora.

A demonstração da indistinguibilidade da visões do ambiente é, neste caso, extremamente similar ao caso anterior.

O comportamento do adversário $\mathcal{A}$ é determinado, neste caso, por $r_{\mathcal{A}}$, pela entrada Y' e pelas mensagens μ'_B e μ'_2. Além disso, como no caso anterior, ele pode desviar do protocolo enviando uma entrada Y' diferente daquela prescrita no protocolo ou envia na mensagem μ'_1 um valor de M' que pode não ser um elemento de $\mathbb{F}_q$ ou um valor de M' que não seja consistente com Y', μ'_A e R'. Entretanto, em ambos os casos, a

simulação captura o mesmo comportamento da Alice honesto. Isto é, ela aborta se $M' \notin \mathbb{F}_q$ e continua a tentar calcular t' se M' não for consistente.

A principal diferença deste caso para o caso anterior, é que o simulador precisa se preocupar com a saída do Bob. O simulador não conhece a entrada X da parte *dummy* que representa a Alice honesta. Além disso, o único valor que ele tem relacionado à entrada "real"da parte *dummy* honesta é a saída z recebida por $\mathcal{F}_{\text{det}(X,Y)}$. Então, se o adversário real A' decidir que a saída do Bob corrupto é consistente com as entradas e mensagens recebidas até então, o simulador tem que garantir que isso é possível na simulação. É por isso, então, que o simulador faz a verificação da saída do Bob corrupto que está sobre o controle do adversário A' e troca pela saída recebida por $\mathcal{F}_{\text{det}(X,Y)}$, se esse for o caso.

Observe que a simulação espera o tempo que for necessário para o Bob corrupto enviar a entrada e calcular a saída, mesmo que isso implique em esperar para sempre. Contudo, esse é o comportamento esperado da simulação para que ela capture o comportamento do Bob corrupto.

Dessa maneira, sendo o comportamento do adversário capturado pelo simulador e desde que as entradas, saídas e mensagens trocadas possuam o mesma distribuição, pode-se concluir que a visão de qualquer ambiente interagindo com $\mathcal{A}$ e as partes que executam o protocolo real possui a mesma distribuição da visão deste mesmo ambiente interagindo com $\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}}$ e as partes *dummy* que executam o protocolo ideal que tem acesso à $\mathcal{F}_{\text{det}(X,Y)}$. Então, fazendo $\rho =$ Protocolo 6.3, tem-se para todo $\mathcal{Z}$:

$$\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{\text{det}(X,Y)},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

6.2.3.3 Alice e Bob Corruptos

Neste caso, $\mathcal{A}$ irá gerar todas as trocas de mensagens entre Alice e Bob na execução do protocolo real e $\hat{\mathcal{S}}$ apenas entrega todas as mensagens entre as partes *dummy* e a funcionalidade ideal $\mathcal{F}_{\text{det}(X,Y)}$. Portanto, as saídas de $\mathcal{A}$, de $\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}}$, de Alice e Bob e suas versões *dummy* possuam distribuições idênticas. Então, fazendo $\rho =$ Protocolo 6.3, tem-se para todo $\mathcal{Z}$:

$$\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{\text{det}(X,Y)},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

6.2.3.4 Alice e Bob Honestos

Quando nenhuma parte está corrompida, $\hat{\mathcal{S}}$ simula a funcionalidade ideal do TI e recebe a mensagem (BrecebeSaída, $ssid, sid$) e a mensagem (EntradaDeArecebida, $ssid, sid$) ou (EntradaDeBrecebida, $ssid, sid$) de $\mathcal{F}_{\det(X,Y)}$, além da transcrição de todas as trocas de mensagens entre Alice e Bob.

Observe que quando nenhuma das partes é corrompida, tanto Alice quanto Bob, quanto as versões *dummy* deles se comportarão da mesma forma, produzindo saídas com a mesma distribuição. Isto ocorre porque as entradas foram enviadas igualmente pelo ambiente e como em ambos os casos eles seguem o que foi prescrito pelo protocolo real, eles terão saídas do protocolo real e do protocolo ideal com a mesma distribuição.

Logo, fazendo $\rho =$ Protocolo 6.3, tem-se para todo $\mathcal{Z}$:

$$\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{\det(X,Y)},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

Como a visão de todo ambiente é indistinguível em todos os quatro casos possíveis (Alice corrupta, Bob honesto; Alice honesta, Bob corrupto; Alice e Bob corruptos; Alice e Bob honestos), tem-se que o Protocolo 6.3 UC-realiza $\mathcal{F}_{\det(X,Y)}$.

6.3 AUTOVALOR

Autovalores e autovetores são conceitos intimamente relacionados. Eles são extremamente importantes no estudo de matrizes. Em especial, com eles pode-se fatorar e diagonalizar matrizes. Considerando outras aplicações, como na física e engenharia, pode-se citar que os autovalores e autovetores são importantes na análise de níveis de energia de átomos e moléculas, no análise de estabilidade de corpos, no estudo de fenômenos de vibração, entre outros.

Para se definir autovalores e autovetores, define-se primeiro transformação linear.

Definição 34. *Sejam V e W dois espaços vetoriais. Uma transformação linear é uma função de V em W, $T : V \to W$, satisfazendo as seguintes condições:*

- *Para todo $\vec{u}, \vec{v} \in V, T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$;*

- *Para todo escalar* k *e* $\vec{v} \in V$, $T(k\vec{v}) = kT(\vec{v})$.

Com a definição de transformação linear, pode-se definir autovalores e autovetores.

Definição 35. *Seja* $T : V \to V$ *uma transformação linear. Se existirem* $\vec{v} \in V, \vec{v} \neq 0$ *e* $\lambda \in \mathbb{F}_q$ *tais que* $T(\vec{v}) = \lambda\vec{v}$, λ *é um* autovalor *de* T *e* $\vec{v}$ *é um* autovetor *de* T *associado a* λ.

Desde que toda matriz $m \times n$ com elementos em $\mathbb{F}_q$ está associado a uma transformação linear de $\mathbb{F}_q^n$ em $\mathbb{F}_q^m$ e vice-versa, (isto é toda transformação linear de $\mathbb{F}_q^n$ em $\mathbb{F}_q^m$ também está associado a uma matriz $m \times n$ com elementos em $\mathbb{F}_q$), será considerado transformações lineares como matrizes. Entretanto, como as transformações lineares de interesse são aquelas de $\mathbb{F}_q^n$ em $\mathbb{F}_q^n$, as matrizes de interesse são as quadradas $n \times n$.

A partir da definição de autovalores e autovetores, a equação $A\vec{v} = \lambda\vec{v} \Rightarrow A\vec{v} = (\lambda I)\vec{v} \Rightarrow (A - \lambda I)\vec{v} = 0$, onde I é a matriz identidade. A única maneira de se encontrar as soluções não nulas $\vec{v}$ (os autovetores) de $(A - \lambda I)\vec{v} = 0$ é tendo o $\det(A - \lambda I) = 0$. Considerando a definição de determinante dada na seção anterior e estendendo a equação $\det(A - \lambda I)$ tem-se um polinômio em λ de grau n: $\det(A - \lambda I) = (a_{11} - \lambda)...(a_{nn} - \lambda) +$ termos de grau menor do que n, onde a_{ij} é o elemento da matriz A que está na i-ésima linha e j-ésima coluna. Esse polinômio é chamado de polinômio característico. Observe que as raízes deste polinômio são os valores de λ associados aos autovetores $\vec{v}$. Portanto, as raízes deste polinômio são os autovalores da matriz A.

O protocolo apresentado a seguir para se calcular autovalor de forma distribuída pretende calcular os autovalores da matriz $X+Y$, sendo X a entrada da Alice e Y a entrada do Bob através do cálculo das raízes do polinômio característico $\det(X + Y - \lambda I)$. A principal desvantagem dessa abordagem reside no fato de que se n for muito grande, encontrar autovalores através do polinômio característico pode ser muito custoso.

6.3.1 Implementação

O protocolo apresentado a seguir tem como entrada as matrizes $X \in \mathbb{F}_q^{n \times n}$ de Alice e $Y \in \mathbb{F}_q^{n \times n}$ de Bob e gera como saída para Bob o vetor $\vec{\lambda}$ que contém os autovalores da matriz $X + Y$. Alice não recebe nenhuma saída. Além disso, deseja-se que o protocolo seja privado, isto é, não deve ser possível extrair as entradas das partes através das trocas de mensagens e da saída calculada.

Esquematicamente, tem-se:

Tabela 6.4: Funcionalidade de Autovalores

	Alice	Bob
Entradas	$X \in \mathbb{F}_q^{n \times n}$	$Y \in \mathbb{F}_q^{n \times n}$
Saídas	$\perp$	$\vec{\lambda}$

O Protocolo 6.4 trabalha da seguinte forma: No início do protocolo, o inicializador confiável (TI) escolhe matrizes aleatórias para proteger as entradas de Alice e de Bob. O TI envia uma matriz aleatória e o resultado de uma função das matrizes aleatórias para o Bob (R, V) e envia o restante das matrizes aleatórias para Alice (P, Q, U). Aqui também vale a observação acima de que o TI é o mesmo para esse protocolo e para os protocolos 6.2 e 6.3 que serão usados como subprotocolos e a que fase de inicialização do protocolo principal e dos subprotocolos ocorrem simultaneamente. Então, de fato, o TI escolhe além dessas matrizes aleatórias (P, Q, U, R) outras matrizes aleatórias para ser as matrizes P, Q, U, R usados no Protocolo 6.3 e as matrizes X_0 e Y_0 usadas no Protocolo 6.2 (além de calcular o vetor $\overrightarrow{s_0}$ utilizado no Protocolo 6.2).

Após receber as mensagens μ_A e μ_B do TI, Alice e Bob invocam uma instância do Protocolo 6.2, sendo a matriz P a entrada da Alice e sendo Y a entrada do Bob. Observe que, neste caso, Alice invoca o Protocolo 6.2 no papel de Bob e Bob o faz no papel da Alice. Portanto, ao final da execução do Protocolo 6.2, Alice recebe a matriz $C = PY - R$ e Bob recebe a mesma matriz R que foi inicialmente distribuída pelo TI.

Alice envia a entrada dela para o Bob através da mensagem μ_1. Especificamente, Alice calcula e envia para Bob $M := CQ + P(X - \lambda I)Q - U$. Bob verifica se a distribuição da mensagem μ_1 recebida é a mesma da esperada. Se não for, aborta. Se for, Bob retira parte da aleatoriedade adicionada por Alice na mensagem μ_1, isto é, Bob calcula $N = M + V$. Finalmente, Bob calcula os autovalores de $X + Y$, calculando as soluções do polinômio característico $\det(N) = 0$.

A figura 6.4 mostra a troca de mensagens entre Alice, Bob e TI, além das entradas de Alice e Bob. Sem perda de generalidade, a saída de Bob do protocolo para produto de matrizes é ignorado nessa imagem.

Protocolo 6.4 Protocolo para Autovalor

1: O Inicializador Confiável escolhe $P \in_R SL(\mathbb{F}_q^n), R \in_R \mathbb{F}_q^{n\times n}, Q \in_R SL(\mathbb{F}_q^n), V = RQ + U, U \in_R \mathbb{F}_q^{n\times n}$ e envia $\mu_A := (P, Q, U)$ para Alice e $\mu_B := (R, V)$ para Bob.

2: Alice e Bob executam o Protocolo 6.2 com P como entrada da Alice e Y como entrada de Bob, onde $Y \in \mathbb{F}_q^{n\times n}$. Ao final da execução do Protocolo 6.2 Alice tem como resultado a matriz C.

3: Alice faz $T := X - \lambda I$, onde $X \in \mathbb{F}_q^{n\times n}$; calcula $M := CQ + PTQ - U$ e envia $\mu_1 := (M)$ para Bob.

4: Bob aborta se $M \notin \mathbb{F}_q^{n\times n}$. Caso contrário, Bob faz $N := M + V$ e calcula a equação $\det(N) = 0$. Ao final, Bob faz $z := \vec{\lambda}$, onde $\vec{\lambda}$ é o vetor cujas entradas são as soluções da equação $\det(N) = 0$.

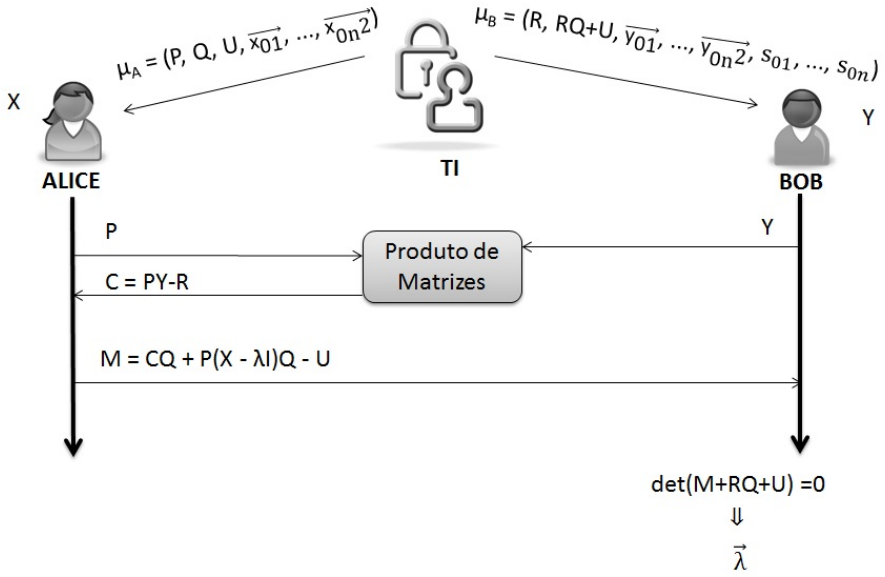

Figura 6.4: Troca de Mensagens no Protocolo para Cálculo de Autovalores

Correção: Deseja-se mostrar que a saída do Protocolo 6.4 é de fato um vetor cujas entradas são as soluções da equação $\det(X + Y - \lambda I) = 0$. Observe que este vetor pode ter nenhuma entrada ou no máximo n entradas.

Desde que a saída de Bob é derivada da equação $\det(N) = 0$, analisa-se N. De fato,

$$\begin{aligned} N: &= M + V = CQ + PTQ - U + RQ + U \\ &= (PY - R)Q + P(X - \lambda I)Q - U + RQ + U \\ &= P(X + Y - \lambda I)Q. \end{aligned}$$

Como as matrizes consideradas são quadradas, a propriedade 7 de determinantes afirma

que $\det(AB) = \det(A)\det(B)$. Uma vez que o TI escolheu $P \in_R SL(\mathbb{F}_q^n)$ e $Q \in_R SL(\mathbb{F}_q^n)$, tem-se que $\det(P) \neq 0$ e $\det(Q) \neq 0$. Então, tem-se:

$$\begin{aligned} \det(N) = 0 &\Leftrightarrow \det(P(X+Y-\lambda I)Q) = 0 \\ &\Leftrightarrow \det(P)\det(X+Y-\lambda I)\det(Q) = 0 \\ &\Leftrightarrow \det(X+Y-\lambda I) = 0. \end{aligned}$$

Alice não consegue extrair Y através de C, dado que o Protocolo 6.2 é seguro. Por outro lado, Bob não consegue extrair X (nem P) de μ_1 porque ele não conhece Q nem U. Além disso, desde que o determinante é um número em $\mathbb{F}_q$ e que, em geral, $\det(A+B) \neq \det(A) + \det(B)$, tem-se que Bob não extrai X através de sua saída.

Apresenta-se a prova de que o Protocolo 6.4 é seguro no *framework* UC a seguir. Na próxima subseção apresenta-se as funcionalidades ideais para o Inicializador Confiável e para o Protocolo 6.4. Na subseção 6.3.3, constrói-se o simulador e demonstra-se a indistinguibilidade.

6.3.2 Funcionalidades Ideais

Funcionalidade Ideal para o Inicializador Confiável $\mathcal{F}_{TI}^2$

- Quando ativado, escolha $P \in_R SL(\mathbb{F}_q)$, $Q \in_R SL(\mathbb{F}_q)$, $R \in_R \mathbb{F}_q^{n\times n}$ e $U \in_R \mathbb{F}_q^{n\times n}$ e calcule $V = RQ + U$. Envie P, Q e U para Alice R e V para Bob.

Como foi discutido anteriormente, não é explicitado nessa funcionalidade para TI as matrizes usadas no subprotocolo que calcula o produto entre matrizes. Contudo, isso pode ser feito sem perda de generalidade, pois o subprotocolo já foi provado ser UC seguro. Um ponto interessante a ser notado é que a funcionalidade ideal para TI apresentado anteriormente é extremamente similar à funcionalidade equivalente do caso para determinante. A diferença está nos dados entregues para Alice e os entregue para Bob.

Funcionalidade Ideal para o Cálculo de Autovalores $\mathcal{F}_{autovalor(x,y)}$

- Ao receber uma mensagem (AenviaEntrada, X, *sid*) de A:
  - Ignore qualquer mensagem AenviaEntrada subsequente;
  - Se $X \notin \mathbb{F}_q^{n \times n}$, então envie uma mensagem EntradaInválida para A e B e pare;
  - Se nenhuma mensagem BenviaEntrada foi recebida anteriormente de B, guarde X e *sid* e envie uma mensagem (EntradaDeArecebida, *sid*) para B e $\mathcal{S}$. De outra forma, calcule $z = \vec{\lambda}$, tal que os elementos de $\vec{\lambda}$ são as soluções da equação $\det(X + Y - \lambda I) = 0$, onde I é a matriz identidade $n \times n$, e envie uma mensagem (BrecebeSaída, z, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$;
- Ao receber uma mensagem (BenviaEntrada, Y, *sid*) de B:
  - Ignore qualquer mensagem BenviaEntrada subsequente;
  - Se $Y \notin \mathbb{F}_q^{n \times n}$, então envie uma mensagem EntradaInválida para A e B e pare;
  - Se nenhuma mensagem AenviaEntrada foi recebida anteriormente de A, guarde Y e *sid*, e envie uma mensagem (EntradaDeBrecebida, *sid*) para A e $\mathcal{S}$. De outra forma, calcule $z = \vec{\lambda}$, tal que os elementos de $\vec{\lambda}$ são as soluções da equação $\det(X + Y - \lambda I) = 0$, onde I é a matriz identidade $n \times n$, e envie uma mensagem (BrecebeSaída, z, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$.

6.3.3 Prova de Segurança Universalmente Composta

Dado o Protocolo 6.4 (protocolo real) executado por Alice e Bob na presença do adversário real $\mathcal{A}$, deseja-se construir no modelo ideal um adversário ideal $\mathcal{S}$ que executa junto com as partes *dummy* um protocolo ideal que tem acesso a funcionalidade $\mathcal{F}_{autovalor(x,y)}$. A construção de $\mathcal{S}$ deve objetivar que todo ataque que o adversário real possa realizar no modelo real seja simulado pelo adversário ideal no modelo ideal.

O adversário ideal $\mathcal{S}$ considerado é chamado de simulador *shell*. $\mathcal{S}$ executa dentro dele uma cópia A' do adversário real $\mathcal{A}$ e um simulador caixa-preta $\hat{\mathcal{S}}$. De fato, o que será

construído é o simulador caixa-preta $\hat{\mathcal{S}}$. Para tal, descreve-se as ações que $\hat{\mathcal{S}}$ toma para os casos em que apenas Bob é honesto, apenas Alice é honesta, ambos Alice e Bob são corruptos e ambos Alice e Bob são honestos.

Em todos os casos apresentados a seguir, descreve-se a simulação de $\hat{\mathcal{S}}$ e em seguida demonstra-se a indistinguibilidade da visão do ambiente durante a execução do protocolo real (Protocolo 6.4) pelas partes e na presença do adversário $\mathcal{A}$ e da visão durante a execução do protocolo ideal (protocolo que interage com $\mathcal{F}_{autovalor(x,y)}$) pelas partes *dummy* e na presença do adversário $\mathcal{S}$.

Lembre-se que a comunicação de $\mathcal{S}$ com o ambiente $\mathcal{Z}$ ocorre da seguinte forma: toda entrada que $\mathcal{S}$ recebe de $\mathcal{Z}$ é escrito na fita de entrada do adversário A', sem alteração, e todo valor de saída de A' é copiado por $\mathcal{S}$, possivelmente modificado, e enviado para a fita de saída de $\mathcal{S}$ para ser lido pelo ambiente.

6.3.3.1 Alice Corrupta e Bob Honesto

As interações de A' com $\hat{\mathcal{S}}$ são aquelas de Alice no protocolo real com $\mathcal{Z}$, TI e Bob. Segue as ações de $\hat{\mathcal{S}}$ em cada interação com A' e com $\mathcal{F}_{autovalor(x,y)}$.

1. **Na fase de inicialização:**
   - $\hat{\mathcal{S}}$ escolhe $P' \in_R SL(\mathbb{F}_q)$, $Q' \in_R SL(\mathbb{F}_q)$, $R' \in_R \mathbb{F}_q^{n\times n}$, $U' \in \mathbb{F}_q^{n\times n}$, calcula $V' = R'Q' + U'$, e envia $\mu'_A := (P', Q', U')$ para A';
2. **Ao receber a entrada X de A':**
   - $\hat{\mathcal{S}}$ faz $X' = X$, escolhe $Y' \in_R \mathbb{F}_q^{n\times n}$ e calcula $V' = R'Q' + U'$;
3. **Ao receber mensagem $\mu'_1 := (M)$ de A'**:
   - $\hat{\mathcal{S}}$ verifica se M é consistente com X', Y', P', R', Q' e V'. Se esse for o caso, $\hat{\mathcal{S}}$ envia a mensagem (AenviaEntrada, X', $ssid, sid$) para $\mathcal{F}_{autovalor(x,y)}$. Se esse não for o caso, $\hat{\mathcal{S}}$ escolhe um $X'' \notin \mathbb{F}_q^{n\times n}$ e envia a mensagem (AenviaEntrada, X'', $ssid, sid$) para $\mathcal{F}_{autovalor(x,y)}$;
4. **Ao receber mensagem** EntradaInválida:
   - $\hat{\mathcal{S}}$ entrega essa mensagem para $\mathcal{Z}$ através da interface de A';

5. **Ao receber mensagem** EntradaDeArecebida **ou** EntradaDeBrecebida **ou** BrecebeSaída **de** $\mathcal{F}_{autovalor(x,y)}$:

   - $\hat{\mathcal{S}}$ ignora.

A análise de indistinguibilidade é a mesma daquela do Protocolo 6.3. De fato, o comportamento do adversário no modelo real $\mathcal{A}$, e portanto da sua cópia A', é completamente determinado pelos bits aleatórios de $\mathcal{A}$, $r_{\mathcal{A}}$, pela entrada X' e pelas mensagens μ'_A e μ'_1. Qualquer que seja a estratégia escolhida pelo adversário $\mathcal{A}$, ele somente pode desviar da especificação do protocolo enviando uma entrada X' diferente daquela prescrita no protocolo ou envia na mensagem μ'_1 um valor de M' que pode não ser um elemento de $\mathbb{F}_q^{n\times n}$ ou um valor de M' que não seja consistente com X', μ'_A e Y', R'. Entretanto, em ambos os casos, a simulação captura o mesmo comportamento do Bob honesto. Isto é, ele aborta se $M' \notin \mathbb{F}_q^{n\times n}$ e continua a tentar calcular os autovalores se M' não for consistente.

Além das estratégias de A' serem copiadas na simulação, observe que as propriedades do TI garantem que mensagem mu'_A foi construída usando a mesma distribuição da mensagem μ_A recebida pela Alice no protocolo real. Falta considerar as aleatoriedade de $r_{\mathcal{A}}$ do adversário $\mathcal{A}$ para determinar que o comportamento dele é capturado perfeitamente na simulação. Entretanto, como é $\mathcal{Z}$ quem distribui $r_{\mathcal{A}}$, ele o faz igualmente no modelo real e no modelo ideal.

Então, como os ataques do adversário real são capturados por $\mathcal{S}$ tem-se que as saídas de $\mathcal{A}$, $\mathcal{S}$, Alice, Bob e suas versões *dummy* possuem a mesma distribuição. Portanto, pode-se afirmar para todo ambiente $\mathcal{Z}$ (fazendo $\rho =$ Protocolo 6.4):

$$\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{autovalor(x,y)},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

6.3.3.2 Alice Honesta e Bob Corrupto

As interações de A' com $\hat{\mathcal{S}}$ são aquelas de Bob no protocolo real com $\mathcal{Z}$, TI e Alice. Segue as ações de $\hat{\mathcal{S}}$.

1. **Na fase de inicialização:**

   - $\hat{\mathcal{S}}$ escolhe $Q' \in_R SL(\mathbb{F}_q)$, $U' \in \mathbb{F}_q^{n\times n}$, $R' \in_R \mathbb{F}_q^{n\times n}$, calcula $V' = R'Q' + U'$ e envia $\mu'_B := (R', V')$ para A';

2. **Ao receber a entrada Y de A':**

   - $\hat{\mathcal{S}}$ faz $Y' = Y$, escolhe $X' \in_R \mathbb{F}_q^{n\times n}, P' \in_R SL(\mathbb{F}_q)$;

3. **Ao receber mensagem** EntradaDeArecebida:

   - $\hat{\mathcal{S}}$ calcula $T' = X' - \lambda I$, onde I é a matriz identidade $n \times n$, e $M' = (P'Y' - R')Q' + V'$. Então, $\hat{\mathcal{S}}$ envia a mensagem (BenviaEntrada, Y', $ssid, sid$) para $\mathcal{F}_{autovalor(x,y)}$;

4. **Ao receber mensagem** (BrecebeSaída, z, $ssid, sid$) **de** $\mathcal{F}_{autovalor(x,y)}$:

   - $\hat{\mathcal{S}}$ guarda z, calcula N' e envia $\mu'_1 := M'$ para A'. $\hat{\mathcal{S}}$ intercepta o valor de z' calculado por A' e verifica se z' é consistente com X', Y'. Se esse for o caso, $\hat{\mathcal{S}}$ substitui z' pelo valor z recebido por $\mathcal{F}_{autovalor(x,y)}$. Senão, não será feita a substituição;
   - $\hat{\mathcal{S}}$ entrega a saída z' para o ambiente $\mathcal{Z}$ através da interface de A';

5. **Ao receber mensagem** EntradaInválida:

   - $\hat{\mathcal{S}}$ entrega essa mensagem para o ambiente $\mathcal{Z}$ através da interface de A';

6. **Ao receber mensagem**EntradaDeBrecebida **de** $\mathcal{F}_{autovalor(x,y)}$:

   - $\hat{\mathcal{S}}$ ignora.

Para demonstrar a indistinguibilidade das visões do ambiente neste caso, basta fazer as seguintes observações:

- Seja qual for a estratégia de $\mathcal{A}$, o simulador se comporta de forma igual à Alice honesta. No caso especial, do adversário seguir o protocolo, a saída de Bob *dummy* deve ter a mesma distribuição da saída do Bob na execução do protocolo real. Esse comportamento é garantido através da verificação da saída do Bob corrupto que está sobre o controle do adversário A' pelo simulador. Se a saída for consistente com as informações fornecidas pelo simulador, significa que o Bob corrupto está seguindo o protocolo. Então, o simulador troca pela saída recebida por $\mathcal{F}_{autovalor(x,y)}$, se esse for o caso;

- As mensagens simuladas μ'_B, μ'_1 possuem a mesma distribuição das suas versões respectivas no modelo real.

Portanto, como o ambiente constrói sua visão a partir das entradas e saídas de $\mathcal{A}$, $\mathcal{S}$, Alice, Bob e suas respectivas versões *dummy*, e considerando ρ = Protocolo 6.4, pode-se afirmar que para todo $\mathcal{Z}$:

$$\text{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{autovalor(x,y)},\mathcal{S}^{\mathcal{A},\hat{\mathcal{S}}},\mathcal{Z}}.$$

6.3.3.3 Alice e Bob Corruptos ou Alice e Bob Honestos

A análise é exatamente igual aquela feita para o Protocolo 6.3.

Dado que para todo ambiente a sua visão é indistinguível em todos os quatro casos possíveis (Alice corrupta, Bob honesto; Alice honesta, Bob corrupto; Alice e Bob corruptos; Alice e Bob honestos), tem-se que o Protocolo 6.4 UC-realiza $\mathcal{F}_{autovalor(x,y)}$.

6.4 AUTOVETOR

A motivação e a definição de autovetor já foram feitas na seção imediatamente anterior. Resumidamente, autovetores são definidos de acordo com a uma determinada transformação linear que pode ser representada por uma matriz. Um autovetor é um vetor não nulo em um espaço vetorial tal que quando aplicado a transformação linear considerada no vetor, produz um múltiplo do próprio vetor. O fator que multiplica o autovetor na saída da aplicação da transformação linear é chamado de autovalor e embora não especificado, cada autovalor está associado a um autovetor.

O protocolo desta seção calcula o autovetor associado ao i-ésimo (de acordo com uma ordem lexicográfica) autovalor da matriz $X + Y$. Esse protocolo utiliza o Protocolo 6.2 e o Protocolo 6.4 como subprotocolos.

6.4.1 Implementação

Especificamente, o protocolo apresentado considera que Alice possui como entrada uma matriz $X \in \mathbb{F}_q^{n \times n}$ e Bob possui como entrada um par (Y, i), onde $Y \in \mathbb{F}_q^{n \times n}$ e

$i \in_R \{1, ..., n\}$. Ao término da execução do protocolo, Bob recebe o autovetor associado ao i-ésimo autovalor da matriz $X+Y$, isto é, Bob recebe $z = \vec{s}$ tal que $(X+Y)\vec{s} = \lambda_i \vec{s}$, onde λ_i é o i-ésimo autovalor da matriz $X + Y$.

Tabela 6.5: Funcionalidade de Autovetores

	Alice	Bob
Entradas	$X \in \mathbb{F}_q^{n \times n}$	$Y \in \mathbb{F}_q^{n \times n}$, $i \in_R \{1, ..., n\}$
Saídas	$\perp$	$z = \vec{s}$

O protocolo apresentado a seguir funciona da seguinte maneira: O inicializador confiável escolhe as matrizes $R \in_R \mathbb{F}_q^{n \times n}, P \in_R SL(\mathbb{F}_q), Q \in_R SL(\mathbb{F}_q), U \in_R \mathbb{F}_q^{n \times n}$, calcula $V = RQ + U$ e envia $\mu_A := (R, V)$ para Alice e $\mu_B := (P, Q, U)$ para Bob. Esses dados são necessários para "esconder"as entradas de Alice e Bob. Como observado nos casos anteriores, o inicializador confiável é o mesmo para o protocolo em estudo e para os protocolos 6.2 e 6.4. Além disso a fase de inicialização do protocolo para autovetor e de seus subprotocolos ocorre ao mesmo tempo.

Após receber os dados enviados pelo inicializador confiável, Alice e Bob executam o Protocolo 6.2. A entrada da Alice é a sua entrada X e a entrada do Bob é P. Ao término da execução do Protocolo 6.2, Alice recebe novamente a matriz R e Bob recebe a matriz $C = PX - R$.

A seguir, Alice e Bob executam o Protocolo 6.4 para encontrar os autovalores da matriz $X + Y$. Portanto, a entrada da Alice para esse protocolo é X e a entrada de Bob para esse protocolo é Y. Ao término da execução do Protocolo 6.4, Alice nada recebe e Bob recebe o vetor $\vec{\lambda}$, onde os elementos do vetor são os autovalores de $X + Y$. Como a matriz $X + Y$ tem dimensão $n \times n$, $\vec{\lambda}$ possui no máximo n elementos. De fato, se $\mathbb{F}_q$ é um corpo algebricamente fechado, todos os elementos de $\vec{\lambda}$ são elementos de $\mathbb{F}_q$.

Bob, então, seleciona a i-ésima entrada de λ,λ_i (de acordo com uma ordem lexicográfica previamente estabelecida), calcula $T := Y - \lambda_i I$, $M := CQ + PTQ - U$ e envia $\mu_1 := (M)$. Ao receber μ_1, Alice aborta se $\mu_1 \notin \mathbb{F}_q^{n \times n}$. Caso contrário, ela calcula $N := M + V$ e $\vec{s}$ tal que $N\vec{s} = 0$. Finalmente, Bob retira a aleatoriedade que protege a entrada dele, fazendo $\vec{v} = Q^{-1}\vec{s}$.

Observe que outros métodos para calcular o autovetor correspondente a um autovalor existem, mas geralmente eles são baseados em estimação. Como no caso dos métodos

da potência com translação da origem ou da iteração inversa com translação da origem. Além disso, o protocolo apresentado nesta seção é elegantemente implementado usando os protocolos já definidos para multiplicação de matrizes, cálculo de determinantes e cálculo de autovetores. Como todos esses protocolos foram provados seguros no *framework* UC, tem-se que a prova de segurança desse protocolo é facilitada. Além do mais, a análise de segurança desse protocolo demonstra toda a potência do Teorema da Composição, isto é, utiliza-se a propriedade de projeto modular de protocolos, análise de protocolos complexos a partir de blocos de construção e garantia de que a segurança é mantida quando este protocolo é executado em um ambiente computacional multiparte e desconhecido.

Outro ponto importante a se observar é que apesar do protocolo ser provado seguro no *framework* UC, se Alice e Bob repetem as entradas em até n sucessivas instâncias do protocolo para calcular autovetores, um Bob corrupto pode obter a entrada da Alice da seguinte forma: Bob conhece $\vec{\lambda}$, então ele pode montar a matriz diagonal D que possui na sua diagonal principal os autovalores de $\vec{\lambda}$. Se ele executar o protocolo para cálculo de autovetores n vezes variando o índice i de 1 a n, Bob pode montar a matriz de autovetores L. Caso a matriz de autovetores for inversível, Bob pode usar o Teorema de Decomposição e calcular $X + Y = LDL^{-1}$. Com a matriz $X + Y$, Bob pode encontrar facilmente X. Entretanto, esse tipo de ataque pode acontecer em todo protocolo que calcule o autovetor associado a um autovalor quando Bob conhece os autovalores da matriz $X + Y$ e as entradas de Alice e Bob se repetem.

Protocolo 6.5 Protocolo para Autovetor

1: O Inicializador Confiável escolhe $R \in_R \mathbb{F}_q^{n\times n}, P \in_R SL(\mathbb{F}_q), V = RQ + U, Q \in_R SL(\mathbb{F}_q), U \in_R \mathbb{F}_q^{n\times n}$ e envia $\mu_A := (R, V)$ para Alice e $\mu_B := (P, Q, U)$ para Bob.

2: Alice e Bob executam o Protocolo 6.2 com X como entrada da Alice, onde $X \in \mathbb{F}_q^{n\times n}$, e P como entrada de Bob. Ao final da execução do protocolo Bob tem como resultado a matriz $C = PX - R$.

3: Alice e Bob executam o Protocolo 6.4 com X como entrada da Alice e Y como entrada de Bob, onde $Y \in \mathbb{F}_q^{n\times n}$). Ao final da execução do protocolo Bob tem como resultado o vetor $\vec{\lambda}$ dos autovalores da matriz $[X + Y]$.

4: Bob escolhe $i \in_R \{1, ..., n\}$ e toma o i-ésimo valor de λ,λ_i. Bob faz $T := Y - \lambda_i I$; calcula $M := CQ + PTQ - U$ e envia $\mu_1 := (M)$ para Alice.

5: Alice aborta se $\mu_1 \notin \mathbb{F}_q^{n\times n}$. Caso contrário, Alice faz $N := M + V$ e calcula $\vec{s}$ tal que $N\vec{s} = 0$ e envia$\mu_2 := (\vec{s})$ para Bob.

6: Ao receber μ_2, Bob calcula $\vec{v} = Q^{-1}\vec{s}$.

A figura 6.5 mostra a troca de mensagens entre Alice, Bob e TI e as entradas das partes. Sem perda de generalidade, a saída de Alice no protocolo de produto de matrizes não é mostrado nessa figura.

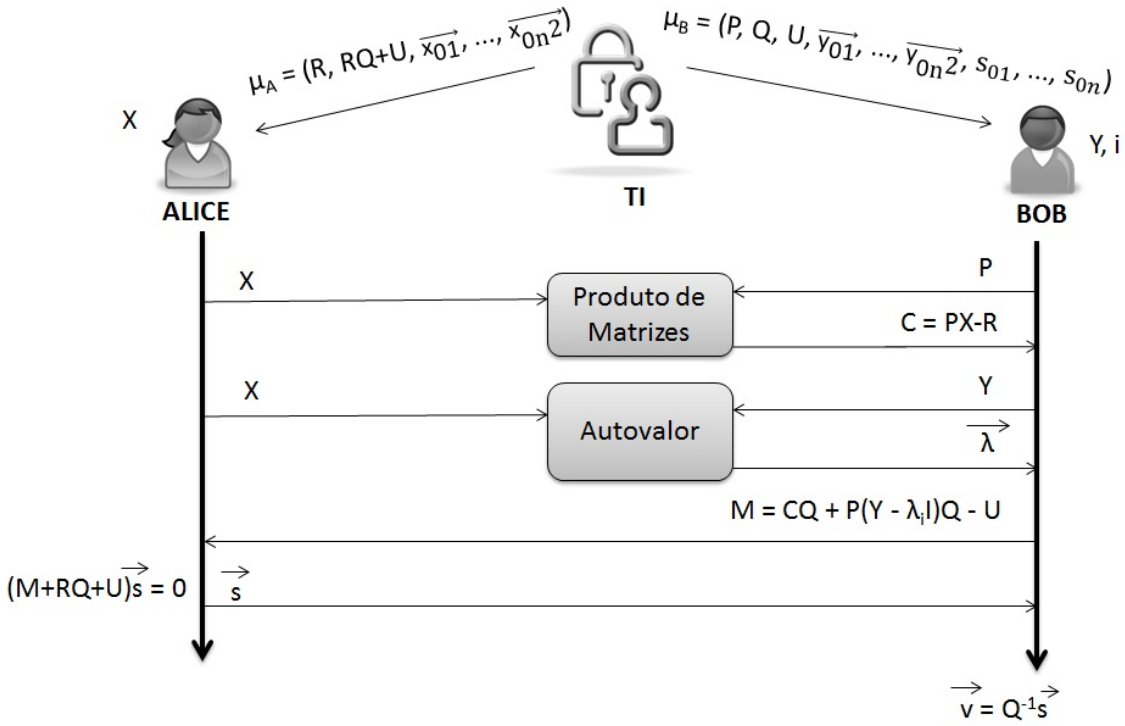

Figura 6.5: Troca de Mensagens no Protocolo para Cálculo de Autovetores

Correção : Deseja-se verificar se a saída de Bob são os autovetores associados ao i-ésimo autovalor da matriz $X + Y$. Tem-se que

$$\begin{aligned} N: &= M + V = CQ + PTQ - U + RQ + U \\ &= (PX - R)Q + P(Y - \lambda_i I)Q - U + RQ + U \\ &= P(X + Y - \lambda_i I)Q. \end{aligned}$$

Alice calcula $P(X + Y - \lambda_i I)Q\vec{s} = P\vec{0} \Rightarrow (X + Y - \lambda_i I)Q\vec{s}$. Portanto, para Bob encontrar a solução de $(X + Y - \lambda_i I)\vec{v} = 0$, basta que ele calcule $\vec{v} = Q^{-1}\vec{s}$.

Alice não consegue extrair Y de μ_1, desde que ela não conhece P nem Q. Por outro lado, Bob não consegue extrair X de C nem de $\vec{\lambda}$, desde que os protocolos 6.2 e 6.4 são UC seguros. Bob também não consegue extrair X de μ_2 e da saída $\vec{v}$ desde que mesmo se Bob montar o sistema linear $(X + Y - \lambda_i I)Q\vec{s} = 0$, fazendo os valores os elementos x_{ij} como incógnitas, ele terá n equações e n^2 incógnitas.

A seguir apresenta-se as funcionalidades ideais pertinentes à prova de segurança no *framework* UC.

6.4.2 Funcionalidades Ideais

Nas funcionalidades ideais seguintes, considere A a Alice, B o Bob e $\mathcal{S}$ o adversário ideal.

Funcionalidade Ideal para o Cálculo do Autovetor $\mathcal{F}_{autovetor(X,Y)}$

- Ao receber uma mensagem (AenviaEntrada, X, *sid*) de A:
  - Ignore qualquer mensagem AenviaEntrada subsequente;
  - Se $X \notin \mathbb{F}_q^{n \times n}$, então envie uma mensagem EntradaInválida para A e B e pare;
  - Se nenhuma mensagem BenviaEntrada foi recebida anteriormente de B, guarde X e *sid* e envie uma mensagem (EntradaDeArecebida, *sid*) para B e $\mathcal{S}$. De outra forma, calcule os autovalores λ da matriz $X + Y$ e selecione λ_i, o i-ésimo autovalor segundo alguma ordem pré-estabelecida. Compute $z = \vec{v}$ tal que $(X + Y)\vec{v} = \lambda_i \vec{v}$ e envie uma mensagem (BrecebeSaída, z, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$.
- Ao receber uma mensagem (BenviaEntrada, (Y, i), *sid*) de B:
  - Ignore qualquer mensagem BenviaEntrada subsequente;
  - Se $Y \notin \mathbb{F}_q^{n \times n}$ ou se $i \notin \{1, ..., n\}$, então envie uma mensagem EntradaInválida para A e B e pare;
  - Se nenhuma mensagem AenviaEntrada foi recebida anteriormente de A, guarde Y e *sid*, e envie uma mensagem (EntradaDeBrecebida, *sid*) para A e $\mathcal{S}$. De outra forma, calcule os autovalores λ da matriz $X + Y$ e selecione λ_i, o i-ésimo autovalor segundo alguma ordem pré-estabelecida. Compute $z = \vec{v}$ tal que $(X + Y)\vec{v} = \lambda_i \vec{v}$ e envie uma mensagem (BrecebeSaída, z, *sid*) para B e (BrecebeSaída, *sid*) para $\mathcal{S}$.

Como citado nas seções anteriores, não se mostra na funcionalidade ideal do TI os

dados necessários para executar os protocolos 6.2 e 6.4. Isso é feito, sem perda de generalidade, desde que os protocolos citados foram provados UC seguros. Além disso, a funcionalidade ideal para o TI se mantém extremamente similar aos demais, de fato a funcionalidade é a mesma $\mathcal{F}_{TI}$ da subseção 6.2.2 .

6.4.3 Prova de Segurança Universalmente Composta

Nessa seção, será construído o adversário ideal $\mathcal{S}$ e demonstra-se a indistinguibilidade da visão do ambiente para a execução do protocolo no modelo real e no modelo ideal. O adversário ideal considerado $\mathcal{S}$ é um simulador *shell* que executa dentro dele uma cópia A' do adversário real $\mathcal{A}$ e um simulador caixa-preta $\hat{\mathcal{S}}$. De fato, o que será construído é o simulador caixa-preta $\hat{\mathcal{S}}$ descrevendo as ações que $\hat{\mathcal{S}}$ toma para os casos em que apenas Bob é honesto, apenas Alice é honesta, ambos Alice e Bob são corruptos e ambos Alice e Bob são honestos.

Ressalta-se que toda entrada que $\mathcal{S}$ recebe de $\mathcal{Z}$ é escrito na fita de entrada do adversário A', sem alteração, e todo valor de saída de A' é copiado por $\mathcal{S}$, possivelmente modificado, e enviado para a fita de saída de $\mathcal{S}$ para ser lido pelo ambiente.

6.4.3.1 Alice Corrupta e Bob Honesto

Segue as ações de $\hat{\mathcal{S}}$ em cada interação com A' e com $\mathcal{F}_{autovetor(X,Y)}$.

1. **Na fase de inicialização:**
   - $\hat{\mathcal{S}}$ escolhe $Q' \in_R SL(\mathbb{F}_q)$, $R' \in_R \mathbb{F}_q^{n\times n}$, $U' \in \mathbb{F}_q^{n\times n}$, calcula $V' = R'Q' + U'$, e envia $\mu'_A := (R', U')$ para A';
2. **Ao receber a entrada X de A':**
   - $\hat{\mathcal{S}}$ faz $X' = X$, escolhe $Y' \in_R \mathbb{F}_q^{n\times n}$, $i' \in_R \{1, ..., n\}$, $P'_0 \in_R SL(\mathbb{F}_q)$;
3. **Ao receber mensagem** EntradaDeBrecebida **de** $\mathcal{F}_{autovetor(X,Y)}$:
   - $\hat{\mathcal{S}}$ calcula $\vec{lambda'}$ através das raízes do polinômio característico associado a $\det(X' + Y' - \lambda' I)$. Ele toma o i-ésimo elemento de $\vec{lambda'}$, λ'_i, calcula $T' = Y' - \lambda'_i I$, $M' = (P'X' - R')Q' + P'T'Q' - U'$ e envia $\mu'_1 := (M')$ para A';

4. **Ao receber mensagem** $\mu_2' := (\vec{s})$ **de** A':

   - $\hat{\mathcal{S}}$ verifica se $\vec{s}$ é consistente com $X', Y', P', R', Q', V', i'$ e $\vec{\lambda'}$. Se esse for o caso, $\hat{\mathcal{S}}$ envia a mensagem (AenviaEntrada, X', $ssid$, sid) para $\mathcal{F}_{autovetor(X,Y)}$. Se esse não for o caso, $\hat{\mathcal{S}}$ escolhe um $X'' \notin \mathbb{F}_q^{n \times n}$ e envia a mensagem (AenviaEntrada, X'', $ssid$, sid) para $\mathcal{F}_{autovetor(X,Y)}$;

5. **Ao receber mensagem** EntradaInválida:

   - $\hat{\mathcal{S}}$ entrega essa mensagem para $\mathcal{Z}$ através da interface de A';

6. **Ao receber mensagem** EntradaDeArecebida **ou** BrecebeSaída **de** $\mathcal{F}_{autovetor(X,Y)}$:

   - $\hat{\mathcal{S}}$ ignora.

A análise de indistinguibilidade é igual à análise feita para Alice corrupta e Bob honesto em relação ao Protocolo 6.3.

6.4.3.2 Alice Honesta e Bob Corrupto

As interações de A' com $\hat{\mathcal{S}}$ são aquelas de Bob no protocolo real com $\mathcal{Z}$, TI e Alice. Segue as ações de $\hat{\mathcal{S}}$.

1. **Na fase de inicialização:**

   - $\hat{\mathcal{S}}$ escolhe $P' \in_R SL(\mathbb{F}_q)$, $Q' \in_R SL(\mathbb{F}_q)$, $U' \in \mathbb{F}_q^{n \times n}$ e envia $\mu_B' := (P', Q', U')$ para A';

2. **Ao receber a entrada** (Y, i) **de** A'**:**

   - $\hat{\mathcal{S}}$ faz $Y' = Y$ e $i' = i$, escolhe $X' \in_R \mathbb{F}_q^{n \times n}$, $R' \in_R \mathbb{F}_q^{n \times n}$, calcula $V' = R'Q' + U'$;

3. **Ao receber mensagem** $\mu_1' = (M)$ **de** A'**:**

   - $\hat{\mathcal{S}}$ calcula $\vec{lambda'}$ através das raízes do polinômio característico associado a $\det(X' + Y' - \lambda' I)$ e verifica se M é consistente com $X', Y', P', R', Q', V', i'$ e $\vec{\lambda'}$. Se esse for o caso, $\hat{\mathcal{S}}$ envia a mensagem (BenviaEntrada, (Y', i'),

$ssid, sid$) para $\mathcal{F}_{autovetor(X,Y)}$. Se esse não for o caso, $\hat{\mathcal{S}}$ escolhe um $Y'' \notin \mathbb{F}_q^{n\times n}$, faz $i'' = 0$ e envia a mensagem (BenviaEntrada, (Y'', i''), $ssid, sid$) para $\mathcal{F}_{autovetor(X,Y)}$;

4. **Ao receber mensagem** (BrecebeSaída, z, $ssid, sid$) **de** $\mathcal{F}_{autovetor(X,Y)}$:
   - $\hat{\mathcal{S}}$ guarda z, calcula N', $\vec{s}$e envia $\mu_2' := (\vec{s'})$ para A';
   - $\hat{\mathcal{S}}$ intercepta o valor de z' calculado por A' e verifica se z' é consistente com X', Y', λ_i'. Se esse for o caso, $\hat{\mathcal{S}}$ substitui z' pelo valor z recebido por $\mathcal{F}_{autovetor(X,Y)}$. Senão, não será feita a substituição.
   - $\hat{\mathcal{S}}$ entrega a saída z' para o ambiente $\mathcal{Z}$ através da interface de A';
5. **Ao receber mensagem** EntradaInválida:
   - $\hat{\mathcal{S}}$ entrega essa mensagem para o ambiente $\mathcal{Z}$ através da interface de A';
6. **Ao receber mensagem**EntradaDeBrecebida **de** $\mathcal{F}_{autovetor(X,Y)}$:
   - $\hat{\mathcal{S}}$ ignora.

A análise de indistinguibilidade é igual à análise feita para Alice honesta e Bob corrupto em relação ao Protocolo 6.4. Basta que se leia μ_2' no lugar de μ_1' e $\mathcal{F}_{autovetor(X,Y)}$ no lugar de $\mathcal{F}_{autovalor(x,y)}$.

6.4.3.3 Alice e Bob Corruptos ou Alice e Bob Honestos

A análise é exatamente igual aquela feita para os protocolos anteriores.

Dado que para todo ambiente a sua visão é indistinguível em todos os quatro casos possíveis (Alice corrupta, Bob honesto; Alice honesta, Bob corrupto; Alice e Bob corruptos; Alice e Bob honestos), tem-se que o Protocolo 6.5 UC-realiza $\mathcal{F}_{autovetor(X,Y)}$.

7 CONCLUSÕES E RECOMENDAÇÕES

Neste trabalho foi apresentado uma série de protocolos criptográficos distribuídos com segurança universalmente composta. Pode-se afirmar que todos os protocolos apresentados são incondicionalmente seguros, pois a segurança alcançada é independente do poder computacional do adversário.

O primeiro protocolo apresentado foi o protocolo para *String Oblivious Transfer* baseado em um Canal com Apagamentos Generalizado. Provou-se que esse protocolo atinge a capacidade de *oblivious transfer* para o canal com apagamentos generalizado quando se considera adversários maliciosos. Como a capacidade de *oblivious transfer* mede quão eficiente é usado um recurso ruidoso, conclui-se que o protocolo é implementado utilizando o canal ruidoso da forma mais eficientemente possível, mesmo na presença de adversários que se comportam de forma arbitrária. Outra vantagem do protocolo para *String Oblivious Transfer* está no fato de ele possuir rodadas constantes. Isso significa que o número de mensagens trocadas para executar o protocolo não depende do tamanho da entrada que se quer transferir. De fato, todos os protocolos apresentados neste trabalho possuem rodadas constantes. O fato do protocolo de *String Oblivious Transfer* possui essa característica é consequência da escolha do protocolo com rodadas constantes que calcula o *Hashing* Interativo. Além das vantagens citadas, outra vantagem desse protocolo é que ele foi provado ser seguro no *framework* UC. A consequência imediata é que o protocolo permanece seguro quando executado em um ambiente multi-parte desconhecido.

Apesar das imensas vantagens oferecidas pelo protocolo para *String Oblivious Transfer*, o protocolo não é eficiente no ponto de vista computacional, uma vez que Bob usa a saída de uma função de *hash* para corrigir os erros da *string* recebida pelo Canal com Apagamentos Generalizado. Então, determinar um protocolo computacionalmente que atinge a capacidade de *oblvious transfer* no Canal com Apagamentos Generalizado é deixado como um problema aberto.

Outros três protocolos foram apresentados no modelo criptográfico baseado em *commodities*, isto é, esses protocolos utilizam-se de uma terceira parte confiável e incorruptível, chamada de inicializador confiável. Até onde se sabe esses são os primeiros

protocolos para calcular determinante, autovalor e autovetor baseado em um inicializador confiável. Como a noção de autovetor se relaciona com a noção de autovalor, que se relaciona com a noção de determinante, pode-se utilizar a propriedade de modularidade da segurança universalmente composta. Isto é, dado que o protocolo de determinante foi provado ser UC seguro, ele pode ser utilizado como subprotocolo dentro do protocolo que calcula autovalor e assim por diante. Então, além da construção ser modular na prova UC do protocolo que calcula o autovalor, por exemplo, a análise pode ignorar a construção interna do protocolo de determinante. Portanto, a análise de segurança foi bastante simplificada.

Além da análise de segurança ser simplificada, a construção em si de cada um dos protocolos de álgebra linear também foi bastante simplificada pela presença do inicializador confiável. Com os dados fornecidos pelo inicializador confiável no início do protocolo, as partes puderam "esconder"as entradas privadas durante a troca de mensagens. Dessa forma, eles puderam calcular a função desejada de forma direta sem precisar executar um protocolo de *verifiable secret sharing*, por exemplo. Além disso, o uso do inicializador confiável permitiu que os protocolos gerados possuíssem rodadas constantes. Como discutido anteriormente, protocolos com rodadas constantes permitem que o número de mensagens trocadas pelas partes seja independente do tamanho das entradas no protocolo.

A principal desvantagem desses protocolos de álgebra linear é que o determinante é calculado através do polinômio característico e se n for muito grande, o cálculo se torna bastante custoso na prática. Uma sugestão é usar funções de aproximações para calcular por exemplo os autovetores associados a um determinado autovalor. O cálculo de determinantes, autovalores e autovetores computacionalmente eficientes no modelo criptográfico baseado em *commodities* é deixado como um problema aberto.

REFERÊNCIAS BIBLIOGRÁFICAS

AHLSWEDE, R.; CSISZAR, I. On oblivious transfer capacity. *IEEE International Symposium on Information Theory*, p. 2061–2064, 2007.

BEAVER, D. Precomputing oblivious transfer. *CRYPTO' 95*, p. 97–109, 1995.

BEAVER, D. Commodity-based cryptography (extended abstract). *STOC'97*, p. 446–455, 1997.

BENNETT, C. H. et al. Generalized privacy amplification. *IEEE Transactions on Information Theory*, p. 1915–1923, 1995.

BENNETT, C. H.; BRASSARD, G.; ROBERT, J. Privacy amplification by public discussion. *SIAM J. Comput.*, p. 210–229, 1988.

BLUNDO, C. et al. Constructions and bonds for unconditionally secure non-interactive commitment schemes. *Des. Codes Cryptography*, 2002.

BRASSARD, G.; CREPEAU, C.; ROBERT, J. Information theoretic reductions among disclosure problems. *FOCS '86*, p. 168–173, 1986.

CACHIN, C.; CREPEAU, C.; MARCIL, J. Oblivious transfer with a memory-bounded receiver. *FOCS '98*, p. 493 – 502, 1998.

CANETTI, R. Universally composable security: A new paradigm for criptography protocols. *Cryptography ePrint Archive, Report 2000/067, revised Jan2005 and Dec 2005.*, 2000.

CANETTI, R. Universally composable security: A new paradigm for cryptographic protocols. *Extended Abstract appeared in: 42nd Symposium on Foundations*, 2001.

CANETTI, R. et al. Universally composable two-party and multi-party secure computation. *STOC'02*, p. 494–503, 2002.

CANETTI, R.; RABIN, T. Universal composition with joint state. *CRYPTO '03*, 2003.

CARTER, J. L.; WEGMAN, M. N. Universal classes of hash functions. *Journal of Computer and System Sciences*, p. 143 – 154, 1979.

CHERNOFF, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statistics*, p. 493–507, 1952.

COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. [S.l.]: Wiley-Interscience, 2006.

CRÉPEAU, C. Equivalence between two flavours of oblivious transfers. *CRYPTO '87*, v. 293, p. 350 – 354, 1987.

CRÉPEAU, C.; KILIAN, J. Achieving oblivious transfer using weakened security assumptions (extended abstract). *FOCS '88*, p. 42–52, 1988.

CRÉPEAU, C.; SAVVIDES, G. G. Optimal reductions between oblivious transfers using interactive hashing. *EUROCRYPT 2006*, p. 201–221, 2006.

CRÉPEAU, C.; WULLSCHLEGER, J. Statistical security conditions for two-party secure function evaluation. *ICITS 2008*, 2008.

CSISZáR, I.; KÖRNER, J. *Information Theory: Coding Theorems for Discrete Memoryless Channels*. [S.l.]: New York: Academic, 1981.

DING, Y. Z. et al. Constant-round oblivious transfer in the bounded storage model. *J. Cryptology and Conference version appeared at TCC '04*, p. 165–202, 2007.

DODIS, Y. et al. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput. Conference version appeared in EUROCRYPT 2004.*, p. 97–139, 2008.

DOWSLEY, R. et al. A two-party protocol with trusted initializer for computing the inner product. *WISA'10 Proceedings of the 11th international conference on Information security applications*, p. 337–350, 2010.

EVEN, S.; GOLDREICH, O.; LEMPEL, A. A randomized protocol for signing contracts. *Communications of the ACM*, p. 637 – 647, 1985.

GOLDREICH, O. *Foundations of Cryptography. Volume I: Basic Tools*. [S.l.]: Cambridge University Press, 2000.

GOLDREICH, O. *Foundations of Cryptography. Volume II: Basic Applications*. [S.l.]: Cambridge University Press, 2000.

GOLDREICH, O.; MICALI, S.; WIGDERSON, A. How to play any mental game or a completeness theorem for protocols with honest majority. *ACM STOC '87*, p. 218–229, 1987.

GOLDWASSER, S.; MICALI, S.; RACKOFF, C. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, p. 186–208, 1989.

HASTAD, J. et al. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, p. 1364 – 1396, 1999.

IMAI, H.; MOROZOV, K.; NASCIMENTO, A. C. A. On the oblivious transfer capacity of the erasure channel. *2006 IEEE International Symposium on Information Theory*, p. 1428 – 1431, 2006.

IMAI, H.; NASCIMENTO, A. C. A.; WINTER, A. Commitment capacity of discrete memoryless channels. *IMA Int. Conf. 2003*, p. 35–51, 2003.

IMPAGLIAZZO, R.; LEVIN, L. A.; LUBY, M. Pseudo-random generation from one-way functions (extended abstracts). *STOC '89*, p. 12–24, 1989.

KILIAN, J. Founding cryptography on oblivious transfer. *STOC 1988*, p. 20–31, 1988.

MAURER, U. M. Secret key agreement by public discussion. *IEEE Transactions on Information Theory*, p. 733 – 742, 1993.

NAOR, M. et al. Perfect zero-knowledge arguments for np using any one-way permutation. *Journal of Cryptology. Preliminary version in CRYPTO '92*, p. 87–108, 1998.

NASCIMENTO, A. C. et al. Unconditionally non-interactive verifiable secret sharing secure against faulty majorities in the commodity baseed model. *ACNS '04*, p. 355–368, 2004.

NASCIMENTO, A. C. A.; WINTER, A. On the oblivious-transfer capacity of noisy resources. *IEEE Transactions on Information Theory*, p. 2572–2581, 2008.

NISAN, N.; ZUCKERMAN, D. Randomness is linear in space. *Journal of Computer and System Sciences*, p. 43 – 53, 1996.

PINTO, A. C. B. et al. Achieving oblivious transfer capacity of generalized erasure channels in the malicious model. *IEEE Transactions on Information Theory*, p. 556–5571, 2011.

RABIN, M. O. How to exchange secrets by oblivious transfer. *Technical Report TR-81, Aiken Computation Laboratory, Harvard University*, 1981.

RABIN, T.; BEN-OR, M. Verifiable secret sharing and multiparty protocols with honest majority. *ACM STOC '89*, p. 73–85, 1989.

RADHAKRISHNAN, J.; TA-SHMA, A. Bounds for dispersers, extractors, and depthtwosuperconcentrators. *SIAM Journal on Discrete Mathematics*, p. 2 – 24, 2000.

RIVEST, R. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. *http://people.csail.mit.edu/rivest/Rivest-commitment.pdf*, 1999.

SAVVIDES, G. *Interactive Hashing and reductions between Oblivious Transfer variants*. Tese (Doutorado) — School of Computer Science, McGill University, 2007.

SHALTIEL, R. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, p. 67 – 95, 2002.

WIESNER, S. Conjugate coding. *SIGACT News*, ACM Press, v. 15, n. 1, p. 78–88, 1983.

WYNER, A. D. The wire-tap channel. *Bell Syst. Tech. J*, v. 54, p. 1355 – 1387, 1975.

YAO, A. C. Protocols for secure computation. *FOCS '82*, p. 160 – 164, 1982.