SAPIENZA
UNIVERSITÀ DI ROMA

Matchmaking Encryption with Security against Chosen-Ciphertext Attacks

Faculty of Information Engineering, Informatics, and Statistics

Master of Science in Engineering in Computer Science

Candidate

Luigi Russo
ID number 1699981

Thesis Advisors

Prof. Riccardo Lazzeretti
Prof. Daniele Venturi

Academic Year 2019/2020

Matchmaking Encryption with Security against Chosen-Ciphertext Attacks

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

Acknowledgments

Add section

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 (cf. §2.5) 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 order 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.

In particular, in this context, we consider two parties: the sender Alice with attributes σ and the receiver Bob 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, Alice can encrypt a message m containing her 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, Bob will be able to decrypt the message only if Alice 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.

In Matchmaking Encryption, a trusted authority generates encryption and decryption keys associated, respectively, to attributes of the sender and the receiver. The authority also generates an additional decryption key for the

receiver, associated to an arbitrary policy of its choice. The sender of the message can specify on the fly an arbitrary policy the receiver must satisfy in order for the message to be revealed. The guarantee is now that the receiver will obtain the message if and only if a match occurs. An example is given in figure 1.1.

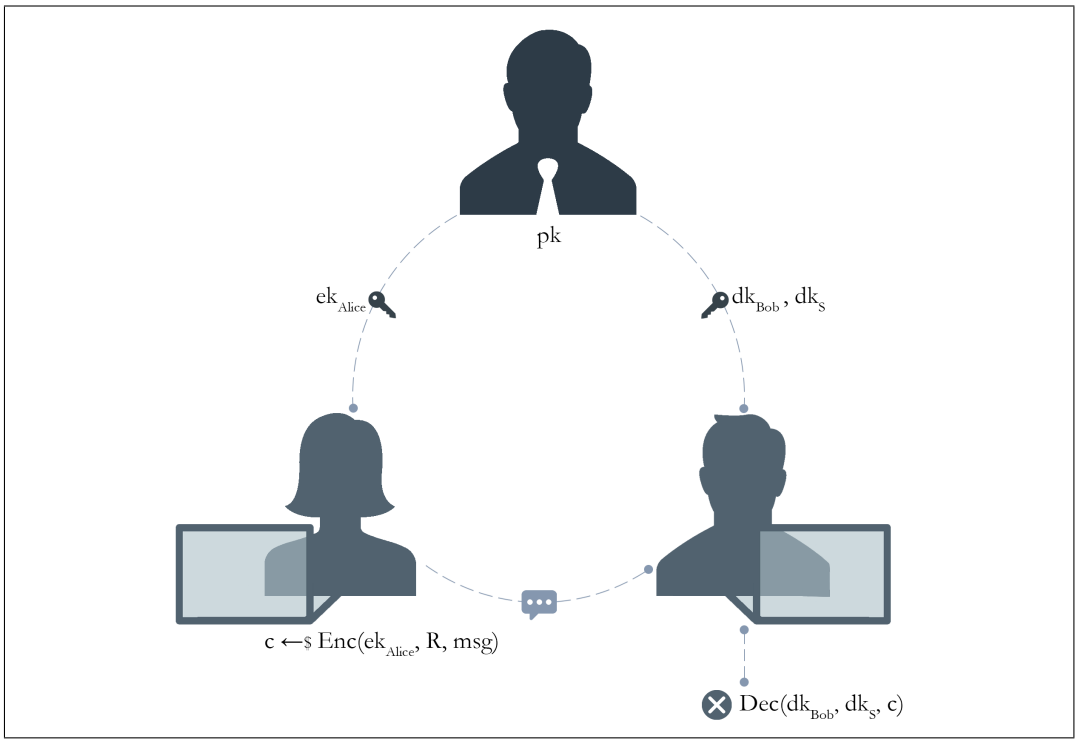

Figure 1.1. ME example. Alice encrypts a message with her encryption key $\mathsf{ek}_{\text{Alice}}$, provided by the trusted party: she specifies some arbitrary policy $\mathbb{R}$. When Bob receives the ciphertext, tries to decrypt it with his decryption key dk_{Bob} and a key associated with some arbitrary policy $\mathbb{S}$.

Matchmaking Encryption can be thought of as a non-interactive version of Secret Handshakes (cf. §2.4), introduced in 2003 by Balfanz et al. [2]. Indeed, Matchmaking Encryption gives privacy guarantees similar to that of SH, but it provides a more efficient way to communicate (being non-interactive) and, at the same time, it is more flexible since a party is not constrained to a group.

1.1 Security

Security is defined via two properties: privacy and authenticity.

Privacy. 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. 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 operation.

Authenticity. 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.

1.2 Arranged ME

Ateniese et al. also defined a possible variant of ME, namely Arranged Matchmaking Encryption (A-ME). The main difference is that in A-ME there is a single decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$ which describes simultaneously the receiver's attributes ρ and the policy $\mathbb{S}$ chosen by the receiver, while ME makes use of two different keys: one for the policy and the other one for the receiver's attributes.

1.3 Thesis contributions

CCA security. In this document, we revamp the notion of ME by extending the security definitions to the setting of chosen-ciphertext attacks (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 that answers decryption queries. This captures privacy even in the presence of attackers that can obtain the decryption of possibly mauled ciphertexts. Cf. §4.1.
- CCA-authenticity: the attacker is also given access to an encryption oracle. This captures authenticity in the presence of attackers that try

[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.

[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.

to generate valid ciphertexts by mauling previously obtained ciphertexts. Cf. §4.2.

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

Constructions. We define black-box constructions to achieve CCA-privacy (cf. §4.3.1) and CCA-authenticity (cf. §4.3.2), by relying on a Non-Interactive Zero-Knowledge (NIZK) argument system and a Signatures scheme. We finally define black-box constructions to directly transform any CPA-secure ME (resp. A-ME) scheme into a CCA-private ME (resp. A-ME) scheme, again based on a NIZK argument system and Digital Signatures (cf. §4.3.3).

In particular, in our transformation (cf. §4.3), 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.

Chapter 2

Preliminaries

In this chapter we introduce some basic ingredients: we give a formal definition of Digital Signatures (§2.2), Non-Interactive Zero-Knowledge proof systems (§2.3), Attribute-Based Encryption (§2.5) and finally Secret Handshakes (§2.4).

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

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

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*:*

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

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).*

2.4 Secret Handshakes

Introduced by Balfanz et al. [2] in 2003, a Secret Handshake is a key agreement protocol that allows two members of the same group to secretly authenticate to each other and agree on a symmetric key. During the protocol, a party can additionally specify the precise group identity that the other party should have: e.g. it can specify its role.

SH preserves the privacy of the participants: when the handshake is successful (i.e. the key is correctly derived) they only learn that they both belong to the same group; yet, their identities remain secret. On the contrary, they learn nothing if the handshake fails.

Subsequent work in the area have focused on improving on various aspects of SH: e.g. members' privacy and expressiveness of the matching policies (attribute-based Secret Handshakes).

2.5 Attribute-Based Encryption

Attribute-Based Encryption (ABE) was introduced in 2005 by Sahai and Waters [5]. It enables an access control mechanism over encrypted data: both users' private keys and ciphertexts are associated with a set of ascribed attributes or some policy over them. A user is then able to decrypt a ciphertext if there is a match between his private key and the ciphertext, whereas in traditional PKE schemes messages are encrypted to one particular user only.

ABE comes in two flavors, namely Ciphertext-Policy (CP) ABE and Key-Policy (KP) ABE.

CP-ABE. In Ciphertext-Policy ABE, Alice wants to encrypt a message and can express any access policy, stating what kind of receivers will be able to decrypt the ciphertext. Such a policy is specified in terms of access structure

over receiver attributes. Each user is indeed given a private key by the authority on the basis of its attributes. Such a user can decrypt a ciphertext if his attributes satisfy the access policy associated with the ciphertext. In figure 2.1 there is an illustration for a CP-ABE example.

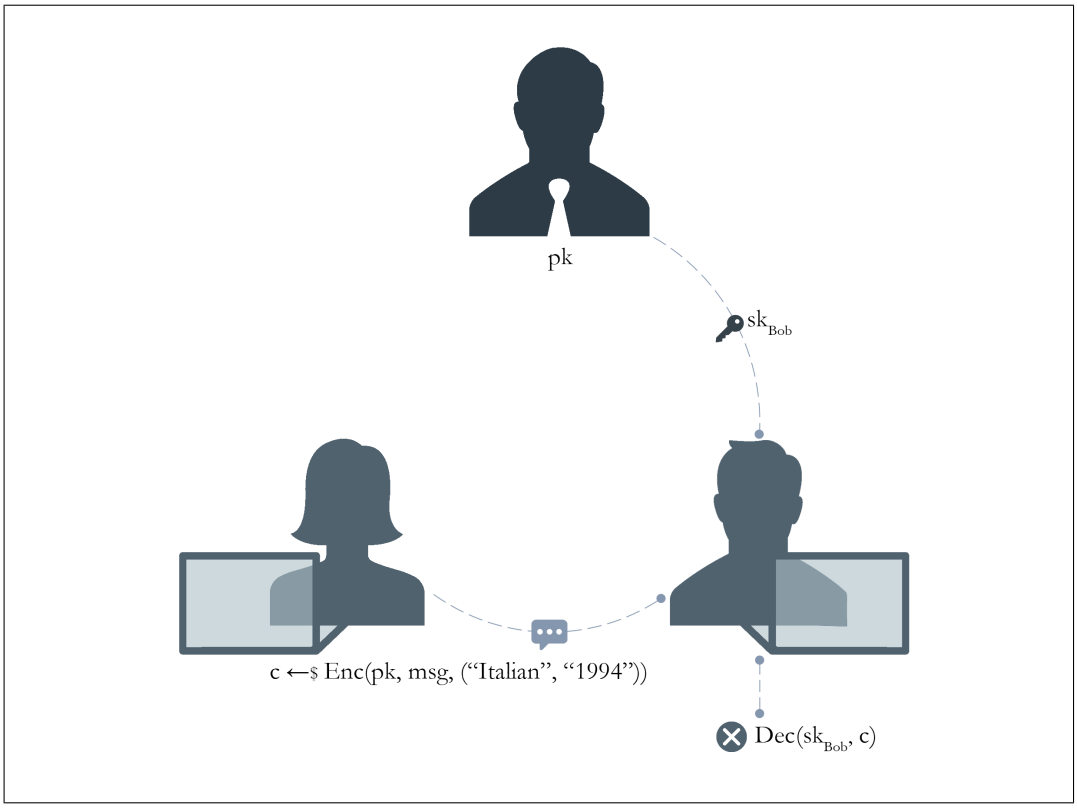

Figure 2.1. CP-ABE example. Alice wants her message to be decrypted by anyone who lives in Italy and was born in 1990. Bob lives in Milan and was born in 1994, so receives a private key associated with his attributes; since there is not a match, he is not able to recover the message.

KP-ABE. In Key-Policy ABE, instead, the roles of an attribute set and an access policy are swapped. Indeed, attribute sets are used to annotate the ciphertexts and access policies over these attributes are associated with users' secret keys. In figure 2.2 there is an illustration for a KP-ABE example.

Dual-policy ABE. In ABE, it is clear that only one party can specify a policy, and thus only one user has the possibility to select the source (resp. the destination) of an encrypted message. Motivated by this limitation, Attrapadung and Imai [3] introduced dual-policy ABE. Here, the sender encrypts a message by choosing both a policy and a set of attributes; the receiver can decrypt the ciphertext using a single decryption key associated with both

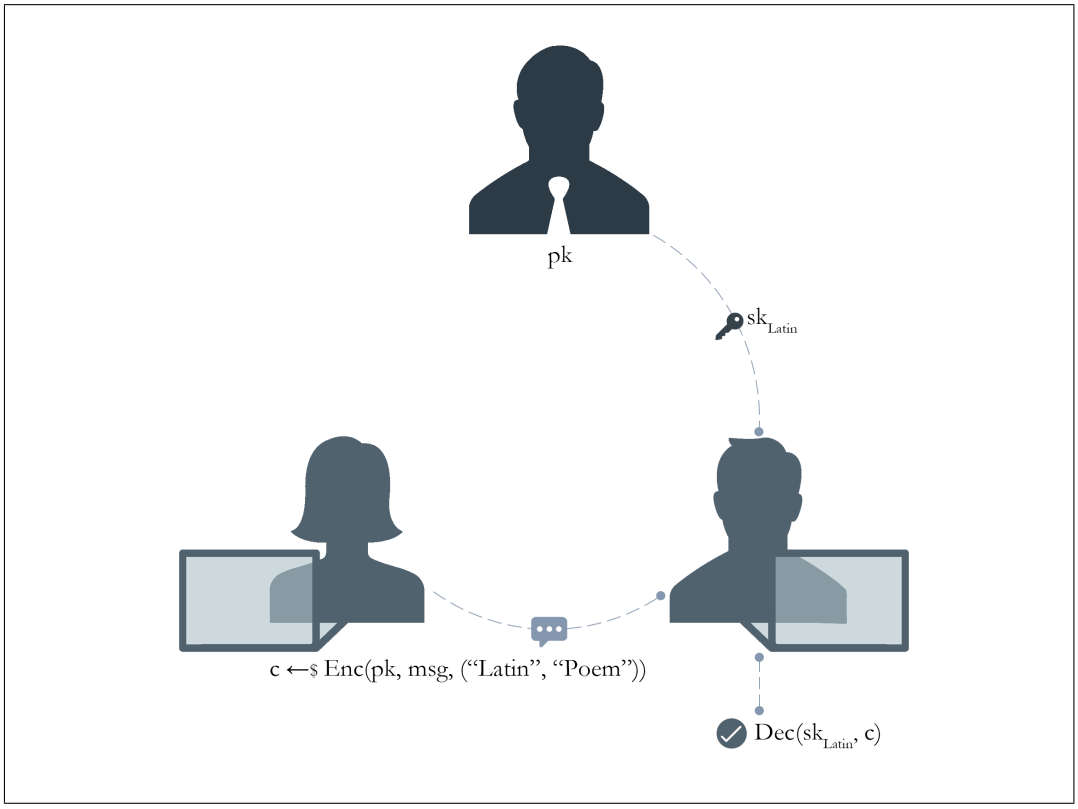

Figure 2.2. KP-ABE example. Alice annotates her message as a Latin poem. Bob is issued some key thanks to which can decrypt any Latin work. In this case, there is a match, i.e. Bob can decrypt the ciphertext.

the receiver's policy and its attributes. If both policies are satisfied by the respective counterpart, the message is revealed.

Chapter 3

Matchmaking Encryption

In this chapter we briefly introduce in a more formal way 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. §3.1).

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 Attacks Security

In this chapter we extend the security definitions of an ME to the setting of chosen-ciphertext attacks (CCA). As in the original definition, security is defined via two properties: privacy and authenticity.

4.1 Privacy

We want privacy to capture secrecy of the sender's inputs (i.e., the attributes σ, the policy for the receiver $\mathbb{R}$, and the plaintext m), in two different scenarios: in case of a match between the sender's and receiver's attributes/policy, and in case of mismatch. More formally, we require that the distributions $\mathsf{Enc}(\mathsf{ek}_{\sigma_0}, \mathbb{R}_0, m_0)$ and $\mathsf{Enc}(\mathsf{ek}_{\sigma_1}, \mathbb{R}_1, m_1)$ are computationally indistinguishable to the eyes of an attacker with oracle access to SKGen, RKGen, PolGen, where the values $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$ are all chosen by the adversary. In order to exclude some trivial attacks, it is necessary to quantify privacy over all valid adversaries, as explained in [1].

In this thesis, we aim at extending the power and the flexibility of the attacker, providing an oracle access to $\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$, where the input is controlled by the attacker: clearly, we need to exclude the trivial attack of exploiting the oracle to directly decrypt the challenge chipertext[1]. We introduce the formal definition of CCA privacy below for ME (Definition 7) and A-ME too (Definition 8).

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

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

[1] This is standard in CCA games definitions.

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

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

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*

$$(\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho) = 0) \vee (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1) = 0)$$
$$\vee\, (\mathbb{R}_0(\rho) = \mathbb{S}(\sigma_1) = 0) \vee (\mathbb{R}_1(\rho) = \mathbb{S}(\sigma_0) = 0); \tag{4.1}$$

- ***(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)*

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

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*

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

- ***(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

Authenticity, instead, demands that the only way to produce a valid ciphertext under sender attributes σ is to obtain an encryption key ek_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. This captures security against malicious senders and is modeled by a game in which the attacker has oracle access to SKGen, RKGen, and PolGen. The attacker's goal is to output a tuple $(\rho, \mathbb{S}, c)$ such that $\mathsf{Dec}(dk_\rho, \mathsf{dk}_\mathbb{S}, c) \neq \perp$, and none of the encryption keys ek_σ for attributes σ (obtained by the adversary via oracle queries) satisfies the policy $\mathbb{S}$.

The adversary is now given access to an encryption oracle: given in input sender attributes σ, a policy $\mathbb{R}$ and a message m, the attacker can compute an honestly generated ciphertext c, without the need to obtain an encryption key ek_σ from the authority. We clearly need to avoid trivial attacks: we do not allow the attacker to return as challenge ciphertext one provided by the oracle. We introduce the formal definition of CCA privacy below for ME (Definition 9) and A-ME too (Definition 10).

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*:*

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

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

We are going to define:

- a first black-box transformation from CPA to CCA privacy, while preserving the authenticity
- a second black-box transformation from CPA to CCA authenticity, while preserving privacy

It is evident that in order to achieve CCA security, we can simply apply both of these transformations in sequence: however, for sake of convenience, we will present a more efficient one which directly achieves both CCA privacy and CCA authenticity. Figure 4.3 gives a visual representation of the transformations presented in this thesis, explaining how to achieve CCA.

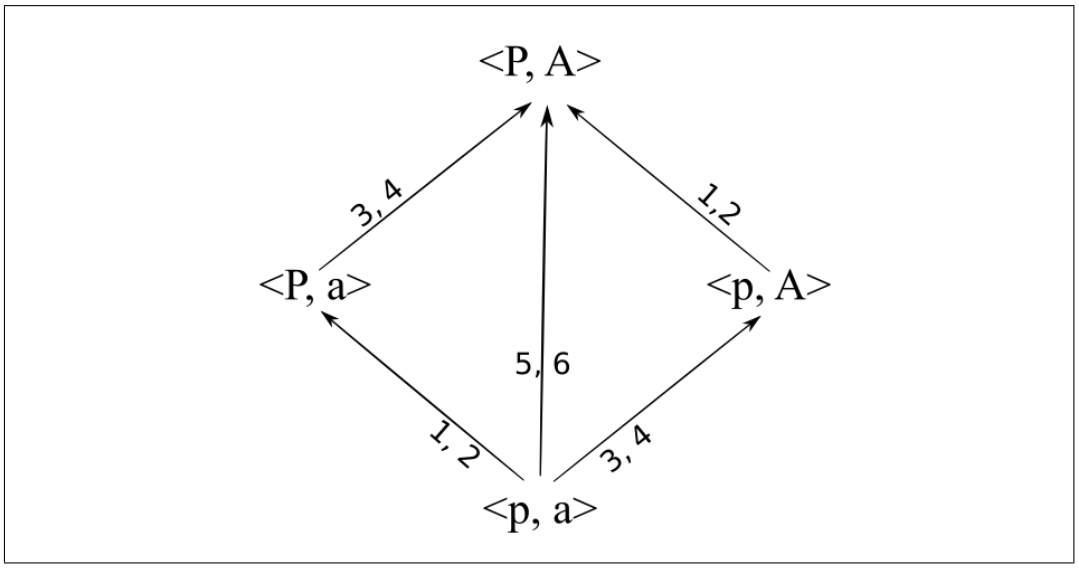

Figure 4.3. CCA Transormations Lattice. P represents Privacy, while A stands for Authenticity; the small letter indicates CPA, while the capital letter indicates CCA; the numbers refer to the corresponding theorem.

The starting point for our transformations are the following constructions, based on Digital Signatures and NIZK.

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:*

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

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

$\mathsf{Setup}(1^\lambda)$**:** *on input the security parameter, gets* $(\mathsf{msk}', \mathsf{kpol}, \mathsf{mpk}') \leftarrow_\$ \mathsf{Setup}'(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$ *and* $\omega \leftarrow_\$ \mathsf{I}(1^\lambda)$*. Then 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 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, it simply returns* $\perp$*; otherwise, it returns* $\mathsf{Dec}'(\mathsf{dk}'_\rho, \mathsf{dk}_\mathbb{S}, c')$*.*

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 and* $\mathsf{NIZK} = (\mathsf{I}, \mathsf{P}, \mathsf{V})$ *be a f-tSE NIZK argument for the following NP relation:*

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

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

$\mathsf{Setup}(1^\lambda)$**:** *on input the security parameter* 1^λ*, 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* $\mathsf{Dec}'(\mathsf{dk}'_{\rho,\mathbb{S}}, c')$*.*

We also give two simple variants which do not require any Signatures Scheme.

Construction 3. *Let* $\Pi' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{RKGen}', \mathsf{PolGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be an ME scheme, and* $\mathsf{NIZK} = (\mathsf{I}, \mathsf{P}, \mathsf{V})$ *be a* f*-tSE NIZK argument for the following NP relation:*

$$R \stackrel{def}{=} \left\{ ((\sigma, \mathbb{R}, m, r), (\mathsf{mpk}', c)) \ : c = \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m; r) \right\}.$$

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

$\mathsf{Setup}(1^\lambda)$**:** *on input the security parameter, gets* $(\mathsf{msk}', \mathsf{kpol}, \mathsf{mpk}') \leftarrow_\$ \mathsf{Setup}'(1^\lambda)$*, and* $\omega \leftarrow_\$ \mathsf{I}(1^\lambda)$*. Then outputs* $\mathsf{msk} = (\mathsf{msk}', \mathsf{kpol})$ *as the master secret key and* $\mathsf{mpk} = (\mathsf{mpk}', \omega)$ *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* msk *and attributes* $\sigma \in \{0,1\}^*$*. The algorithm returns the encryption key* $\mathsf{ek}_\sigma \leftarrow_\$ \mathsf{SKGen}'(\mathsf{msk}', \sigma)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho)$**:** *the randomized receiver-key generator takes as input the master secret key* msk *and attributes* $\rho \in \{0,1\}^*$*. The algorithm 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 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_σ *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* $\pi \leftarrow_\$ \mathsf{P}(\omega, (\sigma, \mathbb{R}, m, r), (\mathsf{mpk}', c))$*.*

$\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c)$**:** *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* $\mathsf{V}(\omega, (c', \mathsf{mpk}'), \pi) = 0$*: if true, it simply returns* $\perp$*; otherwise, it returns* $\mathsf{Dec}'(\mathsf{dk}'_\rho, \mathsf{dk}_\mathbb{S}, c')$*.*

Construction 4. *Let* $\Pi' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{RKGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be an A-ME scheme, and* $\mathsf{NIZK} = (\mathsf{I}, \mathsf{P}, \mathsf{V})$ *be a f-tSE NIZK argument for the following NP relation:*

$$R \stackrel{def}{=} \left\{ \; ((\sigma, \mathbb{R}, m, r), (\mathsf{mpk}', c)) \;\; : c = \mathsf{Enc}'(\mathsf{ek}'_\sigma, \mathbb{R}, m; r) \; \right\}.$$

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

$\mathsf{Setup}(1^\lambda)$**:** *on input the security parameter* 1^λ*, computes* $(\mathsf{msk}', \mathsf{mpk}') \leftarrow_\$ \mathsf{Setup}'(1^\lambda)$*, and* $\omega \leftarrow_\$ \mathsf{I}(1^\lambda)$*. The algorithm outputs* $\mathsf{msk} = \mathsf{msk}'$ *as the master secret key and* $\mathsf{mpk} = (\mathsf{mpk}', \omega)$ *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* msk *and attributes* $\sigma \in \{0,1\}^*$*. The algorithm returns the encryption key* $\mathsf{ek}_\sigma \leftarrow_\$ \mathsf{SKGen}'(\mathsf{msk}', \sigma)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho, \mathbb{S})$**:** *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_σ *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* $\pi \leftarrow\!\!\$\ \mathsf{P}(\omega, (\sigma, \mathbb{R}, m, r), (\mathsf{mpk}', c))$*.*

$\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}}, c)$**:** *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* $\mathsf{V}(\omega, (c', \mathsf{mpk}'), \pi) = 0$*. If true, returns* $\perp$*; else it returns* $\mathsf{Dec}'(\mathsf{dk}'_{\rho,\mathbb{S}}, c')$*.*

The correctness of all the previous schemes follows directly by the correctness of the underlying ME (resp. A-ME) schemes.

4.3.1 CCA privacy

We define a black-box construction to transform any CPA-private ME scheme into a CCA-private ME scheme, using construction 1.

Theorem 1. *Let* Π', NIZK *be as above. If* Π' *is CPA private, and* NIZK *satisfies true-simulation extractability for* $f(\sigma, \mathbb{R}, m, r) = (\sigma, \mathbb{R}, m)$*, then the ME scheme* Π *from construction 1 is CCA-private and preserves its authenticity.*

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

Lemma 1. *Constructions 1 and 3 achieve CCA privacy if* f *can extract the sender attributes* σ*, the policy* $\mathbb{R}$ *and the message* m*.*

Proof. Without loss of generality, we give the proof for construction 1 only: the other case is implied since it is a subcase. 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 computes $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$. 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 $(\sigma, \mathbb{R}, m)$ 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. □

Lemma 2. *Construction 3 preserves its authenticity.*

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

1. (setup) A' receives the master public key mpk' from the challenger. Then it computes $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda)$, and gives A the new master public key $\mathsf{mpk} = (\mathsf{mpk}', \omega)$.
2. (O_1) on input σ, A' invokes
3. (O_1) on input σ, A' invokes $\mathsf{O}_1(\sigma)$ to have back the sender key ek_σ which is forwarded to the A.
4. (O_2) on input ρ, A' invokes $\mathsf{O}_2(\rho)$ and forwards the received ek'_ρ.
5. (O_3) on input $\mathbb{S}$, A' invokes $\mathsf{O}_3(\mathbb{S})$ and simply forwards the received $\mathsf{ek}'_\mathbb{S}$.

6. (encryption - O_5)[2] invokes $\mathsf{O}_5(\rho)$ and forwards the received ciphertext c.
7. (challenge) on input $((c', \pi), \rho, \mathbb{S})$, forwards the tuple $(c, \rho, \mathbb{S})$ to the challenger.

□

Since we have shown that Construction 3 achieves CCA-privacy and preserves its authenticity, we have concluded the proof. □

In order to achieve CCA-private A-ME schemes, we can rely on construction 2 and define the following transormation.

Theorem 2. *Let* Π', NIZK *be as above. If* Π' *is CPA private, and* NIZK *satisfies true-simulation extractability for* $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, \mathbb{R}, m)$*, then the ME scheme* Π *from construction 2 is CCA-private and preserves its authenticity.*

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

Lemma 3. *Constructions 2 and 4 achieve CCA privacy.*

Proof. As before, this proof is almost the same as Lemma 1 (wlog, we only proove the lemma for construction 2): the only difference is that here we do not need to simulate anymore the generation of the policy keys. Let assume that Π does not achieve CCA-privacy. This implies that there exists a valid A able to win with non negligible probability $\mathbf{G}^{\mathsf{arr\text{-}priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$. If this is the case, we can build a valid A' to win with non negligible probability $\mathbf{G}^{\mathsf{arr\text{-}priv}}_{\Pi',\mathsf{A}}(\lambda)$. 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 computes $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$. Then it gives A the new sender key $\mathsf{ek}_\sigma = (\mathsf{ek}'_\sigma, s)$.
3. (O_2) on input $(\rho, \mathbb{S})$, A' invokes $\mathsf{O}_2(\rho, \mathbb{S})$ and forwards the received $\mathsf{ek}'_{\rho,\mathbb{S}}$.
4. (decryption - O_3) 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 $\perp$.
5. (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.

[2] only for CCA game

6. (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. OraclesO_2 is perfectly simulated, since it is 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_3 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. □

Lemma 4. *Construction 2 preserves its authenticity.*

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

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

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

5. (encryption - O_4)[3] invokes $\mathsf{O}_4(\rho)$ and forwards the received ciphertext c.

6. (challenge) on input $((c', \pi), \rho, \mathbb{S})$, forwards the tuple $(c, \rho, \mathbb{S})$ to the challenger.

□

Since we have shown that Construction 4 achieves CCA-privacy and preserves its authenticity, we have concluded the proof. □

[3]only for CCA game

4.3.2 CCA authenticity

We define a black-box construction to achieve CCA-authenticity, using construction 1.

Theorem 3. *Let* $\Pi', \mathsf{SS}, \mathsf{NIZK}$ *be as above. If* Π' *has CPA authenticity,* SS *is EUF-CMA secure and* NIZK *satisfies true-simulation extractability for* $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, \mathbb{R}, m)$*, then the ME scheme* Π *from construction 1 achieves CCA authenticity and preserves its privacy.*

Proof. We can separately prove the two properties, by using Lemma 5 and Lemma 6

Lemma 5. *Construction 1 preserves CPA (resp. CCA) privacy.*

Proof. Let assume that Π does not preserve its privacy. This implies that there exists a valid A able to win with non negligible probability $\mathbf{G}^{\mathsf{priv}}_{\Pi,\mathsf{A}}(\lambda)$ (resp. $\mathbf{G}^{\mathsf{priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$). If this is the case, we can build a valid A' to win with non negligible probability $\mathbf{G}^{\mathsf{priv}}_{\Pi',\mathsf{A}}(\lambda)$ (resp. $\mathbf{G}^{\mathsf{priv\text{-}cca}}_{\Pi',\mathsf{A}}(\lambda)$). 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 computes $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$. 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)[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, invokes O_4 and forwards the result.
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, O_3 and O_4 are perfectly simulated, since they are offered

[4] Only for CCA game.

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). □

Lemma 6. *Construction 1 achieves CCA authenticity if f can extract the sender attributes σ and the signature s.*

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 the tuple $(\sigma, \mathbb{R}, m)$, 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 $\mathbf{G}^{\mathsf{euf}}_{\mathsf{SS},\mathsf{A}'}(\lambda)$. □

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

We present here how to achieve CCA-authenticity for A-ME, using construction 2.

Theorem 4. *Let* $\Pi', \mathsf{SS}, \mathsf{NIZK}$ *be as above. If* Π' *has CPA authenticity,* SS *is EUF-CMA secure and* NIZK *satisfies true-simulation extractability for* $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, \mathbb{R}, m)$*, then the ME scheme* Π *from construction 2 achieves CCA authenticity and preserves its privacy.*

Proof. We can separately prove the two properties, by using Lemma 7 and Lemma 8

Lemma 7. *Construction 2 preserves CPA (resp. CCA) privacy.*

Proof. Let assume that Π does not preserve its privacy. This implies that there exists a valid A able to win with non negligible probability $\mathbf{G}^{\mathsf{arr\text{-}priv}}_{\Pi,\mathsf{A}}(\lambda)$ (resp. $\mathbf{G}^{\mathsf{arr\text{-}priv\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$). If this is the case, we can build a valid A' to win with non negligible probability $\mathbf{G}^{\mathsf{arr\text{-}priv}}_{\Pi',\mathsf{A}}(\lambda)$ (resp. $\mathbf{G}^{\mathsf{arr\text{-}priv\text{-}cca}}_{\Pi',\mathsf{A}}(\lambda)$). 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 computes $s \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, \sigma)$. 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. (decryption - O_3)[5] 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, invokes O_3 and forwards the result.
5. (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.
6. (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 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, O_3 are perfectly simulated, since they are offered directly by the

[5] Only for CCA game.

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). □

Lemma 8. *Construction 2 achieves CCA-authenticity.*

Proof. The proof is almost the same as Lemma 6: the only difference is that here we do not need to simulate anymore the generation of the policy keys. Let assume Π does not achieve CCA-authenticity. This implies that there exists a valid A able to win with non negligible probability $\mathbf{G}^{\mathsf{arr\text{-}auth\text{-}cca}}_{\Pi,\mathsf{A}}(\lambda)$. 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 $(\rho, \mathbb{S})$, A' returns $\mathsf{ek}'_{\rho,\mathbb{S}} \leftarrow\!\!\$\ \mathsf{RKGen}'(\mathsf{msk}, \rho, \mathbb{S})$.
4. (encryption - O_4) on input the tuple $(\sigma, \mathbb{R}, m)$, it 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 $\pi = \mathsf{Z}_1(\zeta, c')$.
5. (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_4 is computationally close to the original one. The simulation of O_2 is tight, since it only requires 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 $\mathbf{G}^{\mathsf{euf}}_{\mathsf{SS},\mathsf{A}'}(\lambda)$. □

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

4.3.3 Direct Transformation

We now define a black-box construction to transform any CPA-secure ME scheme into a CCA-secure ME scheme, by relying on construction 1.

Theorem 5. *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 for* $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, s, \mathbb{R}, m)$*, then the ME scheme* Π *from construction 1 is CCA-secure.*

Proof. We can separately prove the two properties, by using Lemma 6 and Lemma 1. Since we have shown that Construction 1 achieves both CCA-privacy and CCA-authenticity, we have concluded the proof. □

For A-ME schemes we can do the same, using again construction 2 as starting point.

Theorem 6. *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 for* $f(\sigma, s, \mathbb{R}, m, r) = (\sigma, s, \mathbb{R}, m)$*, then the ME scheme* Π *from construction 2 is CCA-secure.*

Proof. We can separately prove the two properties, by using Lemma 8 and Lemma 3. Since we have shown that Construction 2 achieves both CCA-privacy and CCA-authenticity, we have concluded the proof. □

Chapter 5

Conclusions

Matchmaking Encryption leaves open several important questions. First, it would be interesting to construct ME from simpler assumptions. Second, a natural direction is to come up with efficient ME schemes for the identity-based setting without relying on random oracles. Further extensions include ME with multiple authorities and mitigating key escrow, as well as black-box constructions from ABE schemes.

5.1 ME from standard assumptions

All the proposed constructions for Matchmaking Encryption rely on concrete cryptographic assumptions such as Randomized Functional Encryption, Digital Signatures and Non-Interactive Zero-Knowledge proofs. It would be interesting to explore which are the minimal assumptions needed in order to instantiate a ME system, similar to what has been done for ABE schemes by Itkis et al. [6] in 2017. In particular, the authors of [6] show a construction which only requires a semantically secure public-key encryption scheme, at the cost of tolerating collusions of arbitrary but a priori bounded size.

5.2 Efficient IB-ME constructions

Identity-Based ME is a special case of ME which only considers equality policies: this implies that rather than providing circuits, the two parties directly specify the identity of their ideal partner. A natural extension could be defining an IB-ME construction which is fully secure in the standard model, without relying on random oracles[1], and also offering several advantages over general ME schemes: namely, computational efficiency and shorter public parameters. One of the possible constructions could be based on decisional Bilinear

[1] A random oracle responds to every unique query with a truly random value chosen uniformly from its output domain.

Diffie-Hellman Exponent assumption, which has already been used to construct simple and efficient hierarchical IBE schemes [7].

5.3 Mitigating key escrow

In ME, as well as in any ABE system, all users' private keys are issued by an unconditionally trusted authority. Such an authority owns the master secret key of the system, and can decrypt all ciphertexts encrypted to any user: potentially this is the target for some attacker, which can obtain private keys and redistribute them for malicious use. Thus, it has great significance reducing the trust in the authority in an ME system. Some techniques to mitigate the key escrow have already been proposed for IBE and ABE schemes [8], so it is conceivable that these methodologies can be adapted to ME.

Registration-Based ME. The notion of Registration-Based Encryption (RBE) was introduced in 2018 by Garg et al. [9], with the goal of mitigating the key escrow, by substituting the trusted authority with a weaker entity called key curator who has no knowledge of any secret key: each party can generate its own secret key and publicly register its identity and the corresponding public key to the key curator. Notably, RBE schemes can be constructed from standard assumptions. It would be very interesting to construct a Registration-Based ME scheme since key escrow is the main obstacle which restricts applicability in many scenarios.

5.4 Blackbox constructions from ABE

Last but not least, it would be significant to propose black-box constructions from ABE schemes: in particular, analyze how efficiently one can instantiate a Matchmaking Encryption scheme from already existent ABE systems and under which assumptions. This would allow to better understand the relationship between ABE and ME.

Bibliography

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

[2] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE S&P*, pages 180–196, 2003.

[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.

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

[6] Gene Itkis, Emily Shen, Mayank Varia, David Wilson, and Arkady Yerukhimovich. Bounded-collusion attribute-based encryption from minimal assumptions. In Serge Fehr, editor, *Public-Key Cryptography – PKC 2017*, pages 67–87, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[7] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 445–464, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] Y. Wang, X. Liu, L. Liang, W. Feng, and G. Yang. Mitigating key escrow in attribute-based encryption. *International Journal of Network Security*, 17:94–102, 01 2015.

[9] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. *Registration-Based Encryption: Removing Private-Key Generator from IBE: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part I*, pages 689–718. 11 2018.