SAPIENZA
UNIVERSITÀ DI ROMA

Matchmaking Encryption

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Engineering in Computer Science

Candidate

Luigi Russo
ID number 1699981

Thesis Advisors

Prof. Riccardo Lazzeretti
Prof. Daniele Venturi

Academic Year 2019/2020

Thesis not yet defended

Matchmaking Encryption
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: russo.1699981@studenti.uniroma1.it

Contents

Chapter 1

Introduction

Matchmaking Encryption is a new form of encryption introduced by Ateniese et al. [1] as a generalization of Attribute Based Encryption (ABE) [2] which enables several new applications where involved participants can specify fine-grained access policies to encrypted data, opening up new ways for secret communications. ME can be extremely useful in oder to avoid liability or inappropriateness; indeed, some of the killer applications of ME are: social matchmaking, encrypting bids and votes, censorship-resistant communication for marginalized communities in authoritarian countries.

missing ABE formal definition.

In particular, in this context we consider two parties: the sender S with attributes σ and the receiver R with attributes ρ. They can both specify policies (i.e. access rights expressed in terms of attributes) the other party must satisfy in order for the message to be revealed. The policies are represented as circuits $\mathbb{C} : \{0,1\}^* \to \{0,1\}$, while the attributes are binary strings of arbitrary length. For instance, in the social matchmaking setting, S can encrypt a message m containing his personal details and specify a policy $\mathbb{R}$ so that the message can be decrypted only by his ideal partner, i.e. one who matches the policy. On the other end, R will be able to decrypt the message only if S corresponds to his ideal partner: this is again expressed through a policy $\mathbb{S}$. The decryption is successful if $\mathbb{S}(\sigma) = \mathbb{R}(\rho) = 1$, i.e. both the policies are matched.

Security. Security is defined via two properties: privacy and authenticity. Intuitively, privacy aims at capturing secrecy of the sender's inputs in presence of malicious receivers: during the decryption nothing is leaked beyond the fact that a match occurred or did not occur; in particular, in case of mismatch, the reason why the match did not occur is not revealed. This is a much stronger guarantee w.r.t the one addressed by dual policy ABE schemes,[1] introduced by Attrapadung and Imai [3], which only protects the secrecy of the message. In order to ensure such a guarantee, any ME construction needs to face the technical challenge of simultaneously checking the policies chosen by S and R: this check can be seen as an atomic

[1] Here the sender encrypts a message by choosing both a policy and a set of attributes; the receiver can decrypt the ciphertext using a decryption key associated to both the receiver's policy and attributes.

operation.
Authenticity, on the contrary, captures security against malicious senders: more formally, it demands that the only way to produce a valid ciphertext under attributes σ is to obtain an encryption key corresponding to σ from the authority, thus guaranteeing that if a ciphertext decrypts correctly, then it has been created by a sender with the proper encryption key.

A-ME. Ateniese et al. also defined a possible variant of ME, namely Arranged Matchmaking Encryption (A-ME). A-ME implies CP-ABE and KP-ABE with the very same expressiveness, but makes a different use of decryption keys with respect to ME: indeed A-ME keys are associated with both attributes and policies, while ME makes use of two different keys, one for the policy and the other for the attributes.

1.1 Thesis contributions

CCA security. In this document we revamp the notion of ME, by extending the security definitions to the setting of chosen ciphertext attack (CCA). In this context ciphertexts cannot be malleable[2] anymore. We define these two new properties:

- CCA-privacy: the adversary may now additionally have access to a decryption oracle which answers decryption queries. Intuitively it captures the possibility to generate ciphertexts which correctly decrypt to f(m), given an encryption of m, for some function f.
- CCA-authenticity: the attacker is also given access to an encryption oracle

This has been done for both ME and A-ME schemes, in order to capture stronger security guarantees w.r.t. the chosen plaintext attack (CPA) scenario, addressed in the original paper. Indeed, CCA-privacy is of seminal importance in many practical situations: e.g. in 1998 Bleichenbacher showed how to crack a SSL session key in less than a day, simply sending several test ciphertexts to a decryption device [4].

Constructions. We finally define black-box constructions to transform any CPA-secure ME (resp. A-ME) scheme into a CCA-private ME (resp. A-ME) scheme, by relying on a Non-Interactive Zero Knowledge (NIZK) argument system and a Signatures scheme.
In our construction, to encrypt message m under sender attributes σ and policy $\mathbb{R}$ we proceed as follows: we first encrypt m using the underlying ME (resp. A-ME) scheme; then we add a NIZK argument of the knowledge of the plaintext m and a valid signature s on σ, where the signature is provided by a trusted party.
We prove that these two constructions achieve CCA-privacy and preserve the authenticity of the underlying schemes.

[2] If malleable, given an encryption of a message m, it is possible to generate another ciphertext which decrypts to f(m), for some function f, without necessarily knowing or learning m.

Chapter 2

Preliminaries

2.1 Notation

We use the notation $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$. Capital boldface letters (such as $\mathbf{X}$) are used to denote random variables, small letters (such as x) to denote concrete values, calligraphic letters (such as $\mathcal{X}$) to denote sets, and serif letters (such as A) to denote algorithms. All of our algorithms are modeled as (possibly interactive) Turing machines; if algorithm $\mathsf{A} = (\mathsf{A}_1, \ldots, \mathsf{A}_k)$ has oracle access to some oracle O, we often implicitly write $\mathcal{Q}_{\mathsf{O}}$ for the set of queries asked by A to O and $\mathcal{Q}^i_{\mathsf{O}}$ for the set of queries asked by A_i to O. Furthermore, we denote by $\mathcal{O}_{\mathsf{O}}$ (resp. $\mathcal{O}^i_{\mathsf{O}}$) the set of outputs returned to A (resp. A^i) by O.

For a string $x \in \{0,1\}^*$, we let $|x|$ be its length; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the cardinality of $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow\!\!\$\, \mathcal{X}$. If A is an algorithm, we write $y \leftarrow\!\!\$\, \mathsf{A}(x)$ to denote a run of A on input x and output y; if A is randomized, y is a random variable and $\mathsf{A}(x; r)$ denotes a run of A on input x and (uniform) randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathsf{A}(x; r)$ terminates in a polynomial number of steps (in the input size).

Negligible functions. Throughout the document, we denote by $\lambda \in \mathbb{N}$ the security parameter and we implicitly assume that every algorithm takes as input the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is called *negligible* in the security parameter λ if it vanishes faster than the inverse of any polynomial in λ, i.e. $\nu(\lambda) \in O(1/p(\lambda))$ for all positive polynomials $p(\lambda)$. We sometimes write $\mathsf{negl}(\lambda)$ (resp., $\mathsf{poly}(\lambda)$) to denote an unspecified negligible function (resp., polynomial function) in the security parameter.

Indistinguishability. We say that $\mathbf{X}$ and $\mathbf{Y}$ are *computationally* indistinguishable, denoted $\mathbf{X} \approx_c \mathbf{Y}$, if for all PPT distinguishers D we have $\Delta_{\mathsf{D}}(X_\lambda; Y_\lambda) \in \mathsf{negl}(\lambda)$, where

$$\Delta_{\mathsf{D}}(X_\lambda; Y_\lambda) \stackrel{\text{def}}{=} \left| \mathbb{P}\left[\mathsf{D}(1^\lambda, X_\lambda) = 1\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda, \mathbf{Y}_\lambda) = 1\right] \right|.$$

2.2 Signature Schemes

A signature scheme is made of the following polynomial-time algorithms.

$\mathsf{KGen}(1^\lambda)$**:** The randomized key generation algorithm takes the security parameter and outputs a secret and a public key $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$**:** The randomized signing algorithm takes as input the secret key sk and a message $m \in \mathcal{M}$, and produces a signature s.

$\mathsf{Ver}(\mathsf{pk}, m, s)$**:** The deterministic verification algorithm takes as input the public key pk, a message m, and a signature s, and it returns a decision bit.

A signature scheme should satisfy two properties. The first property says that honestly generated signatures always verify correctly. The second property, called unforgeability, says that it should be hard to forge a signature on a fresh message, even after seeing signatures on polynomially many messages.

Definition 1 (Correctness of signatures)**.** *A signature scheme* $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *with message space* $\mathcal{M}$ *is correct if* $\forall \lambda \in \mathbb{N}$*,* $\forall (\mathsf{sk}, \mathsf{pk})$ *output by* $\mathsf{KGen}(1^\lambda)$*, and* $\forall m \in \mathcal{M}$*, the following holds:*

$$\mathbb{P}[\mathsf{Ver}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1] = 1.$$

Definition 2 (Unforgeability of signatures)**.** *A signature scheme* $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *is existentially unforgeable under chosen-message attacks (EUF-CMA) if for all PPT adversaries* A*:*

$$\mathbb{P}\left[\mathbf{G}^{\mathsf{euf}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathbf{G}^{\mathsf{euf}}_{\Pi,\mathsf{A}}(\lambda)$ *is the following experiment:*

1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$.
2. $(m, s) \leftarrow\!\!\$\ \mathsf{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(1^\lambda, \mathsf{pk})$
3. *If* $m \notin \mathcal{Q}_{\mathsf{Sign}}$*, and* $\mathsf{Ver}(\mathsf{pk}, m, s) = 1$*, output* 1*, else output* 0.

2.3 Non-Interactive Zero Knowledge

Let R be a relation, corresponding to an NP language L. A non-interactive zero-knowledge (NIZK) proof system for R is a tuple of polynomial-time algorithms $\Pi = (\mathsf{I}, \mathsf{P}, \mathsf{V})$ specified as follows:

- The randomized algorithm I takes as input the security parameter and outputs a common reference string ω;
- The randomized algorithm $\mathsf{P}(\omega, (y, x))$, given $(y, x) \in R$ outputs a proof π;
- The deterministic algorithm $\mathsf{V}(\omega, (y, \pi))$, given an instance y and a proof π outputs either 0 (for "reject") or 1 (for "accept").

We say that a NIZK for relation R is *correct* if $\forall\lambda \in \mathbb{N}$, every ω output by $\mathsf{I}(1^\lambda)$, and any $(y, x) \in R$, we have that $\mathsf{V}(\omega, (y, \mathsf{P}(\omega, (y, x)))) = 1$.

We define two properties of a NIZK proof system. The first property, called adaptive multi-theorem zero knowledge, says that honest proofs do not reveal anything beyond the fact that $y \in L$. The second property, called knowledge soundness, requires that every adversary creating a valid proof for some statement, must know the corresponding witness.

Definition 3 (Adaptive multi-theorem zero-knowledge)**.** *A NIZK Π for a relation R satisfies adaptive multi-theorem zero-knowledge if there exists a PPT simulator $\mathsf{Z} := (\mathsf{Z}_0, \mathsf{Z}_1)$ such that the following holds:*

- *Algorithm Z_0 outputs ω and a simulation trapdoor ζ.*
- *For all PPT distinguishers D, we have that*

$$\Big|\mathbb{P}\Big[\mathsf{D}^{\mathsf{P}(\omega,(\cdot,\cdot))}(\omega) = 1 :\ \omega \leftarrow\!\!\$\ \mathsf{I}(1^\lambda)\Big] - \mathbb{P}\Big[\mathsf{D}^{\mathsf{O}(\zeta,(\cdot,\cdot))}(\omega) = 1 :\ (\omega, \zeta) \leftarrow\!\!\$\ \mathsf{Z}_0(1^\lambda)\Big]\Big| \leq \mathsf{negl}(\lambda)\,,$$

  where the oracle $\mathsf{O}(\zeta, \cdot, \cdot)$ takes as input a pair (y, x) and returns $\mathsf{Z}_1(\zeta, y)$ if $(y, x) \in R$ (and otherwise $\perp$).

Definition 4 (True-simulation f-extractability)**.** *Let f be a fixed efficiently computable function. A NIZK Π for a relation R satisfies true-simulation f-extractability (f-tSE) if there exists a PPT extractor $\mathsf{K} = (\mathsf{K}_0, \mathsf{K}_1)$ such that the following holds:*

- *Algorithm K_0 outputs ω, a simulation trapdoor ζ and an extraction trapdoor ξ, such that the distribution of (ω, ζ) is computationally indistinguishable to that of $\mathsf{Z}_0(1^\lambda)$.*
- *For all PPT adversaries A, we have that*

$$\mathbb{P}\left[\begin{array}{cc} \mathsf{V}(\omega, y, \pi) = 1 \wedge & (\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda) \\ (y, \pi) \notin \mathcal{O}_{\mathsf{O}} \wedge & : (y, \pi) \leftarrow\!\!\$\ \mathsf{A}^{\mathsf{O}(\zeta,(\cdot,\cdot))}(\omega) \\ \forall x\ s.t.\ f(x) = z, (y, x) \notin R & z \leftarrow\!\!\$\ \mathsf{K}_1(\xi, y, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)\,,$$

where the oracle $\mathsf{O}(\zeta, \cdot, \cdot)$ takes as input a pair (y, x) and returns $\mathsf{Z}_1(\zeta, y)$ if $(y, x) \in R$ (and otherwise $\perp$).

In the case when f is the identity function, we simply say that Π is true-simulation extractable (tSE).

Chapter 3

Matchmaking Encryption

3.1 The General Setting

Formally, an ME is composed of the following polynomial-time algorithms:

$\mathsf{Setup}(1^\lambda)$**:** Upon input the security parameter 1^λ the randomized setup algorithm outputs the master public key mpk, the master policy key kpol, and the master secret key msk. We implicitly assume that all other algorithms take mpk as input.

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$**:** The randomized sender-key generator takes as input the master secret key msk, and attributes $\sigma \in \{0,1\}^*$. The algorithm outputs a secret encryption key ek_σ for attributes σ.

$\mathsf{RKGen}(\mathsf{msk}, \rho)$**:** The randomized receiver-key generator takes as input the master secret key msk, and attributes $\rho \in \{0,1\}^*$. The algorithm outputs a secret decryption key dk_ρ for attributes ρ.

$\mathsf{PolGen}(\mathsf{kpol}, \mathbb{S})$**:** The randomized receiver policy generator takes as input the master policy key kpol, and a policy $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ represented as a circuit. The algorithm outputs a secret decryption key $\mathsf{dk}_\mathbb{S}$ for the circuit $\mathbb{S}$.

$\mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)$**:** The randomized encryption algorithm takes as input a secret encryption key ek_σ for attributes $\sigma \in \{0,1\}^*$, a policy $\mathbb{R} : \{0,1\}^* \to \{0,1\}$ represented as a circuit, and a message $m \in \mathcal{M}$. The algorithm produces a ciphertext c linked to both σ and $\mathbb{R}$.

$\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$**:** The deterministic decryption algorithm takes as input a secret decryption key dk_ρ for attributes $\rho \in \{0,1\}^*$, a secret decryption key $\mathsf{dk}_\mathbb{S}$ for a circuit $\mathbb{S} : \{0,1\}^* \to \{0,1\}$, and a ciphertext c. The algorithm outputs either a message m or $\perp$ (denoting an error).

Correctness. Correctness intuitively says that decrypting an honestly generated ciphertext which encrypts a message m using sender's attributes σ and policy $\mathbb{R}$ using decryption keys for receiver's attributes ρ and access policy $\mathbb{S}$ should equal m if and only if the receiver's attributes ρ match the policy $\mathbb{R}$ specified by the sender,

and at the same time the sender's attributes σ match the policy $\mathbb{S}$ specified by the receiver. On the other hand, in case of mismatch, the decryption algorithm returns $\perp$. More formally:

Definition 5 (Correctness of ME)**.** *An ME with message space $\mathcal{M}$ is correct if* $\forall\lambda \in \mathbb{N}$, $\forall(\mathsf{mpk}, \mathsf{kpol}, \mathsf{msk})$ *output by* $\mathsf{Setup}(1^\lambda)$, $\forall m \in \mathcal{M}$, $\forall\sigma, \rho \in \{0,1\}^*$, $\forall\mathbb{R}, \mathbb{S} : \{0,1\}^* \to \{0,1\}$*:*

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, \mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)) = m] \geq 1 - \mathsf{negl}(\lambda)\,,$$

whenever $\mathbb{S}(\sigma) = 1$ *and* $\mathbb{R}(\rho) = 1$*, and otherwise*

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, \mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)) = \perp] \geq 1 - \mathsf{negl}(\lambda)\,,$$

where $\mathsf{ek}_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}(\mathsf{msk}, \sigma)$, $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{RKGen}(\mathsf{msk}, \rho)$, $\mathsf{dk}_\mathbb{S} \leftarrow\!\!\$\ \mathsf{PolGen}(\mathsf{kpol}, \mathbb{S})$.

3.2 The Arranged Setting

The syntax of an A-ME is similar to that of an ME, except that decryption keys are associated with both attributes and policies. In particular, the following efficient algorithms make an A-ME:

$\mathsf{SKGen}, \mathsf{Enc}$**:** Identical to the ones in an ME (cf. §**??**).

Setup**:** Upon input the security parameter 1^λ, the randomized setup algorithm outputs the master public key mpk and the master secret key msk.

$\mathsf{RKGen}(\mathsf{msk}, \rho, \mathbb{S})$**:** The randomized receiver key generator takes as input the master public key mpk, the master secret key msk, attributes $\rho \in \{0,1\}^*$, and a policy $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ represented as a circuit. The algorithm outputs a secret decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$.

$\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}}, c)$ The deterministic decryption algorithm takes a decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$, and a ciphertext c. The algorithm outputs either a message m or $\perp$ (denoting an error).

The definitions below capture the very same correctness requirement of an ME, but translated to the arranged case.

Definition 6 (Correctness of A-ME)**.** *An A-ME with message space $\mathcal{M}$ is correct if* $\forall\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk})$ *output by* $\mathsf{Setup}(1^\lambda)$, $\forall m \in \mathcal{M}$, $\forall\sigma, \rho \in \{0,1\}^*$, $\forall\mathbb{R}, \mathbb{S} : \{0,1\}^* \to \{0,1\}$*:*

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}}, \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathbb{R}, m)) = m] \geq 1 - \mathsf{negl}(\lambda)\,,$$

whenever $\sigma \in \mathbb{S}$ *and* $\rho \in \mathbb{R}$*, and otherwise*

$$\mathbb{P}[\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}}, \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathbb{R}, m)) = \perp] \geq 1 - \mathsf{negl}(\lambda)\,,$$

where ek_σ *and* $\mathsf{dk}_{\rho,\mathbb{S}}$ *are generated by* $\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma)$ *and* $\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho, \mathbb{S})$.

Chapter 4

Chosen Ciphertext Security

4.1 Privacy

Definition 7 (Privacy of ME)**.** *We say that an ME* Π *satisfies* privacy *if for all valid PPT adversaries* A*:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)=1\right]-\frac{1}{2}\right|\leq \mathsf{negl}(\lambda)\,,$$

where game $\mathbf{G}^{\mathsf{priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig.4.1. Adversary* A *is called valid if* $\forall(\rho_i,\mathbb{S}_i,c_i)\in \mathcal{Q}_{\mathsf{O}_4}, c_i\neq c$ *and* $\forall\rho\in\mathcal{Q}_{\mathsf{O}_2},\forall\mathbb{S}\in\mathcal{Q}_{\mathsf{O}_3}$ *it satisfies the following invariant:*

- ***(Mismatch condition).*** *Either*

$$\begin{aligned}&(\mathbb{R}_0(\rho)=\mathbb{R}_1(\rho)=0)\vee(\mathbb{S}(\sigma_0)=\mathbb{S}(\sigma_1)=0)\\&\quad\vee(\mathbb{R}_0(\rho)=\mathbb{S}(\sigma_1)=0)\vee(\mathbb{R}_1(\rho)=\mathbb{S}(\sigma_0)=0);\end{aligned}\tag{4.1}$$

- ***(Match condition).*** *Or (if* $\exists\hat{\rho}\in\mathcal{Q}_{\mathsf{O}_2},\hat{\mathbb{S}}\in\mathcal{Q}_{\mathsf{O}_3}$ *s.t. Eq.* (4.1) *does not hold)*

$$(m_0=m_1)\wedge(\mathbb{R}_0(\rho)=\mathbb{R}_1(\rho))\wedge(\mathbb{S}(\sigma_0)=\mathbb{S}(\sigma_1)).$$

$\mathbf{G}^{\mathsf{priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{mpk},\mathsf{kpol},\mathsf{msk})\leftarrow\$\ \mathsf{Setup}(1^\lambda)$
$(m_0,m_1,\mathbb{R}_0,\mathbb{R}_1,\sigma_0,\sigma_1,\alpha)\leftarrow\$\ \mathsf{A}_1^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3,\mathsf{O}_4}(1^\lambda,\mathsf{mpk})$
$b\leftarrow\$\ \{0,1\}$
$\mathsf{ek}_{\sigma_b}\leftarrow\$\ \mathsf{SKGen}(\mathsf{msk},\sigma_b)$
$c\leftarrow\$\ \mathsf{Enc}(\mathsf{ek}_{\sigma_b},\mathbb{R}_b,m_b)$
$b'\leftarrow\$\ \mathsf{A}_2^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3,\mathsf{O}_4}(1^\lambda,c,\alpha)$
If $(b'=b)$ **return** 1
Else **return** 0

$\mathbf{G}^{\mathsf{auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{mpk},\mathsf{kpol},\mathsf{msk})\leftarrow\$\ \mathsf{Setup}(1^\lambda)$
$(c,\rho,\mathbb{S})\leftarrow\$\ \mathsf{A}^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3,\mathsf{O}_5}(1^\lambda,\mathsf{mpk})$
$\mathsf{dk}_\rho\leftarrow\$\ \mathsf{RKGen}(\mathsf{msk},\rho)$
$\mathsf{dk}_{\mathbb{S}}\leftarrow\$\ \mathsf{PolGen}(\mathsf{kpol},\mathbb{S})$
$m=\mathsf{Dec}(\mathsf{dk}_\rho,\mathsf{dk}_{\mathbb{S}},c)$
If $(c\notin\mathcal{O}_{\mathsf{O}_5})\wedge\forall\sigma\in\mathcal{Q}_{\mathsf{O}_1}:(\mathbb{S}(\sigma)=0)\wedge(m\neq\perp)$
return 1
Else **return** 0

Figure 4.1. Games defining privacy and authenticity of ME. Oracles O_1, O_2, O_3 are implemented by $\mathsf{SKGen}(\mathsf{msk},\cdot)$, $\mathsf{RKGen}(\mathsf{msk},\cdot)$, $\mathsf{PolGen}(\mathsf{kpol},\cdot)$; O_4 takes in input a tuple $(\rho,\mathbb{S},c)$ and returns $\mathsf{Dec}(\mathsf{dk}_\rho,\mathsf{dk}_{\mathbb{S}},c)$; O_5 takes in input a tuple $(\sigma,\mathbb{R},m)$ and returns $\mathsf{Enc}(\mathsf{ek}_\sigma,\mathbb{R},m)$.

$\mathbf{G}^{\mathsf{arr\text{-}priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{mpk},\mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1, \alpha) \leftarrow\!\$\ \mathsf{A}_1^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3}(1^\lambda, \mathsf{mpk})$
$b \leftarrow\!\$\ \{0,1\}$
$\mathsf{ek}_{\sigma_b} \leftarrow\!\$\ \mathsf{SKGen}(\mathsf{mpk},\mathsf{msk},\sigma_b)$
$c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{mpk},\mathsf{ek}_{\sigma_b},\mathbb{R}_b,m_b)$
$b' \leftarrow\!\$\ \mathsf{A}_2^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3}(1^\lambda, c, \alpha)$
If $(b' = b)$ **return** 1
Else **return** 0

$\mathbf{G}^{\mathsf{arr\text{-}auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$

$(\mathsf{mpk},\mathsf{msk}) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$
$(c, \rho, \mathbb{S}) \leftarrow\!\$\ \mathsf{A}^{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_4}(1^\lambda, \mathsf{mpk})$
$\mathsf{dk}_{\rho,\mathbb{S}} \leftarrow\!\$\ \mathsf{RKGen}(\mathsf{mpk},\mathsf{msk},\rho,\mathbb{S})$
$m = \mathsf{Dec}(\mathsf{mpk},\mathsf{dk}_{\rho,\mathbb{S}},c)$
If $(c \notin \mathcal{O}_{\mathsf{O}_4(\cdot,\cdot,\cdot)}) \wedge \forall \sigma \in \mathcal{Q}_{\mathsf{O}_1} : (\mathbb{S}(\sigma) = 0) \wedge (m \neq \perp)$
return 1
Else **return** 0

Figure 4.2. Games defining privacy and authenticity of A-ME. Oracles O_1, O_2 are implemented by $\mathsf{SKGen}(\mathsf{msk},\cdot)$ and $\mathsf{RKGen}(\mathsf{msk},\cdot)$; O_3 takes in input a tuple $(\rho,\mathbb{S},c)$ and returns $\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}},c)$; O_4 takes in input a tuple $(\sigma,\mathbb{R},m)$ and returns $\mathsf{Enc}(\mathsf{ek}_\sigma,\mathbb{R},m)$.

Definition 8 (Privacy of A-ME)**.** *An A-ME* Π *satisfies* privacy *if for all valid PPT adversaries* A*:*

$$\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{arr\text{-}priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)\,,$$

where game $\mathbf{G}^{\mathsf{arr\text{-}priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig. 4.2. Adversary* A *is called valid if if* $\forall(\rho_i,\mathbb{S}_i,c_i) \in \mathcal{Q}_{\mathsf{O}_3}, c_i \neq c$ *and* $\forall(\rho,\mathbb{S}) \in \mathcal{Q}_{\mathsf{O}_2}$ *it satisfies the following invariant:*

- ***(Mismatch condition).*** *Either*

$$\begin{aligned}(\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho) = 0) \vee (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1) = 0)\\ \vee (\mathbb{R}_0(\rho) = \mathbb{S}(\sigma_1) = 0) \vee (\mathbb{R}_1(\rho) = \mathbb{S}(\sigma_0) = 0);\end{aligned} \tag{4.2}$$

- ***(Match condition).*** *Or (if* $\exists(\hat{\rho},\hat{\mathbb{S}}) \in \mathcal{Q}_{\mathsf{O}_2}$ *s.t. Eq.* (4.2) *does not hold)*

$$(m_0 = m_1) \wedge (\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho)) \wedge (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1)).$$

4.2 Authenticity

4.2.1 The General Setting

Definition 9 (Authenticity of ME)**.** *We say that an ME* Π *satisfies* authenticity *if for all PPT adversaries* A*:*

$$\mathbb{P}\left[\mathbf{G}^{\mathsf{auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where game $\mathbf{G}^{\mathsf{auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig.4.1.*

4.2.2 The Arranged Setting

Definition 10 (Authenticity of A-ME)**.** *We say that an A-ME* Π *satisfies* authenticity *if for all PPT adversaries* A*:*

$$\mathbb{P}\left[\mathbf{G}^{\mathsf{arr\text{-}auth}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where game $\mathbf{G}^{\mathsf{arr\text{-}auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig. 4.2.*

4.3 CPA to CCA Transformation

Let Π' be an ME scheme which satisfies privacy. If we use a signature scheme and a NIZK proof system, we can obtain a new ME scheme Π which preserves its privacy, but also achieves authenticity. The construction is as follows.

Construction 1. *Let* $\Pi' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{RKGen}', \mathsf{PolGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be an ME scheme,* $\mathsf{SS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme* SS *and* $\mathsf{NIZK} = (\mathsf{I}, \mathsf{P}, \mathsf{V})$ *be a* f*-tSE NIZK argument for the following NP relation:*

$$R \stackrel{def}{=} \left\{ ((\sigma, s, \mathbb{R}, m, r), (\mathsf{mpk}', \mathsf{pk}, c)) \ : \begin{array}{c} c = \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m; r) \wedge \\ \mathsf{Ver}(\mathsf{pk}, (\sigma, s)) = 1 \end{array} \right\}.$$

where $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, s, \mathbb{R}, m)$*, so* f *extracts all the input but the randomness* r*.*

Bind ek_σ *with some randomness* r_2

We construct the new ME scheme Π *as follows:*

$\mathsf{Setup}(1^\lambda)$**:** *on input the security parameter* 1^λ*, the algorithm computes* $(\mathsf{msk}', \mathsf{kpol}, \mathsf{mpk}') \leftarrow\!\!\$\ \mathsf{Setup}'(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ *and* $\omega \leftarrow\!\!\$\ \mathsf{I}(1^\lambda)$*. The algorithm outputs* $\mathsf{msk} = (\mathsf{msk}', \mathsf{kpol}, \mathsf{sk})$ *as the master secret key and* $\mathsf{mpk} = (\mathsf{mpk}', \mathsf{pk}, \omega)$ *as the master public key. All other algorithms are implicitly given* mpk *as additional input.*

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$**:** *the randomized sender-key generator takes as input the master secret key* $\mathsf{msk} = (\mathsf{msk}', \mathsf{sk})$ *and attributes* $\sigma \in \{0,1\}^*$*. The algorithm returns the encryption key* $\mathsf{ek}_\sigma = (\mathsf{ek}'_\sigma, s)$*, where* $\mathsf{ek}'_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}'(\mathsf{msk}', \sigma)$ *and* $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho)$**:** *the randomized receiver-key generator takes as input the master secret key* $\mathsf{msk} = (\mathsf{msk}', \mathsf{sk})$ *and attributes* $\rho \in \{0,1\}^*$*. The algorithm computes the key* $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{RKGen}'(\mathsf{msk}', \rho)$*.*

$\mathsf{PolGen}(\mathsf{kpol}, \mathbb{S})$**:** *the receiver policy generator takes as input the master policy key* kpol *and a policy* $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ *and simply outputs* $\mathsf{dk}_\mathbb{S} \leftarrow\!\!\$\ \mathsf{PolGen}'(\mathsf{kpol}, \mathbb{S})$*.*

$\mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)$**:** *the randomized encryption algorithm takes as input a secret encryption key* $ek_\sigma = (\mathsf{ek}'_\sigma, s)$ *for attributes* $\sigma \in \{0,1\}^*$*, a policy* $\mathbb{R} : \{0,1\}^* \to \{0,1\}$ *and a message* $m \in \{0,1\}^*$*. The algorithm first encrypts the message by computing* $c' \leftarrow\!\!\$\ \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m)$*; then, it returns the ciphertext* $c = (c', \pi)$*, where* $\pi \leftarrow\!\!\$\ \mathsf{P}(\omega, (\sigma, s, \mathbb{R}, m, r), (\mathsf{mpk}', \mathsf{pk}, c))$*.*

$\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$**:** *the deterministic decryption algorithm takes as input a secret decryption key* $\mathsf{dk}_\rho = \mathsf{dk}'_\rho$*, a secret decryption key* $\mathsf{dk}_\mathbb{S} = \mathsf{dk}'_\mathbb{S}$ *for a circuit* $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ *and a ciphertext* $c = (c', \pi)$*. The algorithm first checks whether* $\mathsf{V}(\omega, (c', \mathsf{pk}, \mathsf{mpk}'), \pi) = 0$*. If true, returns* $\perp$*; else it returns* $\mathsf{Dec}'(\mathsf{dk}'_\rho, \mathsf{dk}_\mathbb{S}, c')$*.*

The correctness of the scheme follows directly by the correctness of the underlying ME scheme.

Theorem 1. *Let* $\Pi', \mathsf{SS}, \mathsf{NIZK}$ *be as above. If* Π' *is CPA private,* SS *is EUF-CMA secure and* NIZK *satisfies true-simulation extractability, then the ME scheme* Π *from construction 1 is CCA-secure.*

Proof. We can separately prove the two properties, by using Lemma 1 and Lemma 2.

Lemma 1. *Construction 1 achieves CCA authenticity.*

Proof. Let assume Π does not achieve CCA-authenticity. This implies that there exists a valid A able to win with non negligible probability $\mathbf{G}^{\mathsf{auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$. If this is the case, we can build a valid A' to win with non negligible probability $\mathbf{G}^{\mathsf{euf}}_{\mathsf{SS},\mathsf{A}'}(\lambda)$. The reduction is the following.

1. (setup) A' receives the public key pk from the challenger. Then it generates the keys $(\mathsf{msk}', \mathsf{kpol}, \mathsf{mpk}') \leftarrow\!\!\$\ \mathsf{Setup}'(1^\lambda)$ together with $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda)$. It gives A the master public key $\mathsf{mpk} = (\mathsf{mpk}', \mathsf{pk}, \omega)$.
2. (O_1) on input σ, A' invokes $\mathsf{O}_{\mathsf{Sign}}(\sigma)$ to receive a valid signature s for σ. Then it runs $\mathsf{ek}'_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}'(\mathsf{msk}, \sigma)$ and returns $\mathsf{ek}_\sigma = (\mathsf{ek}'_\sigma, s)$.
3. (O_2) on input ρ, A' returns $\mathsf{ek}'_\rho \leftarrow\!\!\$\ \mathsf{RKGen}'(\mathsf{msk}, \rho)$.
4. (O_3) on input $\mathbb{S}$, A' returns $\mathsf{ek}'_{\mathbb{S}} \leftarrow\!\!\$\ \mathsf{PolGen}'(\mathsf{msk}, \mathbb{S})$.
5. (encryption - O_5) on input $(\sigma, \mathbb{R}, m)$, computes a valid ciphertext $c' \leftarrow\!\!\$\ \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m)$, for some $\mathsf{ek}'_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}'(\mathsf{msk}, \sigma)$. Then returns $c = (c', \pi)$, for a simulated proof $\pi = \mathsf{Z}_1(\zeta, c')$.
6. (challenge) on input $((c', \pi), \rho, \mathbb{S})$, extracts (σ, s), running $\mathsf{K}_1(\xi, c', \pi)$, and forwards the tuple to the challenger.

First of all, note that on setup A' simulates an honest challenger for A in all but a single event: the setup of the CRS ω, indeed, is done through K_0, which allows A' to obtain the useful trapdoors. But this, clearly, does not alter (except with some negligible probability) the view of A, because of the assumption of true-simulation extractability of NIZK: moreover, for the very same reason, the distribution of the proofs generated by O_5 is computationally close to the original one. The simulation of O_2 and O_3 is tight, since they only require keys honestly generated by A'. Also O_1 queries are perfectly simulated, because of the powerful $\mathsf{O}_{\mathsf{Sign}}$ offered by the challenger, which allows A' to compute a valid signature on the sender attributes, thus preserving the correctness.

So, by assumption, A wins the game with some non negligible probability, thus producing a tuple $((c', \pi), \rho, \mathbb{S})$; in order to be acceptable, such tuple should be such that the proof is accepted and, except with some negligible probability, the extractor K_1 is able to reconstruct the sender attributes σ and a valid s on the very same attributes. Note that, since A is valid, A' never queries $\mathsf{O}_{\mathsf{Sign}}(\sigma)$: so (σ, s) is a valid forgery for $\mathbf{G}^{\mathsf{euf}}_{\mathsf{SS},\mathsf{A}'}(\lambda)$. □

Lemma 2. *Construction 1 achieves CCA privacy.*

Proof. Let assume that Π does not achieve CCA-privacy. This implies that there exists a valid A able to win with non negligible probability $\mathbf{G}^{\mathsf{priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$. If this is the case, we can build a valid A' to win with non negligible probability $\mathbf{G}^{\mathsf{priv}}_{\Pi',\mathsf{A}}(\lambda)$. The reduction is the following.

1. (setup) A' receives the master public key mpk' from the challenger. Then it computes $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda)$, the signatures keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ and gives A the new master public key $\mathsf{mpk} = (\mathsf{mpk}', \mathsf{pk}, \omega)$.

2. (O_1) on input σ, A' invokes $\mathsf{O}_1(\sigma)$ to have back ek'_σ; moreover it runs $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$. Then it gives A the new sender key $\mathsf{ek}_\sigma = (\mathsf{ek}'_\sigma, s)$.

3. (O_2) on input ρ, A' invokes $\mathsf{O}_2(\rho)$ and forwards the received ek'_ρ.

4. (O_3) on input $\mathbb{S}$, A' invokes $\mathsf{O}_3(\mathbb{S})$ and simply forwards the received $\mathsf{ek}'_\mathbb{S}$.

5. (decryption - O_4) on input $(\rho, \mathbb{S}, (c', \pi))$, it first parses the ciphertext and checks whether the proof π is valid or not (in this case simply outputs $\perp$). Then, extracts $(\sigma, s, \mathbb{R}, m)$ using $\mathsf{K}_1(\xi, c', \pi)$ and if there is a match returns m, otherwise returns $\perp$.

6. (challenge) the input tuple is passed to the challenger that computes c'_b; A' returns back (c'_b, π), where $\pi = \mathsf{Z}_1(\zeta, c'_b)$ is a simulated proof for c'_b.

7. (output) the bit b' is given to the challenger.

The view offered by A' to A is computationally close to the original one: the difference, again, is due to the different mechanism to generate the CRS ω, but this cannot be distinguished by A, because of the assumption of f-tSE of NIZK. Oracles O_2 and O_3 are perfectly simulated, since they are offered directly by the challenger. O_1 queries are again perfectly simulated, because the sender key is honestly generated by the challenger and the signature is valid (and verifiable by A). By assumtpion, O_4 queries are perfectly simulated except with some negligible probability, due to the failure of the extractor $\mathsf{K}_1(\xi, \cdot, \cdot)$: the correctness is still preserved because A' can extract all the necessary information to succesfully decrypt a message if and only if a match occurs. □

Since we have shown that Construction 1 achieves both CCA-privacy and CCA-authenticity, we have concluded the proof. □

Let Π' be an A-ME scheme which satisfies privacy. If we use a signature scheme and a NIZK proof system, we can obtain a new A-ME scheme Π which preserves its privacy, but also achieves authenticity. The construction is as follows.

Construction 2. *Let* $\Pi' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{RKGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be an A-ME scheme,* $\mathsf{SS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme* SS *and* $\mathsf{NIZK} = (\mathsf{I}, \mathsf{P}, \mathsf{V})$ *be a f-tSE NIZK argument for the following NP relation:*

$$R \stackrel{def}{=} \left\{ ((\sigma, s, \mathbb{R}, m, r), (\mathsf{mpk}', \mathsf{pk}, c)) \ : \begin{array}{c} c = \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m; r) \wedge \\ \mathsf{Ver}(\mathsf{pk}, (\sigma, s)) = 1 \end{array} \right\}.$$

where $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, s, \mathbb{R}, m)$*, so* f *extracts all the input but the randomness* r*.*

We construct the new A-ME scheme Π *as follows:*

$\mathsf{Setup}(1^\lambda)$**:** *on input the security parameter* 1^λ*, the algorithm computes* $(\mathsf{msk}', \mathsf{mpk}') \leftarrow\!\!\$\ \mathsf{Setup}'(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ *and* $\omega \leftarrow\!\!\$\ \mathsf{I}(1^\lambda)$*. The algorithm outputs* $\mathsf{msk} = (\mathsf{msk}', \mathsf{sk})$ *as the master secret key and* $\mathsf{mpk} = (\mathsf{mpk}', \mathsf{pk}, \omega)$ *as the master public key. All other algorithms are implicitly given* mpk *as additional input.*

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$**:** *the randomized sender-key generator takes as input the master secret key* $\mathsf{msk} = (\mathsf{msk}', \mathsf{sk})$ *and attributes* $\sigma \in \{0,1\}^*$*. The algorithm returns the encryption key* $\mathsf{ek}_\sigma = (\mathsf{ek}'_\sigma, s)$*, where* $\mathsf{ek}'_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}'(\mathsf{msk}', \sigma)$ *and* $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho, \mathbb{S})$**:** *the randomized receiver-key generator takes as input the master secret key* $\mathsf{msk} = (\mathsf{msk}', \mathsf{sk})$*, attributes* $\rho \in \{0,1\}^*$ *and a policy* $\mathbb{S} : \{0,1\}^* \to \{0,1\}$*. The algorithm computes the key* $\mathsf{dk}_{\rho,\mathbb{S}} \leftarrow\!\!\$\ \mathsf{RKGen}'(\mathsf{msk}', \rho, \mathbb{S})$*.*

$\mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)$**:** *the randomized encryption algorithm takes as input a secret encryption key* $ek_\sigma = (\mathsf{ek}'_\sigma, s)$ *for attributes* $\sigma \in \{0,1\}^*$*, a policy* $\mathbb{R} : \{0,1\}^* \to \{0,1\}$ *and a message* $m \in \{0,1\}^*$*. The algorithm first encrypts the message by computing* $c' \leftarrow\!\!\$\ \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m)$*; then, it returns the ciphertext* $c = (c', \pi)$*, where* $\pi \leftarrow\!\!\$\ \mathsf{P}(\omega, (\sigma, s, \mathbb{R}, m, r), (\mathsf{mpk}', \mathsf{pk}, c))$*.*

$\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}}, c)$**:** *the deterministic decryption algorithm takes as input a decryption key* $\mathsf{dk}_{\rho,\mathbb{S}}$ *and a ciphertext* $c = (c', \pi)$*. The algorithm first checks whether* $\mathsf{V}(\omega, (c', \mathsf{pk}, \mathsf{mpk}'), \pi) = 0$*. If true, returns* $\perp$*; else it returns* $\mathsf{Dec}'(\mathsf{dk}'_{\rho,\mathbb{S}}, c')$*.*

The correctness of the scheme follows directly by the correctness of the underlying A-ME scheme.

Theorem 2. *Let* $\Pi', \mathsf{SS}, \mathsf{NIZK}$ *be as above. If* Π' *is CPA private,* SS *is EUF-CMA secure and* NIZK *satisfies true-simulation extractability, then the A-ME scheme* Π *from construction 2 is CCA-secure.*

Proof. We can separately prove the two properties, by using Lemma 3 and Lemma 4.

Lemma 3. *Construction 2 achieves CCA-authenticity.*

Proof. The proof is almost the same of Lemma 1: the only difference is that here we do not need to simulate anymore the generation of the policy keys. □

Lemma 4. *Construction 2 achieves CCA privacy.*

Proof. As before, this proof is almost the same of Lemma 2: the only difference is that here we do not need to simulate anymore the generation of the policy keys. □

Since we have shown that Construction 2 achieves both CCA-privacy and CCA-authenticity, we have concluded the proof. □

Chapter 5

Conclusions

Write conclusions

Bibliography

[1] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In *EUROCRYPT*, 2019.

[2] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2005*, pages 457–473. Springer Berlin Heidelberg, 2005.

[3] Nuttapong Attrapadung and Hideki Imai. Dual-policy attribute based encryption. In *ACNS*, page 168–185, 2009.

[4] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. 1998.