Match Me if You Can: Matchmaking Encryption and its Applications

Giuseppe Ateniese[*1], Danilo Francati[†1], David Nuñez[‡2], and Daniele Venturi[§3]

[1] *Stevens Institute of Technology, USA*
[2] *NuCypher*
[3] *Sapienza University of Rome, Italy*

February 14, 2019

Abstract

We introduce a new form of encryption that we name *matchmaking encryption* (ME). Using ME, sender S and receiver R (each with its own attributes) can both specify policies the other party must satisfy in order for the message to be revealed. The main security guarantee is that of privacy-preserving policy matching: During decryption nothing is leaked beyond the fact that a match occurred/did not occur.

ME opens up new ways of secretly communicating, and enables several new applications where both participants can specify fine-grained access policies to encrypted data. For instance, in social matchmaking, S can encrypt a file containing his/her personal details and specify a policy so that the file can be decrypted only by his/her ideal partner. On the other end, a receiver R will be able to decrypt the file only if S corresponds to his/her ideal partner defined through a policy.

On the theoretical side, we define security for ME, as well as provide generic frameworks for constructing ME from functional encryption.
These constructions need to face the technical challenge of simultaneously checking the policies chosen by S and R, to avoid any leakage.

On the practical side, we construct an efficient scheme for the identity-based setting, with provable security in the random oracle model under the standard BDH assumption. We implement and evaluate our scheme and provide experimental evidence that our construction is practical. We also apply identity-based ME to a concrete use case, in particular for creating an anonymous bulletin board over a Tor network.

Keywords. Secret handshake, attribute-based encryption, social matchmaking, Tor.

[*] gatenies@stevens.edu
[†] dfrancat@stevens.edu
[‡] david@nucypher.com
[§] venturi@di.uniroma1.it

1 Introduction

Intelligence operations often require secret agents to communicate with other agents from different organizations. When two spies meet to exchange secrets, they use a type of secret handshake to ensure that the parties participating in the exchange are the ones intended. For example, an FBI agent may want to communicate only with CIA agents, and if this is not the case, the communication should drop without revealing membership information and why the communication failed. This form of *live drop* communication,[1] when parties are online and interact, has been implemented in cryptography and it is referred to as secret handshake (SH) protocol [9]. In SH, two parties agree on the same secret key only if they are both from the same group. Privacy is preserved in the sense that, if the handshake fails, nobody learns anything relevant other than the participants are not in the same group. In SH with dynamic matching [5], groups and roles can even be determined just before the protocol execution.

SH can be thought of as an evolution of traditional key exchange protocols, where protecting privacy of the participants assumes an essential role. As any other key agreement protocol, SH is inherently interactive and its purpose is for the parties to converge on a secret key. A natural question is whether there exists a non-interactive version of SH, in a similar way as ElGamal public-key encryption can be interpreted as a non-interactive version of the classical Diffie-Hellman key exchange. This new cryptographic primitive would allow senders to encrypt messages offline given only the public key of the receiver, thus getting rid of real-time interactions, while at the same time providing strong privacy guarantees for time-delayed communications such as email. Non-interactivity mitigates or prevents *traffic analysis* which affects all SH protocols when deployed within a network environment (see, e.g., [5]). In particular, increased traffic between nodes may signal to an adversary that the SH protocol was successful, even though the nodes' group affiliations and roles remain private.

Non-interactive SH is even more relevant if we consider that the most common method of espionage tradecraft is the *dead drop* one,[1] which maintains operational security by using a secret location for the exchange of information, thus relieving the agents from meeting in person. Unfortunately, dead-drop communication cannot be captured by any existing cryptographic primitive, since it requires a form of expressiveness that is not currently provided by encryption and its more advanced forms.

Matchmaking encryption. In this paper, we are revamping the encryption primitive and introducing a new concept termed *"Matchmaking Encryption"*, or ME. In ME, 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 (i.e., the sender's attributes match the receiver's policy and vice-versa). Nothing beyond that is leaked; furthermore, the sender's attributes are certified by the authority, so that no malicious sender can forge a valid ciphertext which embeds fake attributes.

For instance, the sender, during encryption, can specify that the message is intended for an FBI agent that lives in NYC. The receiver, during decryption, can also specify that he wants to read messages only if they come from CIA agents. If any of these two policies is not satisfied, the message remains secret, but nobody learns which policy failed. In this vein, ME can be seen as a non-interactive version of SH, but with much more enhanced functionality. Indeed, an SH works only for groups and roles, while attribute-based key agreements [24] do not consider

[1] See `https://en.wikipedia.org/wiki/Dead_drop`.

privacy. We refer the reader to §1.3 for a comparison between ME and other primitives in the realm of attribute-based cryptography.

Other killer applications of ME are those where the receiver must be sheltered from the actual content of messages to avoid liability, inconvenience or inappropriateness. ME naturally tackles *social matchmaking* confidentiality, where potential partners open files intended for them but only if they contain the traits of the desired person; if decryption fails, nobody learns why, so that privacy is preserved. Encrypting bids (or votes) under ME provides an exciting twist to well-studied problems. Bidders send private bids to a collector and specify the conditions under which the encryption should be opened. The collector opens only the bids that match specific requirements. If decryption fails, the collector does not learn why, and the actual bid (or vote) remain sealed. ME avoids exposing information connected to unlooked-for bids which could influence the receiver and adversely affect the bidding process outcome.

ME also supports marginalized and dissident communities in authoritarian countries. It can act as an enabler for journalists, political activists and minorities in free-speech technical applications such as SecurePost ([39]) that provides verified group anonymity. Indeed, in their thorough study [39], the authors reveal that, in authoritarian countries, anonymous communication may not be credible and cannot be trusted since sources are unknown.[2] ME provides a comprehensive technical solution for censorship-resistant communication while providing source authenticity and strong privacy guarantees that cannot be obtained with existing tools. For instance, the ability to check ciphertexts against a policy before decryption allows journalists or activists to vet messages and avoid exposure to unwanted information that would make them liable. To this end, in Section §6, we introduce and implement a privacy-preserving bulletin board that combines Tor hidden services with ME to allow parties to collect information from anonymous but authentic sources.

1.1 Our Contributions

We initiate a systematic study of ME, both in terms of definitions and constructions. Our main contributions are summarized below.

Syntax of ME. In ME, a trusted authority publishes a master public key mpk, associated to a master secret key msk. The master secret key msk is used by the authority to generate three types of keys: (i) An encryption key ek_σ, associated with attributes σ for the sender (created using an algorithm SKGen); (ii) A decryption key dk_ρ, associated with attributes ρ for the receiver (created using an algorithm RKGen); (iii) A decryption key $\mathsf{dk}_\mathbb{S}$, associated to a policy $\mathbb{S}$ that the sender's attributes should satisfy, but that is chosen by the receiver (created using an algorithm PolGen).

A sender with attributes σ, and corresponding encryption key ek_σ obtained from the authority, can encrypt a plaintext m by additionally specifying a policy $\mathbb{R}$ (chosen on the fly), thus yielding a ciphertext c that is associated with both σ and $\mathbb{R}$. Finally, the receiver can attempt to decrypt c using keys dk_ρ and $\mathsf{dk}_\mathbb{S}$: In case of a match (i.e., the attributes of both parties satisfy the counterparty's policy), the receiver obtains the plaintext, and otherwise an error occurs.

Security of ME. We consider two properties termed *privacy*, and *authenticity*. On rough terms, privacy looks at the secrecy of the sender w.r.t. the plaintext m, the chosen policy $\mathbb{R}$, and its attributes σ, whenever a malicious receiver, possessing decryption keys for several attributes ρ and policies $\mathbb{S}$:

[2] See `https://www.news.ucsb.edu/2019/019308/anonymous-yet-trustworthy`

- Can't decrypt the ciphertext ("mismatch condition"), i.e., either the sender's attributes do not satisfy the policies held by the receiver ($\mathbb{S}(\sigma) = 0$), or the receiver's attributes do not satisfy the policy specified by the sender ($\mathbb{R}(\rho) = 0$).
- Can decrypt the ciphertext ("match condition"), i.e., both the sender's and the receiver's attributes satisfy the corresponding policy specified by the counterpart ($\mathbb{R}(\rho) = 1$ and $\mathbb{S}(\sigma) = 1$). Of course, in such a case the receiver is allowed to learn the plaintext.

On the other hand, authenticity says that an attacker not possessing attributes σ should not be able to create a valid ciphertext (i.e., a ciphertext not decrypting to $\perp$) w.r.t. any access policy that is satisfied by σ.

Black-box constructions. It turned out that building matchmaking encryption is quite difficult. While a compiler turning key agreement into public-key encryption exists (e.g., Diffie-Hellman key exchange into ElGamal public-key encryption), there is no obvious way of building ME from SH, even by extending the model of SH to include attributes and policies in order to achieve something akin to attribute-based key agreement protocols. The main technical challenge is to ensure that the policies established by the sender and receiver are simultaneously checked to avoid any leakage. This simultaneity requirement is so elusive that even constructions combining attribute-based encryption (ABE) with authentication mechanisms fail to achieve it (more on this later).

Our first technical contribution is a construction of an ME for arbitrary policies based on three tools: (i) an FE scheme for randomized functionalities [1] (rFE), (ii) digital signatures, and (iii) non-interactive zero-knowledge (NIZK) proofs. When using the rFE scheme from [1], we can instantiate our scheme assuming the existence of *either* semantically secure public-key encryption schemes and low-depth pseudorandom generators, *or* concrete assumptions on multi-linear maps, *or* polynomially-secure indistinguishability obfuscation (iO).

This construction satisfies only security against bounded collusions, where there is an a-priori upper bound on the number of queries a malicious receiver can make to oracles RKGen and PolGen. We additionally give a simpler construction of ME for arbitrary policies that even achieves full security (i.e., security against unbounded collusions), albeit under stronger assumptions. In particular, we replace rFE with 2-input functional encryption (2FE) [23]. When using the 2FE scheme by Goldwasser *et al.* [23], we can instantiate this construction based on sub-exponentially secure iO.

Being based on strong assumptions, the above constructions should be mainly understood as feasibility results showing the possibility of constructing ME for arbitrary policies. It is nevertheless worth pointing out a recent construction of iO based on LWE, bilinear maps, and weak pseudorandomness [4], which avoids multi-linear maps. Additionally, Fisch *et al.* [20] show how to implement efficiently FE and 2FE using Intel's Software Guard Extensions (SGX), a set of processors allowing for the creation of isolated execution environments called *enclaves*. At a high level, in their practical implementation, a functional decryption key sk_f consists of a signature on the function f, while messages are encrypted using standard PKE. In order to run the decryption algorithm, a client sends sk_f together with ciphertext c to a *decryption enclave*, which first checks if the signature is valid (i.e., the function evaluation has been authorized by the authority), and if so it decrypts c by using the corresponding secret key, and outputs the function f evaluated on the plaintext. Lastly, the enclave erases its memory. This approach can be applied directly to FE, 2FE, and even rFE for arbitrary functionalities, which, thanks to our results, makes ME for arbitrary policies practical in the trusted hardware setting.

The identity-based setting. Next, we turn to the natural question of obtaining efficient ME in restricted settings. In particular, we focus on the identity-based setting where access policies are simply bit-strings representing identities (as for standard identity-based encryption). This yields identity-based ME (IB-ME). For this setting, we provide an efficient construction that we prove secure in the random oracle model (ROM), based on the standard bilinear Diffie-Hellman assumption (BDH) over bilinear groups.

Recall that in ME the receiver needs to obtain from the authority a different key for each access policy $\mathbb{S}$. While this requirement is perfectly reasonable in the general case, where the policy might consist of the conjunction of several attributes, in the identity-based setting a receiver that wants to receive messages from several sources must obtain one key for each source. As this would not scale well in practice, we change the syntax of IB-ME and remove the PolGen algorithm. In particular, the receiver can now specify on the fly an identity string snd (playing the role of the access policy $\mathbb{S}$) that is directly input to the decryption algorithm (together with the secret key associated to the receiver's identity).

While the above modification yields much more efficient IB-ME schemes, it comes with the drawback that an adversary in the privacy game can try to unlock a given ciphertext using different target identities snd chosen on the fly. The latter yields simple attacks that required us to tweak the definition of privacy in the identity-based setting slightly. We refer the reader to §5 for more details.

Concrete use case and implementation. We give evidence of the practical viability of our IB-ME construction by providing a prototype implementation in Python. Our experimental evaluation can be found in §6. There, we also detail a concrete use case where IB-ME is used in order to realize a prototype of a new privacy-preserving bulletin board that is run on the Tor network [47]. Our system allows parties to communicate privately, or entities such as newspapers or organizations to collect information from anonymous sources.

A public bulletin board is essentially a broadcast channel with memory. Messages can be encrypted under ME so that their content is revealed only in case of a policy match. The privacy-preserving feature of ME ensures that, if decryption fails, nobody learns which policies were not satisfied. This effectively creates secure and private virtual rooms or sub-channels.

Arranged ME. We also consider an alternative flavor of ME, called *arranged* matchmaking encryption (A-ME), where there is a single decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$ that describes simultaneously the receiver's attributes ρ and the policy $\mathbb{S}$ chosen by the receiver. Thus, an A-ME scheme does not come with a PolGen algorithm. This feature makes sense in applications where a receiver could have multiple attributes, but with restricted access rights. A-ME is simpler to construct, in fact we show how to obtain A-ME for arbitrary policies from FE for deterministic functionalities, digital signatures, and NIZK proofs.

See Tab. 1 for a summary of our constructions in terms of assumptions and for different flavors of ME.

1.2 Technical Approach

Below, we describe the main ideas behind our constructions of ME. We start by presenting two unsuccessful attempts, naturally leading to our secure constructions. Both attempts are based on FE. Recall that FE allows us to generate decryption keys dk_f associated to a functionality f, in such a way that decrypting a ciphertext c, with underlying plaintext x, under dk_f, yields $f(x)$ (and nothing more). Note that FE implies both ciphertext-policy ABE [13] (CP-ABE) and key-policy ABE [27] (KP-ABE).

	Type	Privacy	Authenticity	Assumptions
§4	ME	✓‡	✓‡	rFE + Signatures + NIZK
§5	IB-ME	✓†	✓†	BDH (RO model)
§4.2	ME	✓	✓	2FE + Signatures + NIZK
§3.2	A-ME	✓	✓	FE + Signatures + NIZK

Table 1: Results achieved in this work. † Security only holds in the identity-based setting. ‡ Security only holds in case of bounded collusions.

First attempt. A first natural approach would be to construct an ME scheme by combining two distinct FE schemes. The idea is to apply sequentially two functionalities f^1 and f^2, where the first functionality checks whether the sender's policy $\mathbb{R}$ is satisfied, whereas the second functionality checks whether the receiver's policy $\mathbb{S}$ is satisfied. More in details, let f^1 and f^2 be the following functions:

$$f^1_\rho(\mathbb{R}, c) = \begin{cases} c, & \text{if } \mathbb{R}(\rho) = 1 \\ \perp, & \text{otherwise} \end{cases} \qquad f^2_{\mathbb{S}}(\sigma, m) = \begin{cases} m, & \text{if } \mathbb{S}(\sigma) = 1 \\ \perp, & \text{otherwise} \end{cases}$$

where $\mathbb{R}(\rho) = 1$ (resp. $\mathbb{S}(\sigma) = 1$) means that receiver's attributes ρ (resp. sender's attributes σ) satisfy the sender's policy $\mathbb{R}$ (resp. receiver's policy $\mathbb{S}$). A sender now encrypts a message m under attributes σ by first encrypting (σ, m) under the second FE scheme, and thus it encrypts the corresponding ciphertext concatenated with the policy $\mathbb{R}$ under the first FE scheme. The receiver first decrypts a ciphertext using secret key dk_ρ associated with function f^1_ρ, and then it decrypts the obtained value using secret key $\mathsf{dk}_{\mathbb{S}}$ associated with function $f^2_{\mathbb{S}}$.

While "semantic security" of the underlying FE schemes computationally hides the plaintext of the resulting ME scheme, privacy is not guaranteed completely: In fact, when the first encrypted layer decrypts correctly (resp. does not decrypt correctly), a receiver infers that the sender's attributes σ match (resp. do not match) the policy $\mathbb{S}$.

Second attempt. One could think to savage the above construction as follows. Each function f^i returns a random key r_i in case the corresponding policy (i.e., the policy checked by function f^i) is satisfied, and otherwise it returns a random value generated by running a secure PRF F. Both partial keys r_1, r_2 are then needed to unmask the string $r_1 \oplus r_2 \oplus m$, which is included in the ciphertext.

More precisely, consider functions $f^1_\rho(\mathbb{R}, r_1, k_1)$ and $f^2_{\mathbb{S}}(\sigma, r_2, k_2)$, such that $f^1_\rho(\mathbb{R}, r_1, k_1)$ (resp. $f^2_{\mathbb{S}}(\sigma, r_2, k_2)$) returns r_1 (resp. r_2) if ρ satisfies $\mathbb{R}$ (resp. σ satisfies $\mathbb{S}$); otherwise, it returns $F_{k_1}(\rho)$ (resp. $F_{k_2}(\mathbb{S})$), where k_1 (resp. k_2) is a key for the PRF F. An encryption of message m w.r.t. attributes σ and policy $\mathbb{R}$ would now consist of three values (c_1, c_2, c_3), where c_1 is an encryption of $(\mathbb{R}, r_1, k_1)$ under the first FE scheme, c_2 is an encryption of (σ, r_2, k_2) under the second FE scheme, and finally $c_3 = r_1 \oplus r_2 \oplus m$. A receiver (with keys dk_ρ and $\mathsf{dk}_{\mathbb{S}}$ associated to functions f^1_ρ and $f^2_{\mathbb{S}}$ as before) would decrypt c_1 and c_2 using dk_ρ and $\mathsf{dk}_{\mathbb{S}}$, and finally xor the outputs between them and with c_3.

As before, "semantic security" still follows from the security of the two FE schemes. Furthermore, it might seem that privacy is also satisfied because, by security of the PRF, it is hard to distinguish whether the decryption of each c_i yields the random string r_i (i.e., there was a match) or an output of F_{k_i} (i.e., there was no match). However, a malicious receiver possessing distinct attributes ρ and ρ', such that both satisfy the policy $\mathbb{R}$, is able to figure out whether the sender's policy is matched by simply decrypting c_1 twice (using attributes ρ and ρ') and comparing if the decryption returns twice the same value (i.e., r_1). A similar attack can be carried out using two different keys for distinct policies $\mathbb{S}$ and $\mathbb{S}'$, such that both policies are satisfied by the attributes σ.

ME from 2FE. Intuitively, in order to avoid the above attacks, we need to check simultaneously that $\mathbb{S}(\sigma) = 1$ and $\mathbb{R}(\rho) = 1$. 2FE comes handy to solve this problem, at least if one is willing to give up on authenticity. Recall that in a 2FE scheme we can associate secret keys with 2-ary functionalities, in such a way that decrypting ciphertexts c_0, c_1 computed using independent keys $\mathsf{ek}_0, \mathsf{ek}_1$, and corresponding to plaintexts x_0, x_1, yields $f(x_0, x_1)$ (and nothing more).

Wlog., we reserve the 1st slot to the sender, while the 2nd slot is reserved to the receiver; the administrator gives the key ek_0 to the sender. The sender now encrypts a message m under attributes σ and policy $\mathbb{R}$ by computing $\mathsf{Enc}(\mathsf{ek}_0, (\sigma, \mathbb{R}, m))$, which yields a ciphertext c_0 for the first input of the function f. The receiver, as usual, has a pair of decryption keys dk_ρ, $\mathsf{dk}_\mathbb{S}$ obtained from the administrator; here, $\mathsf{dk}_\mathbb{S} = \mathsf{Enc}(\mathsf{ek}_1, \mathbb{S}) = c_1$ is an encryption of $\mathbb{S}$ under key ek_1. Hence, the receiver runs $\mathsf{Dec}(\mathsf{dk}_\rho, c_0, c_1)$, where dk_ρ is associated to the function $f_\rho((m, \sigma, \mathbb{R}), \mathbb{S})$ that returns m if and only if both $\mathbb{R}(\rho) = 1$ and $\mathbb{S}(\sigma) = 1$ (i.e., a match occurs).

On rough terms, privacy follows by the security of the underlying 2FE scheme, which guarantees that the receiver learns nothing more than the output of f. Unfortunately, this construction does not immediately satisfy authenticity. To overcome this limitation, we tweak it as follows. First, we let the sender obtain from the authority a signature s on its own attributes σ; the signature is computed w.r.t. a verification key that is included in the public parameters of the scheme. Second, during encryption, the sender computes the ciphertext c_0 as above, but now additionally proves in zero knowledge that it knows a valid signature for the attributes that are hidden in the ciphertext. As we show, this modification allows us to prove authenticity, while at the same time preserving privacy. We refer the reader to §4.2 for the formal proof.

ME from rFE. In §4, we give an alternative solution that combines rFE and FE (and thus can be instantiated from weaker assumptions). Recall that rFE is a generalization of FE that supports randomized functionalities. In what follows, we write f^1 for the randomized functionality supported by the rFE scheme, and f^2 for the deterministic functionality supported by the plain FE scheme. The main idea is to let the sender encrypt $(m, \sigma, \mathbb{R})$ under the rFE scheme. We then consider the randomized function f^1_ρ that checks if ρ satisfies $\mathbb{R}$: In case a match occurs (resp. does not occur), it returns an encryption of (m, σ) (resp. of $(\perp, \perp)$, where $\perp$ denotes garbage) for the second function $f^2_\mathbb{S}$ that simply checks whether the policy $\mathbb{S}$ is satisfied or not. The receiver decryption keys are the keys $\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}$ associated to the functions f^1_ρ and $f^2_\mathbb{S}$.

Roughly speaking, since the randomized function f^1 passes encrypted data to f^2, a malicious receiver infers nothing about the satisfiability of policy $\mathbb{R}$. On the other hand, the satisfiability of $\mathbb{S}$ remains hidden, as long as the FE scheme for the function f^2 is secure. However, the actual security proof is much more subtle.

While the above construction does not directly satisfy authenticity, we can show that the same trick explained above for the 2FE-based scheme works here as well.

A-ME from FE. Recall that the difference between ME and A-ME lies in the number of decryption keys: While in ME there are two distinct keys (one for the policy $\mathbb{S}$, and one for the attributes ρ), in A-ME there is a single decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$ that represents both the receiver's attributes ρ and the policy $\mathbb{S}$.

As a result, looking at our construction of ME from 2FE, we can now hard-code the policy $\mathbb{S}$ (together with the attributes ρ) into the function, which allows us to replace 2FE with plain FE. This way, each A-ME decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$ is the secret key corresponding to the function $f_{\rho,\mathbb{S}}$ for the FE scheme. The security proof, which appears in §4.3, only requires FE with game-based security [13], which in turn can be instantiated under much weaker assumptions.

IB-ME. Above, we mentioned that the natural construction of ME where a ciphertext masks the plaintext m with two distinct pads r_1, r_2—where r_1, r_2 are re-computable by the receiver as long as a match occurs—is insecure. This is because the expressiveness of ME allows us to have two distinct attributes ρ and ρ' (resp. two distinct policies $\mathbb{S}$ and $\mathbb{S}'$) such that both satisfy the sender's policy $\mathbb{R}$ (resp. both are satisfied by the sender's attributes σ).

The main idea behind our construction of IB-ME (cf. §5) under the BDH assumption is that the above attack does not work in the identity-based setting, where each receiver's policy $\mathbb{S}$ (resp. receiver's policy $\mathbb{R}$) is satisfied only by the attribute $\sigma = \mathbb{S}$ (resp. $\rho = \mathbb{R}$). This means that an encryption $m \oplus r_1 \oplus r_2$ yields an efficient IB-ME as long as the random pad r_2 (resp. r_1) can be re-computed by the receiver if and only if its policy $\mathbb{S}$ is satisfied (resp. its attributes ρ satisfy the sender's policy). On the other hand, if $\mathbb{S}$ is not satisfied (resp. ρ does not satisfy the sender's policy), the receiver obtains a pad r_2' (resp. r_1') that is completely unrelated to the real r_2 (resp. r_1). In our scheme, we achieve the latter by following a similar strategy as in the Boneh-Franklin IBE construction [12].

1.3 Related Work

Secret handshakes. Introduced by Balfanz *et al.* [9], an SH 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 (e.g., role) that the other party should have.

SH preserves the privacy of the participants, meaning that when the handshake is successful they only learn that they both belong to the same group (yet, their identities remain secret), whereas they learn nothing if the handshake fails. Subsequent work in the area [30, 46, 5, 14, 50, 53, 33, 51, 32, 31, 45] focused on improving on various aspects of SH, including members' privacy and expressiveness of the matching policies (i.e., attribute-based SH).

In this vein, ME can be thought of as a *non-interactive* SH. Indeed, ME 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.

Attribute-based encryption. The concept of ABE was first proposed by Sahai and Waters [44] in the setting of fuzzy identity-based encryption, where users are identified by a single attribute (or identity string), and policies consist of a single threshold gate. Afterwards, Bethencourt *et al.* [11] generalized this idea to the case where users are described by multiple attributes. Their ABE scheme is a CP-ABE, i.e., a policy is embedded into the ciphertext, whereas the attributes are embedded into the receiver's decryption keys. The first CP-ABE with non-monotonic access structures was proposed by Ostrovsky *et al.* [41]. Goyal *et al.* [27], instead, introduced KP-ABE, where ciphertexts contain the attributes, whereas the policy is embedded in the decryption keys. Several other CP-ABE and KP-ABE schemes have been proposed in the litterature, see, among others, [17, 26, 52, 38, 29, 37, 57, 15, 16, 55, 34, 7, 42, 28, 54, 56, 40, 8].

In ABE, only one party can specify a policy, and thus only one entity has the power to select the source (or the destination) of an encrypted message. Motivated by this limitation, Attrapadung and Imai [6] 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 that describes both the receiver's policy and attributes. Similarly to ME, if both policies are satisfied by the respective counterpart, the message is revealed.

Dual-policy ABE and ME differ in several aspects. First, on the syntactical level, in ME there are two distinct decryption keys: One for the attributes and one for the policy specified by the receiver. This yields improved flexibility, as receivers are allowed to choose attributes and policies independently. (Indeed, the syntax of dual-policy ABE is more similar to that of A-ME.) Second, on the security level, both ME and A-ME provide much stronger privacy guarantees than dual-policy ABE. In fact, the security definition for dual-policy ABE only protects the secrecy of the plaintext. Additionally, the actual constructions in [6, 8] are easily seen not to preserve privacy w.r.t. the sender's attributes/policy whenever a match does not occur. Intuitively, this is because the procedure that checks, during decryption, whether a match occurred or not, is not an atomic operation. Also note that dual-policy ABE does not directly provide authenticity, which instead is a crucial property for ME and A-ME (those being a type of non-interactive SH).

Attribute-based key exchange. Gorantla *et al.* [24] introduced attribute-based authenticated key exchange (AB-AKE). This is essentially an interactive protocol which allows sharing a secret key between parties whose attributes satisfy a fixed access policy. Note that the policy must be the same for all the parties, and thus it must, e.g., be negotiated before running the protocol.

In a different work, Kolesnikov *et al.* [36] built a different AB-KE without bilateral authentication. In their setting, a client with some attributes (certificated by an authority) wants to authenticate himself to a server according to a fixed policy. The server will share a secret key with the client if and only if the client's attributes satisfy the server's policy.

Note that in ME both senders and receivers can choose their own policies, a feature not present in attribute-based key exchange protocols.

Access control encryption. Access control encryption (ACE) [19, 35, 21, 48] is a novel type of encryption that allows fine-grained control over information flow. The actors are a set of senders, a set of receivers, and a sanitizer. The goal is to enforce *no-read* and *no-write* rules (described by a policy) over the communication, according to the sender's and receiver's identities.

The flow enforcement is done by the sanitizer, that applies a randomized algorithm to the incoming ciphertexts. The result is that only receivers allowed to communicate with the source will be able to decrypt the sanitized ciphertext correctly, obtaining the original message (*no-read* rule). On the other hand, if the source has not the rights to communicate with a target receiver (e.g., the sender is malicious), then the latter will receive a sanitized ciphertext that looks like an encryption of a random message (*no-write* rule).

ACE and ME accomplish orthogonal needs: The former enables cryptographic control over information flow within a system, whereas the latter enables both the sender and the receiver to specify fine-grained access rights on encrypted data. Furthermore, ACE inherently requires the presence of a trusted sanitizer, whereas ME involves no additional actor (besides the sender and the receiver).

2 Preliminaries

2.1 Notation

We use the notation $[n] \overset{\text{def}}{=} \{1, \ldots, n\}$. 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 A 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.

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 paper, 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.

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 Functional Encryption

2.3.1 Functional Encryption for Randomized Functionalities

A functional encryption scheme for randomized functionalities [25] (rFE) $f:\mathcal{K}\times\mathcal{X}\times\mathcal{R}\to\mathcal{Y}$ consists of the following polynomial-time algorithms.[3]

$\mathsf{Setup}(1^\lambda)$**:** Upon input the security parameter, the randomized setup algorithm outputs a master public key mpk and a master secret key msk.

$\mathsf{KGen}(\mathsf{msk},k)$**:** The randomized key generation algorithm takes as input the master secret key msk and an index $k\in\mathcal{K}$, and outputs a secret key sk_k for f_k.

$\mathsf{Enc}(\mathsf{mpk},x)$**:** The randomized encryption algorithm takes as input the master public key mpk, an input $x\in\mathcal{X}$, and returns a ciphertext c_x.

$\mathsf{Dec}(\mathsf{sk}_k,c_x)$**:** The deterministic decryption algorithm takes as input a secret key sk_k and a ciphertext c_x, and returns a value $y\in\mathcal{Y}$.

Correctness of rFE intuitively says that decrypting an encryption of $x\in\mathcal{X}$ using a secret key sk_k for function f_k yields $f_k(x;r)$, where $r \leftarrow\!\!\$\, \mathcal{R}$. Since $f_k(x)$ is a random variable, the actual definition requires that whenever the decryption algorithm is invoked on a fresh encryption of a message x under a fresh key for f_k, the resulting output is computationally indistinguishable to $f_k(x)$.

Definition 3 (Correctness of rFE)**.** *A rFE scheme* $\Pi=(\mathsf{Setup},\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$ *for a randomized functionality* $f:\mathcal{K}\times\mathcal{X}\times\mathcal{R}\to\mathcal{Y}$ *is correct if the following distributions are computationally indistinguishable:*

$$\{\mathsf{Dec}(\mathsf{sk}_{k_j},c_i)\}_{k_j\in\mathcal{K},x_i\in\mathcal{X}} \qquad \{f_{k_j}(x_i;r_{i,j})\}_{k_j\in\mathcal{K},x_i\in\mathcal{X}}$$

where $(\mathsf{mpk},\mathsf{msk}) \leftarrow\!\!\$\, \mathsf{Setup}(1^\lambda)$*,* $\mathsf{sk}_{k_j} \leftarrow\!\!\$\, \mathsf{KGen}(\mathsf{msk},k_j)$ *for* $k_j\in\mathcal{K}$*,* $c_i \leftarrow\!\!\$\, \mathsf{Enc}(\mathsf{mpk},x_i)$ *for* $x_i\in\mathcal{X}$*, and* $r_{i,j} \leftarrow\!\!\$\, \mathcal{R}$*.*

As for security, the setting of rFE tackles malicious encryptors. However, for our purpose, it will be sufficient to consider a weaker security guarantee that only holds for honest encryptors. In this spirit, the definition below is adapted from [1, Definition 3.3] for the special case of honest encryptors.

Definition 4 ((q_1,q_c,q_2)-NA-SIM-security of rFE)**.** *A rFE scheme* $\Pi=(\mathsf{Setup},\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$ *for a randomized functionality* $f:\mathcal{K}\times\mathcal{X}\times\mathcal{R}\to\mathcal{Y}$ *is* (q_1,q_c,q_2)*-NA-SIM-secure if there exists an efficient (stateful) simulator* $\mathsf{S}=(\mathsf{S}_1,\mathsf{S}_2,\mathsf{S}_3,\mathsf{S}_4)$ *such that for all PPT adversaries* $\mathsf{A}=(\mathsf{A}_1,\mathsf{A}_2)$ *where* A_1 *makes at most* q_1 *key generation queries and* A_2 *makes at most* q_2 *key generation query, the output of the following two experiments are computationally indistinguishable:*

[3] Often, and equivalently, FE schemes are parameterized by a function ensemble $\mathcal{F}=\{f_k:\mathcal{X}\times\mathcal{R}\to\mathcal{Y}\}_{k\in\mathcal{K}}$.

```
$\mathbf{REAL}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(x^*, \alpha) \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{O}_1(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$
    $\textit{where } x^* = (x_0, \ldots, x_{q_c})$
$c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, x_i)\ \textit{for } i \in [q_c]$
$\mathsf{out} \leftarrow\!\!\$\ \mathsf{A}_2^{\mathsf{O}_2(\mathsf{msk},\cdot)}(1^\lambda, \{c_i\}, \alpha)$
$\mathbf{return}\ (x, \{k\}, \mathsf{out})$
```

```
$\mathbf{IDEAL}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \alpha') \leftarrow\!\!\$\ \mathsf{S}_1(1^\lambda)$
$(x^*, \alpha) \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{O}'_1(\alpha',\cdot)}(1^\lambda, \mathsf{mpk})$
    $\textit{where } x^* = (x_0, \ldots, x_{q_c})$
$\textit{Let } \{k_1, \ldots, k_{q_1}\} = \mathcal{Q}_{\mathsf{O}'_1}$
$\textit{For } i \in [q_c], j \in [q_1]$
    $y_{i,j} = f_{k_j}(x_i; r_{i,j}),\ \textit{where } r_{i,j} \leftarrow\!\!\$\ \mathcal{R}$
$(\{c_i\}, \alpha') \leftarrow\!\!\$\ \mathsf{S}_3(\alpha', \{y_{i,j}\})$
$\mathsf{out} \leftarrow\!\!\$\ \mathsf{A}_2^{\mathsf{O}'_2(\alpha',\cdot)}(1^\lambda, \{c_i\}, \alpha)$
$\mathbf{return}\ (x, \{k'\}, \mathsf{out})$
```

where the key generation oracles are defined in the following way:

$\mathsf{O}_1(\mathsf{msk}, \cdot)$ **and** $\mathsf{O}_2(\mathsf{msk}, \cdot)$**:** *Are implemented with the algorithm* $\mathsf{KGen}(\mathsf{msk}, \cdot)$*. The ordered set* $\{k\}$ *is composed of the queries made to oracles* O_1 *and* O_2*.*

$\mathsf{O}'_1(\mathsf{st}', \cdot)$ **and** $\mathsf{O}'_2(\mathsf{st}', \cdot)$**:** *Are implemented with two simulators* $\mathsf{S}_2(\alpha', \cdot)$*,* $\mathsf{S}_4(\alpha', \cdot)$*. The simulator* S_4 *is given oracle access to* $\mathsf{KeyIdeal}(x^*, \cdot)$*, which, on input* k*, outputs* $f_k(x_i; r)$*, where* $r \leftarrow\!\!\$\ \mathcal{R}$ *for every* $x_i \in x^*$*. The ordered set* $\{k'\}$ *is composed of the queries made to oracles* O'_1 *and the queries made by* S_4 *to* $\mathsf{KeyIdeal}$*.*

2.3.2 Functional Encryption for Deterministic Functionalities

Functional encryption (FE) for deterministic functionalities $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ can be cast as a special case of rFE. Since f is a deterministic functionality, correctness now simply says that whenever the decryption algorithm is invoked on a fresh encryption of a message x under a fresh key for f, the resulting output equals $f_k(x)$.

Definition 5 (Correctness of FE)**.** *A functional encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a functionality* $f : \mathcal{K} \times \mathcal{X} \to \rho$ *is correct if* $\forall x \in \mathcal{X}, \forall k \in \mathcal{K}$*, the following holds:*

$$\mathbb{P}\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_k \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, k), \\ \mathsf{Dec}(\mathsf{sk}_k, \mathsf{Enc}(\mathsf{mpk}, x)) = f_k(x) \end{array}\right] = 1$$

Definition 6 ((q_1, q_c, q_2)-SIM-security of FE)**.** *A functional encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a functionality* $f : \mathcal{K} \times \mathcal{X} \to \rho$ *is* (q_1, q_c, q_2)*-SIM-secure if there exists an efficient simulator* $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4)$ *such that for all probabilistic polynomial time adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$*, where* A_1 *makes at most* q_1 *key generation queries and* A_2 *makes at most* q_2 *key generation queries, the output of the following two experiments are computationally indistinguishable:*

```
$\mathbf{REAL}_{\Pi,\mathsf{A}}(1^\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(x^*, \alpha) \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{O}_1(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$
    $\textit{where } x^* = (x_0, \ldots, x_{q_c})$
$c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, x_i)\ \textit{for } i \in [q_c]$
$\mathsf{out} \leftarrow\!\!\$\ \mathsf{A}_2^{\mathsf{O}_2(\mathsf{msk},\cdot)}(1^\lambda, \{c_i\}, \alpha)$
$\mathbf{return}\ (x, \{k\}, \mathsf{out})$
```

```
$\mathbf{IDEAL}_{\Pi,\mathsf{A}}(1^\lambda)$
$(\mathsf{mpk}, \alpha') \leftarrow\!\!\$\ \mathsf{S}_1(1^\lambda)$
$(x^*, \alpha) \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{O}'_1(\alpha',\cdot)}(1^\lambda, \mathsf{mpk})$
    $\textit{where } x^* = (x_0, \ldots, x_{q_c})$
$\textit{Let } \{k_1, \ldots, k_{q_1}\} = \mathcal{Q}_{\mathsf{O}'_1}$
$\textit{For } i \in [q_c], j \in [q_1]$
    $y_{i,j} = f_{k_j}(x_i)$
$(\{c_i\}, \alpha') \leftarrow\!\!\$\ \mathsf{S}_3(\alpha', \{y_{i,j}\})$
$\mathsf{out} \leftarrow\!\!\$\ \mathsf{A}_2^{\mathsf{O}'_2(\alpha',\cdot)}(1^\lambda, \{c_i\}, \alpha)$
$\mathbf{return}\ (x, \{k'\}, \mathsf{out})$
```

where the key generation oracles are defined in the following way:

$\mathsf{O}_1(\mathsf{msk}, \cdot)$ **and** $\mathsf{O}_2(\mathsf{msk}, \cdot)$**:** *Are implemented with the algorithm* $\mathsf{KGen}(\mathsf{msk}, \cdot)$*. The ordered set* $\{k\}$ *is composed of the queries made to oracles* O_1 *and* O_2*.*

$\mathsf{O}'_1(\mathsf{st}', \cdot)$ **and** $\mathsf{O}'_2(\mathsf{st}', \cdot)$**:** *Are implemented with two simulators* $\mathsf{S}_2(\alpha', \cdot)$*,* $\mathsf{S}_4(\alpha', \cdot)$*. The simulator* S_4 *is given oracle access to* $\mathsf{KeyIdeal}(x^*, \cdot)$*, which on input* k*, outputs* $f_k(x_i)$ *for every* $x_i \in x^*$*. The ordered set* $\{k'\}$ *is composed of the queries made to oracles* O'_1 *and the queries made by* S_4 *to* $\mathsf{KeyIdeal}$*.*

Definition 7 (Game-based security of FE)**.** *A functional encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a functionality* $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is secure if for all valid PPT adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$*, we have:*

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

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

1. $(\mathsf{msk}, \mathsf{mpk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
2. $(m_0, m_1, \alpha) \leftarrow\$ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$.
3. $c \leftarrow\$ \mathsf{Enc}(\mathsf{mpk}, m_b)$ *where* $b \leftarrow\$ \{0, 1\}$.
4. $b' \leftarrow\$ \mathsf{A}_2^{\mathsf{KGen}(\mathsf{msk},\cdot)}(1^\lambda, c, \alpha)$.
5. *If* $b = b'$ *then output* 1*, and otherwise output* 0.

Adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *is called valid if* $\forall k \in \mathcal{Q}_{\mathsf{KGen}}$ *we have* $f_k(m_0) = f_k(m_1)$*, where* $\mathcal{Q}_{\mathsf{KGen}}$ *contains all the queries submitted by* A_1 *and* A_2 *to oracle* KGen*.*

2.3.3 Two-Input Functional Encryption

A 2-input FE (2FE) scheme for a 2-arity functionality $f : \mathcal{K} \times \mathcal{X}_0 \times \mathcal{X}_1 \to \mathcal{Y}$ consists of the following efficient algorithms.

$\mathsf{Setup}(1^\lambda)$**:** Upon input the security parameter, the randomized setup algorithm outputs 2 encryption keys $\mathsf{ek}_0, \mathsf{ek}_1$, and a master secret key msk.

$\mathsf{KGen}(\mathsf{msk}, k)$**:** The randomized key generation algorithm takes as input the master secret key msk, and an index $k \in \mathcal{K}$, and outputs a secret key sk_k for f_k.

$\mathsf{Enc}(\mathsf{ek}_i, x_i)$**:** For $i \in \{0, 1\}$, the randomized encryption algorithm takes as input the encryption key ek_i, a value $x_i \in \mathcal{X}_i$, and returns a ciphertext c_{x_i}.

$\mathsf{Dec}(\mathsf{sk}_k, c_{x_0}, c_{x_1})$**:** The deterministic decryption algorithm takes as input a secret key sk_k for f_k, and two ciphertexts c_{x_0}, c_{x_1}, and returns a value $y \in \mathcal{Y}$.

Correctness of a 2FE means that decrypting (c_{x_0}, c_{x_1}), where c_{x_i} is an encryption of x_i, using a secret key sk_k for function f_k yields $f_k(x_0, x_1)$.

Definition 8 (Correctness of 2FE)**.** *A 2FE scheme* $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a functionality* $f : \mathcal{K} \times \mathcal{X}_0 \times \mathcal{X}_1 \to \mathcal{Y}$ *is correct if* $\forall (x_0, x_1) \in \mathcal{X}_0 \times \mathcal{X}_1, \forall k \in \mathcal{K}$*:*

$$\mathbb{P}\left[\begin{array}{l}(\mathsf{ek}_0, \mathsf{ek}_1, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_k \leftarrow\$ \mathsf{KGen}(\mathsf{msk}, k), \\ c_0 \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_0, x_0), c_1 \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_1, x_1) \\ \mathsf{Dec}(\mathsf{sk}_k, c_0, c_1) = f_k(x_0, x_1)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Security changes significantly depending on which keys among $(\mathsf{ek}_0, \mathsf{ek}_1)$ are public. The flavor we require has one public and one private key, with the adversary given oracle access to the encryption algorithm for the private key. The formal definition follows below.

Definition 9 (IND-security of 2FE, $\{0,1\}$-semi-private setting)**.** *For $i \in \{0,1\}$, a 2FE scheme $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for a functionality $f : \mathcal{K} \times \mathcal{X}_0 \times \mathcal{X}_1 \to \rho$ is indistinguishably secure in the i-semi-private setting if for all valid PPT adversaries $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$:*

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

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

1. $(\mathsf{msk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
2. $((m_0^0, m_1^0), (m_0^1, m_1^1), \alpha) \leftarrow\$ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Enc}(\mathsf{ek}_i,\cdot)}(1^\lambda, \mathsf{ek}_{1-i})$.
3. $c_0 \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_0, m_0^b)$, $c_1 \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_1, m_1^b)$, *where* $b \leftarrow\$ \{0,1\}$.
4. $b' \leftarrow\$ \mathsf{A}_2^{\mathsf{KGen}(\mathsf{msk},\cdot),\mathsf{Enc}(\mathsf{ek}_i,\cdot)}(1^\lambda, (c_0, c_1), \alpha)$.
5. *If $b = b'$ then output* 1, *and otherwise output* 0.

Adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is called valid if $\forall k \in \mathcal{Q}_{\mathsf{KGen}}$, $\forall x \in \mathcal{X}_{1-i}$, $\forall x' \in \mathcal{Q}_{\mathsf{Enc}}$, we have:

$$f_k(m_0^0, m_1^0) = f_k(m_0^1, m_1^1)$$

and

$$\begin{aligned} f_k(m_0^0, x) = f_k(m_0^1, x) \text{ and } f_k(x', m_1^0) = f_k(x', m_1^1), \ \text{if } i = 0 \\ f_k(x, m_1^0) = f_k(x, m_1^1) \text{ and } f_k(m_0^0, x') = f_k(m_0^1, x'), \ \text{if } i = 1. \end{aligned}$$

The above definition is a generalization of [23, Def. 4] with parameters $(n, t, q) = (2, 1, 1)$, where the adversary is additionally given oracle access to $\mathsf{Enc}(\mathsf{ek}_i, \cdot)$; this notion is easily seen to be implied by [23, Def. 4] with parameters $(n, t, q) = (2, 2, 1)$ as the latter means that both keys $(\mathsf{ek}_0, \mathsf{ek}_1)$ can be made public. In turn, IND-secure 2FE for arbitrary functionalities in the public setting exists assuming sub-exponentially hard indistinguishability obfuscation for all functions [23].

2.4 Bilinear Diffie-Hellman Assumption

Our practical implementation of IB-ME is provably secure under the BDH assumption, which we recall below.

Definition 10 (BDH assumption)**.** *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of prime order q. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an admissible bilinear map, and let P be a generator of $\mathbb{G}$. The BDH problem is hard in $(\mathbb{G}, \mathbb{G}_T, e)$ if for every PPT adversary A:*

$$\mathbb{P}\left[\mathsf{A}(q, \mathbb{G}, \mathbb{G}_T, e, P, P^a, P^b, P^c) = e(P,P)^{abc} \right] \leq \mathsf{negl}(\lambda),$$

where $P \leftarrow\$ \mathbb{G}^*$, and $a, b, c \leftarrow\$ \mathbb{Z}_q^*$.

2.5 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. (i) The randomized algorithm I takes as input the security parameter and outputs a common reference string ω; (ii) The randomized algorithm $\mathsf{P}(\omega, (y, x))$, given $(y, x) \in R$ outputs a proof π; (iii) 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 for all $\lambda \in \mathbb{N}$, 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 11 (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 12 (Knowledge soundness). *A NIZK Π for a relation R satisfies knowledge soundness if there exists a PPT extractor $\mathsf{K} = (\mathsf{K}_0, \mathsf{K}_1)$ such that the following holds:*

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

$$\mathbb{P}\left[\begin{array}{c} \mathsf{V}(\omega, y, \pi) = 1 \wedge \\ (y, x) \notin R \end{array} : \begin{array}{c} (\omega, \xi) \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda) \\ (y, \pi) \leftarrow\!\!\$\ \mathsf{A}(\omega) \\ x \leftarrow\!\!\$\ \mathsf{K}_1(\xi, y, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

3 Matchmaking Encryption

As explained in the introduction, an ME allows both the sender and the receiver, characterized by their attributes, to choose fined-grained access policies that together describe the access rights both parties must satisfy in order for the decryption of a given ciphertext to be successful.

We present two flavors of ME. In the first, which is the standard one, the receiver's attributes and policy are independent of each other (i.e., a receiver with some given attributes can choose different policies). In the second flavor, dubbed A-ME, the receiver's attributes and policy are tighten together. We present the security model for ME and A-ME in §3.1 and §3.2.

3.1 Security Model

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 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).

Note that the decryption keys dk_ρ and $\mathsf{dk}_\mathbb{S}$ are independent, thus allowing a receiver with attributes ρ to obtain decryption keys for different policies $\mathbb{S}$. We also remark that the master policy key kpol could be considered as part of the master secret key msk, but we preferred to use distinct keys for clarity.

Correctness. The intuition for correctness is that the output of the decryption algorithm using decryption keys for receiver's attributes ρ and access policy $\mathbb{S}$, when decrypting an honestly generated ciphertext which encrypts a message m using sender's attributes σ and policy $\mathbb{R}$, 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 13 (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 $\mathsf{ek}_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}(\mathsf{msk}, \sigma)$, $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{RKGen}(\mathsf{msk}, \rho)$, $\mathsf{dk}_\mathbb{S} \leftarrow\!\!\$\ \mathsf{PolGen}(\mathsf{kpol}, \mathbb{S})$.

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

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

Figure 1: Games defining privacy and authenticity of ME. Oracles O_1, O_2, O_3 are implemented by $\mathsf{SKGen}(\mathsf{msk}, \cdot)$, $\mathsf{RKGen}(\mathsf{msk}, \cdot)$, $\mathsf{PolGen}(\mathsf{kpol}, \cdot)$.

Security. We now turn to defining security of an ME via two properties, that we dub *privacy* and *authenticity*. Intuitively, privacy aims at capturing 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 conditions: In case of a match between the sender's and receiver's attributes/policy, and in case of mismatch. This is formalized by requiring that the distributions $\mathsf{Enc}(\mathsf{ek}_{\sigma_0}, \mathbb{R}_0, m_0)$ and $\mathsf{Enc}(\mathsf{ek}_{\sigma_1}, \mathbb{R}_1, m_1)$ be 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. The actual definition requires some care, as the adversary could, e.g., obtain a decryption key for attributes ρ and policy $\mathbb{S}$ such that $\mathbb{R}_0(\rho) = 0 \vee \mathbb{S}(\sigma_0) = 0$ but $\mathbb{R}_1(\rho) = 1 \wedge \mathbb{S}(\sigma_1) = 1$, which clearly allows him to distinguish by evaluating the decryption algorithm. In order to exclude such "trivial attacks", we quantify privacy over all *valid adversaries*, as explained below:

- In case of a mismatch, i.e., when the adversary cannot decrypt the challenge ciphertext, it must be the case that for each attribute ρ and policy $\mathbb{S}$ for which the adversary knows a valid decryption key: (i) Either ρ does not satisfy policies $\mathbb{R}_0$ and $\mathbb{R}_1$; (ii) or σ_0 and σ_1 do not satisfy policy $\mathbb{S}$; (iii) or ρ does not satisfy $\mathbb{R}_0$ and σ_1 does not satisfy $\mathbb{S}$; (iv) or ρ does not satisfy $\mathbb{R}_1$ and σ_0 does not satisfy $\mathbb{S}$.

- In case of match, i.e., when the adversary can decrypt the challenge ciphertext, it must be the case that $m_0 = m_1$, and additionally, for each attribute ρ and policy $\mathbb{S}$ for which the adversary knows a valid decryption key, it holds that both: (i) ρ does not satisfy policies $\mathbb{R}_0$ and $\mathbb{R}_1$; and (ii) σ_0 and σ_1 do not satisfy policy $\mathbb{S}$.

Definition 14 (Privacy of ME)**.** *We say that an ME Π satisfies* privacy *if for all valid PPT adversaries A:*

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

where game $\mathbf{G}^{\mathsf{priv}}_{\Pi,\mathsf{A}}(\lambda)$ is depicted in Fig.1. Adversary A is called valid if $\forall \rho \in \mathcal{Q}_{\mathsf{O}_2}, \forall \mathbb{S} \in \mathcal{Q}_{\mathsf{O}_3}$ it satisfies the following invariant:

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

$$\begin{aligned}(\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho) = 0) \vee (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1) = 0)& \\ \vee (\mathbb{R}_0(\rho) = \mathbb{S}(\sigma_1) = 0) \vee (\mathbb{R}_1(\rho) = \mathbb{S}(\sigma_0) = 0);& \qquad (1)\end{aligned}$$

- ***(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.* (1) *does not hold)*

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

Note that in the above definition the challenge ciphertext is honestly computed. This is because privacy captures security against malicious receivers. Authenticity, on the other hand, demands that the only way to produce a valid ciphertext under 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.

The latter 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 $\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S}, c) \neq \bot$, and none of the encryption keys ek_σ for attributes σ (obtained by the adversary via oracle queries) satisfies the policy $\mathbb{S}$. Observe that the adversary is not given access to an encryption oracle. The reason for this is that we only consider security in the presence of chosen-plaintext attacks, and thus ciphertexts might be malleable, which makes it possible to forge in the authenticity game.

Definition 15 (Authenticity of ME)**.** *We say that an ME* Π *satisfies* authenticity *if for all PPT adversaries* A*:*

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

where game $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig.1.*

Finally, a secure ME is an ME satisfying all the properties.

Definition 16 (Secure ME)**.** *We say that an ME* Π *is secure, if* Π *satisfies* privacy *(Def. 14) and* authenticity *(Def. 15).*

Sometimes, we will also consider a weaker definition where there is an a priori upper bound on the number of queries an attacker can make to oracles RKGen and PolGen. We refer to this variant as security against bounded collusions. In particular, we say that an ME is (q_1, q_1', q_2, q_2')-secure if it has (q_1, q_1', q_2, q_2')-privacy and authenticity, where q_1, q_1' (resp. q_2, q_2') denote the number of queries to RKGen and PolGen allowed by A_1 (resp. A_2) in the privacy game.

Relation to ABE. An ME for arbitrary policies can be used as a CP-ABE with the same expressiveness. The idea is to ignore the attributes of the sender and the policy of the receiver. It is sufficient to set the ABE master public key to $(\mathsf{mpk}, \mathsf{ek}_\sigma)$ and an ABE receiver's decryption key to $(\mathsf{dk}_\rho, \mathsf{dk}_\phi)$, where ek_σ is the encryption key generated for attributes $\sigma = 0^\lambda$, dk_ϕ is the policy key for a tautology ϕ (i.e., a circuit whose output is always 1 regardless of the input), and dk_ρ is the decryption key for attributes ρ. The encryption of a message m under a policy $\mathbb{R}$ works by running the ME encryption algorithm $\mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)$. The receiver will decrypt the ciphertext by using the keys $(\mathsf{dk}_\rho, \mathsf{dk}_\phi)$. Since ϕ is a tautology, it does not matter under which attributes the message has been encrypted. Thus, the scheme will work as a normal CP-ABE.

Following a similar reasoning, ME implies KP-ABE. This is achieved by setting $\mathsf{ek}_\sigma = \sigma$, and by using the same approach described above (i.e., set the sender's policy circuit $\mathbb{R}$ to a tautology ϕ which ignores the receiver's attributes). Note that for this implication authenticity is not required, which is reminiscent of the fact that in ABE the attributes are not explicitly certified by an authority.

3.2 Arranged Matchmaking Encryption

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:

```
$\mathbf{G}^{\mathsf{arr\text{-}priv}}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk},\mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1, \alpha) \leftarrow\$ \mathsf{A}_1^{\mathsf{O}_1,\mathsf{O}_2}(1^\lambda, \mathsf{mpk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{ek}_{\sigma_b} \leftarrow\$ \mathsf{SKGen}(\mathsf{mpk},\mathsf{msk},\sigma_b)$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{mpk},\mathsf{ek}_{\sigma_b},\mathbb{R}_b,m_b)$
$b' \leftarrow\$ \mathsf{A}_2^{\mathsf{O}_1,\mathsf{O}_2}(1^\lambda, c, \alpha)$
If $(b' = b)$ return 1
Else return 0
```

```
$\mathbf{G}^{\mathsf{arr\text{-}auth}}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk},\mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(c, \rho, \mathbb{S}) \leftarrow\$ \mathsf{A}^{\mathsf{O}_1,\mathsf{O}_2}(1^\lambda, \mathsf{mpk})$
$\mathsf{dk}_{\rho,\mathbb{S}} \leftarrow\$ \mathsf{RKGen}(\mathsf{mpk},\mathsf{msk},\rho,\mathbb{S})$
$m = \mathsf{Dec}(\mathsf{mpk},\mathsf{dk}_{\rho,\mathbb{S}},c)$
If $\forall \sigma \in \mathcal{Q}_{\mathsf{O}_1} : (\mathbb{S}(\sigma) = 0) \land (m \neq \bot)$
  return 1
Else return 0
```

Figure 2: Games defining privacy and authenticity of A-ME. Oracles O_1, O_2 are implemented by $\mathsf{SKGen}(\mathsf{msk},\cdot)$ and $\mathsf{RKGen}(\mathsf{msk},\cdot)$.

$\mathsf{SKGen}, \mathsf{Enc}$: 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 $\bot$ (denoting an error).

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

Definition 17 (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

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

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

Definition 18 (Privacy of A-ME)**.** *An A-ME* Π *satisfies* privacy *if for all valid PPT adversaries* A*:*

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

where game $\mathbf{G}^{\mathsf{arr\text{-}priv}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig. 2. Adversary* A *is called valid if* $\forall(\rho,\mathbb{S}) \in \mathcal{Q}_{\mathsf{O}_2}$ *it satisfies the following invariant:*

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

$$\begin{aligned}(\mathbb{R}_0(\rho) = \mathbb{R}_1(\rho) = 0) &\lor (\mathbb{S}(\sigma_0) = \mathbb{S}(\sigma_1) = 0)\\ \lor (\mathbb{R}_0(\rho) = \mathbb{S}(\sigma_1) = 0) &\lor (\mathbb{R}_1(\rho) = \mathbb{S}(\sigma_0) = 0);\end{aligned} \tag{2}$$

- ***(Match condition).*** *Or (if* $\exists(\hat{\rho}, \hat{\mathbb{S}}) \in \mathcal{Q}_{\mathsf{O}_2}$ *s.t. Eq.* (2) *does not hold)*

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

Definition 19 (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 $\mathbf{G}^{\mathsf{arr\text{-}auth}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig. 2.*

Definition 20 (Secure A-ME)**.** *An A-ME* Π *is* secure *if it satisfies privacy (Def. 18), and authenticity (Def. 19).*

Relation to ABE. As for ME, A-ME for arbitrary policies implies CP-ABE and KP-ABE with the same expressiveness. The constructions are similar to the ones discussed in §3.1 for the case of ME. It is also easy to show that A-ME implies dual-policy ABE, which is achieved by setting $\mathsf{ek}_\sigma = \sigma$. Note that for all these implications, authenticity is not required.

Relation between ME and A-ME. We stress that ME and A-ME are incomparable. On the one hand, it is not clear how to use an A-ME to define an ME. This is because A-ME decryption key $\mathsf{dk}_{\rho,\mathbb{S}}$ describes both receiver's attributes and policy, and thus it is unclear how to implement the PolGen algorithm of an ME.

On the other hand, it is unclear how to define an A-ME starting with an ME. The natural construction which sets $\mathsf{dk}_{\rho,\mathbb{S}} = (\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S})$ does not work. In a nutshell, this is because a malicious receiver can detach the two keys, thus breaking the privacy of the A-ME. For concreteness, let $\mathbb{R}_0$, $\mathbb{R}_1$, $\sigma_0 = \sigma_1 = \sigma$ be the policies and the attributes contained in the challenge chosen by the adversary during the privacy game. The attacker can request a first decryption key $\mathsf{dk}_{\rho,\mathbb{S}} = (\mathsf{dk}_\rho, \mathsf{dk}_\mathbb{S})$ such that ρ satisfies both $\mathbb{R}_0$ and $\mathbb{R}_1$, but $\mathbb{S}(\sigma) = 0$. Next, it can request a second decryption key $\mathsf{dk}_{\rho',\mathbb{S}'} = (\mathsf{dk}_{\rho'}, \mathsf{dk}_{\mathbb{S}'})$ for which the symmetric condition holds: $\mathbb{R}_0(\rho') = 0$ and $\mathbb{R}_1(\rho') = 0$, but $\mathbb{S}'(\sigma) = 1$. Finally, it can interleave the keys creating a new decryption key $\mathsf{dk}_{\rho,\mathbb{S}'} = (\mathsf{dk}_\rho, \mathsf{dk}_{\mathbb{S}'})$, which makes it possible to decrypt the challenge ciphertext and win the game. Observe that both decryption keys are legal queries in the privacy game. Hence, the attacker is valid.

4 Black-Box Constructions

We explore black-box constructions of ME and A-ME from several types of FE schemes. In particular, in §4.1 we give a construction of ME based on rFE and FE. As discussed in the introduction, such a construction allows us to obtain ME from weaker assumptions, at the price of achieving only security against bounded collusions. In §4.2, we give a construction of ME that is secure against unbounded collusions, based on 2FE (and thus on stronger assumptions). Finally, in §4.3, we show a construction of A-ME based on FE. All schemes additionally rely on digital signatures and on NIZK proofs.

4.1 ME from rFE

Our construction is based on the following two functionalities f^{FE} and f^{rFE}:

$$f^{\mathsf{FE}}_{\mathbb{S}}(\sigma, m) = \begin{cases} m, & \text{if } \sigma \neq \bot \wedge \mathbb{S}(\sigma) = 1 \\ \bot, & \text{otherwise} \end{cases}$$

and

$$f^{\mathsf{rFE}}_{(\rho,\mathsf{mpk}_{\mathsf{FE}})}(\mathbb{R},\sigma,m;r) = \begin{cases}\mathsf{Enc}(\mathsf{mpk}_{\mathsf{FE}},(\sigma,m);r), \text{if } \mathbb{R}(\rho)=1\\ \mathsf{Enc}(\mathsf{mpk}_{\mathsf{FE}},(\bot,\bot);r), \text{ otherwise.}\end{cases}$$

Construction 1 (ME for arbitrary policies)**.** *Let* FE*,* rFE*,* SS*,* NIZK *be respectively an FE scheme for the deterministic functionality* f^{FE}*, a rFE scheme for the randomized functionality* f^{rFE}*, a signature scheme, and a NIZK proof system for the NP relation:*

$$R_1 \overset{def}{=} \left\{ ((c,\mathsf{pk},\mathsf{mpk}_{\mathsf{rFE}}),(\sigma,s)) : \begin{array}{c} \exists r, m, \mathbb{R} \ s.t. \\ c = \mathsf{Enc}_{\mathsf{rFE}}(\mathsf{mpk}_{\mathsf{rFE}},(\mathbb{R},\sigma,m);r) \wedge \\ \mathsf{Ver}(\mathsf{pk},s,\sigma)=1 \end{array} \right\}.$$

We construct an ME scheme in the following way:

$\mathsf{Setup}(1^\lambda)$**:** *On input the security parameter* 1^λ*, the setup algorithm computes* $(\mathsf{mpk}_{\mathsf{FE}},\mathsf{msk}_{\mathsf{FE}}) \leftarrow\$ \mathsf{Setup}_{\mathsf{FE}}(1^\lambda)$*,* $(\mathsf{sk},\mathsf{pk}) \leftarrow\$ \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$*,* $(\mathsf{mpk}_{\mathsf{rFE}},\mathsf{msk}_{\mathsf{rFE}}) \leftarrow\$ \mathsf{Setup}_{\mathsf{rFE}}(1^\lambda)$*, and* $\omega \leftarrow\$ \mathsf{I}(1^\lambda)$*. Finally, it outputs the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{rFE}},\mathsf{sk})$*, the master policy key* $\mathsf{kpol} = \mathsf{msk}_{\mathsf{FE}}$*, and the master public key* $\mathsf{mpk} = (\mathsf{pk},\omega,\mathsf{mpk}_{\mathsf{FE}},\mathsf{mpk}_{\mathsf{rFE}})$*. Recall that all other algorithms are implicitly given* mpk *as input.*

$\mathsf{SKGen}(\mathsf{msk},\sigma)$**:** *On input the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{rFE}},\mathsf{sk})$*, and attributes* $\sigma \in \{0,1\}^*$*, the algorithm returns the encryption key* $\mathsf{ek}_\sigma = (\sigma,s)$ *where* $s \leftarrow\$ \mathsf{Sign}(\mathsf{sk},\sigma)$ *(i.e., s is a signature on attributes* $\sigma \in \{0,1\}^*$*).*

$\mathsf{RKGen}(\mathsf{msk},\rho)$**:** *On input the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{rFE}},\mathsf{sk})$*, and attributes* $\rho \in \{0,1\}^*$*, the algorithm computes the decryption key* $\mathsf{sk}_{(\rho,\mathsf{mpk}_{\mathsf{FE}})} \leftarrow\$ \mathsf{KGen}_{\mathsf{rFE}}(\mathsf{msk}_{\mathsf{rFE}},(\rho,\mathsf{mpk}_{\mathsf{FE}}))$*. Then, it outputs the decryption key* $\mathsf{dk}_\rho = \mathsf{sk}_{(\rho,\mathsf{mpk}_{\mathsf{FE}})}$*.*

$\mathsf{PolGen}(\mathsf{kpol},\mathbb{S})$**:** *On input the master policy key* $\mathsf{kpol} = \mathsf{msk}_{\mathsf{FE}}$*, and policy* $\mathbb{S}$ *represented as a circuit, the algorithm computes the function key* $\mathsf{sk}_{\mathbb{S}}$ *by running* $\mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}},\mathbb{S})$*. Then, it outputs the decryption key* $\mathsf{dk}_{\mathbb{S}} = \mathsf{sk}_{\mathbb{S}}$*.*

$\mathsf{Enc}(\mathsf{ek}_\sigma,\mathbb{R},m)$**:** *On input an encryption key* $\mathsf{ek}_\sigma = (\sigma,s)$*, a policy* $\mathbb{R}$ *represented as a circuit, and a message m, the algorithm encrypt the message by computing* $c \leftarrow\$ \mathsf{Enc}_{\mathsf{rFE}}(\mathsf{mpk}_{\mathsf{rFE}},(\mathbb{R},\sigma,m))$*. Finally, it returns the ciphertext* $\hat{c} = (c,\pi)$ *where* $\pi \leftarrow\$ \mathsf{P}(\omega,(\mathsf{pk},c,\mathsf{mpk}_{\mathsf{rFE}}),(\sigma,s))$*.*

$\mathsf{Dec}(\mathsf{dk}_\rho,\mathsf{dk}_{\mathbb{S}},c)$**:** *On input two keys* $\mathsf{dk}_\rho = \mathsf{sk}_{(\rho,\mathsf{mpk}_{\mathsf{FE}})}$*,* $\mathsf{dk}_{\mathbb{S}} = \mathsf{sk}_{\mathbb{S}}$*, and a ciphertext* $\hat{c} = (c,\pi)$*, the algorithm first checks whether* $\mathsf{V}(\omega,(\mathsf{pk},c,\mathsf{mpk}_{\mathsf{rFE}}),\pi) = 1$*. If that is not the case, it returns* $\bot$*, and else it returns* $\mathsf{Dec}_{\mathsf{FE}}(\mathsf{sk}_{\mathbb{S}},\mathsf{Dec}_{\mathsf{rFE}}(\mathsf{sk}_{(\rho,\mathsf{mpk}_{\mathsf{FE}})},c))$*.*

Correctness of the scheme follows directly by the correctness of the underlying primitives. As for security, we establish the following result, whose proof appears in §A.1 of the appendix.

Theorem 1. *Let* rFE*,* FE*,* SS*,* NIZK *be as above. If* rFE *is* $(q_1,1,q_2)$*-NA-SIM-secure (Def.4),* FE *is* (q_1',q_1,q_2')*-SIM-secure,* SS *is EUF-CMA (Def.2), and* NIZK *satisfied adaptive multi-theorem zero knowledge (Def.11) and knowledge soundness (Def.12), then the ME scheme* Π *from Construction 1 is* (q_1,q_1',q_2,q_2')*-secure.*

4.2 ME from 2-Input FE

In this section we explain how to construct an ME combining a signature scheme SS, a non-interactive zero-knowledge proof NIZK, an 2FE scheme. In order to build ME from 2FE, we use a 2-ary functionality f that checks if a match occurs. More formally, consider the following functionality $f : \mathcal{K} \times \mathcal{X}_0 \times \mathcal{X}_1 \to \{0,1\}^* \cup \{\perp\}$:

$$f_\rho((\mathbb{R}, \sigma, m), \mathbb{S}) = \begin{cases} m, & \mathbb{S}(\sigma) = 1 \land \mathbb{R}(\rho) = 1 \\ \perp, & \text{otherwise.} \end{cases}$$

Construction 2 (ME for arbitrary policies)**.** *Let* 2FE*,* SS*, and* NIZK *be respectively a 2FE for the functionality f above, a signature scheme, and a NIZK proof system for the NP relation:*

$$R_2 \overset{def}{=} \left\{ ((\mathsf{pk}, c, \mathsf{ek}_0), (\sigma, s)) : \begin{array}{c} \exists r, m, \mathbb{R} \ s.t. \\ c = \mathsf{Enc}_{\mathsf{2FE}}(\mathsf{ek}_0, (\mathbb{R}, \sigma, m); r) \land \mathsf{Ver}(\mathsf{pk}, s, \sigma) = 1 \end{array} \right\}.$$

We build an ME scheme in the following way:

$\mathsf{Setup}(1^\lambda)$**:** *On input the security parameter 1^λ, the setup algorithm runs* $(\mathsf{ek}_0, \mathsf{ek}_1, \mathsf{msk}_{\mathsf{2FE}}) \leftarrow\$ \mathsf{Setup}_{\mathsf{2FE}}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$*, and* $\omega \leftarrow\$ \mathsf{I}(1^\lambda)$*. Finally, it outputs the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{2FE}}, \mathsf{sk})$*, the master policy key* $\mathsf{kpol} = \mathsf{ek}_1$*, and the master public key* $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{ek}_0)$*.*

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$**:** *On input the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{2FE}}, \mathsf{sk})$*, and attributes* $\sigma \in \{0,1\}^*$*, the algorithm returns the encryption key* $\mathsf{ek}_\sigma = (\sigma, s)$ *where* $s = \mathsf{Sign}(\mathsf{sk}, \sigma)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho)$**:** *On input the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{2FE}}, \mathsf{sk})$*, and attributes* $\rho \in \{0,1\}^*$*, the algorithm computes the key* $\mathsf{sk}_\rho \leftarrow\$ \mathsf{KGen}_{\mathsf{2FE}}(\mathsf{msk}_{\mathsf{2FE}}, \rho)$*. Then, it outputs* $\mathsf{dk}_\rho = \mathsf{sk}_\rho$*.*

$\mathsf{PolGen}(\mathsf{kpol}, \mathbb{S})$**:** *On input the master policy key* $\mathsf{kpol} = \mathsf{ek}_1$*, and a policy* $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ *represented as a circuit, the algorithm runs* $c_1 \leftarrow\$ \mathsf{Enc}_{\mathsf{2FE}}(\mathsf{ek}_1, \mathbb{S})$*. Then, it outputs the decryption key* $\mathsf{dk}_{\mathbb{S}} = c_1$*.*

$\mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)$**:** *On input an encryption key* $\mathsf{ek}_\sigma = (\sigma, s)$*, a policy* $\mathbb{R} : \{0,1\}^* \to \{0,1\}$ *represented as a circuit, and a message m, the algorithm encrypts the message by computing* $c_0 \leftarrow\$ \mathsf{Enc}_{\mathsf{2FE}}(\mathsf{ek}_0, (\mathbb{R}, \sigma, m))$*. Finally, it returns the ciphertext* $c = (c_0, \pi)$ *where* $\pi \leftarrow\$ \mathsf{P}(\omega, (\mathsf{pk}, c_0, \mathsf{ek}_0), (\sigma, s))$*.*

$\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{dk}_{\mathbb{S}}, c)$**:** *On input a decryption key* $\mathsf{dk}_\rho = \mathsf{sk}_{\mathsf{pk}}$*, a decryption key* $\mathsf{dk}_{\mathbb{S}} = c_1$*, and a ciphertext* $c = (c_0, \pi)$*, the algorithm first checks whether* $\mathsf{V}(\omega, (\mathsf{pk}, c_0, \mathsf{ek}_0), \pi) = 1$*. If that is not the case, it returns $\perp$, and else it returns* $\mathsf{Dec}_{\mathsf{2FE}}(\mathsf{sk}_\rho, c_0, c_1)$*.*

Correctness of the scheme follows directly by the correctness of the underlying primitives. As for security, we establish the following result, whose proof appears in §A.2 of the appendix.

Theorem 2. *Let* 2FE*,* SS*, and* NIZK *be respectively a 2FE scheme, a signature scheme, and a NIZK proof for the relation R_2. If* 2FE *is indistinguishably secure in the 1-semiprivate setting (Def. 9),* SS *is EUF-CMA (Def. 2), and* NIZK *satisfies adaptive multi-theorem zero knowledge (Def. 11) and knowledge soundness (Def. 12), then the ME scheme Π from Construction 2 is secure (Def. 16).*

4.3 A-ME from FE

In this section, we show a general construction of A-ME from FE. Consider the following functionality $f : \mathcal{K} \times \mathcal{X} \to \{0,1\}^* \cup \{\perp\}$:

$$f_{(\rho, \mathbb{S})}(\mathbb{R}, \sigma, m) = \begin{cases} m, & \mathbb{S}(\sigma) = 1 \land \mathbb{R}(\rho) = 1 \\ \perp, & \text{otherwise.} \end{cases}$$

Construction 3 (A-ME). *Let* FE *and* SS *be respectively an FE scheme for the functionality* f *above and a signature scheme, and a NIZK proof system for the NP relation:*

$$R_3 \overset{def}{=} \left\{ ((\mathsf{pk}, c, \mathsf{mpk}_{\mathsf{FE}}), (\sigma, s)) : \begin{array}{c} \exists r, m, \mathbb{R} \text{ s.t.} \\ c = \mathsf{Enc}_{\mathsf{FE}}(\mathsf{mpk}_{\mathsf{FE}}, (\mathbb{R}, \sigma, m); r) \wedge \mathsf{Ver}(\mathsf{pk}, s, \sigma) = 1 \end{array} \right\}.$$

We build an A-ME scheme in the following way:

$\mathsf{Setup}(1^\lambda)$: *On input the security parameter* 1^λ*, the setup algorithm runs* $(\mathsf{mpk}_{\mathsf{FE}}, \mathsf{msk}_{\mathsf{FE}}) \leftarrow\$ \mathsf{Setup}_{\mathsf{FE}}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$*, and* $\omega \leftarrow\$ \mathsf{I}(1^\lambda)$*. Finally, it outputs the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{FE}}, \mathsf{sk})$*, and the master public key* $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{mpk}_{\mathsf{FE}})$*.*

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$: *On input the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{FE}}, \mathsf{sk})$*, and* $\sigma\{0,1\}^*$*, the algorithm returns the encryption key* $\mathsf{ek}_\sigma = (\sigma, s)$ *where* $s = \mathsf{Sign}(\mathsf{sk}, \sigma)$*.*

$\mathsf{RKGen}(\mathsf{msk}, \rho, \mathbb{S})$: *On input the master secret key* $\mathsf{msk} = (\mathsf{msk}_{\mathsf{FE}}, \mathsf{sk})$*, attributes* $\rho \in \{0,1\}^*$*, and a policy* $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ *represented as a circuit, the algorithm computes the encryption key* $\mathsf{sk}_{(\rho,\mathbb{S})} \leftarrow\$ \mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, (\rho, \mathbb{S}))$*. Then, it outputs* $\mathsf{dk}_{\rho,\mathbb{S}} = \mathsf{sk}_{(\rho,\mathbb{S})}$*.*

$\mathsf{Enc}(\mathsf{ek}_\sigma, \mathbb{R}, m)$: *On input , an encryption key* $\mathsf{ek}_\sigma = (\sigma, s)$*, a policy* $\mathbb{R} : \{0,1\}^* \to \{0,1\}$ *represented as a circuit, and a message* m*, the algorithm encrypts the message in the following way:* $c' \leftarrow\$ \mathsf{Enc}_{\mathsf{FE}}(\mathsf{mpk}_{\mathsf{FE}}, (\mathbb{R}, \sigma, m))$*. Finally, it returns the ciphertext* (c', π) *where* $\pi \leftarrow\$ \mathsf{P}(\omega, (\mathsf{pk}, c', \mathsf{mpk}_{\mathsf{FE}}), (\sigma, s))$*.*

$\mathsf{Dec}(\mathsf{dk}_{\rho,\mathbb{S}}, c)$: *On input, a decryption key* $\mathsf{dk}_{\rho,\mathbb{S}} = \mathsf{sk}_{(\rho,\mathbb{S})}$*, and a ciphertext* $c = (c', \pi)$*, the algorithm first checks whether* $\mathsf{V}(\omega, (\mathsf{pk}, c', \mathsf{mpk}_{\mathsf{FE}}), \pi) = 1$*. If that is not the case, it returns* $\perp$*, and else it returns* $\mathsf{Dec}_{\mathsf{FE}}(\mathsf{sk}_{(\rho,\mathbb{S})}, c')$*.*

The correctness of the above scheme follows directly by the correctness of the underlying primitives. As for security, we establish the following result, whose proof appears in §A.3 of the appendix.

Theorem 3. *Let* FE*,* SS*, and* NIZK *be respectively an FE scheme, a signature scheme, and a NIZK proof for the relation* R_3*. If* FE *is secure (Def. 7),* SS *is EUF-CMA (Def. 2), and* NIZK *satisfies adaptive multi-theorem zero knowledge (Def. 11) and knowledge soundness (Def. 12), then the A-ME* Π *from Construction 3 is secure (Def. 20).*

5 Identity-Based Matchmaking Encryption

In this section, we present a practical ME for the identity-based setting. As in ME, attributes are encoded by bit strings, but now each attribute $x \in \{0,1\}^*$ satisfies only the access policy $\mathbb{A} = x$, which means that both the sender and the receiver specify a single identity instead of general policies (represented as a circuit). We will denote by snd and rcv, respectively, the target identities (i.e., the access policies) specified by the receiver and by the sender.

While any ME as defined in §3 perfectly works for this restricted setting, the problem is that in order to select the identity snd of the source, a receiver must ask to the administrator the corresponding key $\mathsf{dk}_{\mathsf{snd}}$ such that $\mathbb{S} = \mathsf{snd}$. (Recall that the sender, instead, can already specify the target identity $\mathbb{R} = \mathsf{rcv}$ on the fly, during encryption.) In particular, if the receiver is interested in decrypting ciphertexts from several distinct sources, it must ask for several decryption keys $\mathsf{dk}_{\mathsf{snd}}$, which is impractical.[4]

[4] This is *not* an issue for an ME that supports arbitrary policies, as in that case, a single policy encodes a large number of attributes.

We resolve this issue by removing algorithm PolGen from the syntax of an IB-ME, so that the decryption algorithm takes directly as input the description of the target identity snd (i.e., $\mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{snd}, c)$). This way, the receiver can specify the target identity the source must satisfy on the fly, without talking to the authority.

5.1 Security of IB-ME

The choice of removing the PolGen algorithm has an impact on the security properties for IB-ME. Below, we revisit each security guarantee in the identity-based setting and explain how (and why) the security definition has to be adapted. We refer the reader to Fig. 3 for the formal definitions.

Privacy of IB-ME. We cannot require that the sender's identity remains hidden in case of a decryption failure due to a mismatch condition. In particular, a malicious receiver can always change the sender's target identity in order to infer under which identity a ciphertext has been encrypted.

More formally, consider the adversary that chooses a tuple $(m, m, \mathsf{rcv}, \mathsf{rcv}, \sigma_0, \sigma_1)$, and receives a ciphertext c such that $c \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_{\sigma_b}, \mathsf{rcv}, m)$, where the encryption key ek_{σ_b} corresponds to identity σ_b; the attacker can simply pick a target identity snd' such that, say, $\sigma_0 = \mathsf{snd}'$ (whereas $\sigma_1 \neq \mathsf{snd}'$), and thus distinguish σ_0 from σ_1 by decrypting c with dk_ρ and target identity snd'.[5] On the other hand, privacy might still hold in case of mismatch, as long as the keys dk_ρ held by the receiver correspond to identities ρ that do not match the receiver's target identity. Thus, in the security game, an attacker is now valid if for every decryption key dk_ρ obtained from the oracle, it holds that $\rho \neq \mathsf{rcv}_0$ and $\rho \neq \mathsf{rcv}_1$, where the target identities $\mathsf{rcv}_0, \mathsf{rcv}_1$ are chosen by the adversary. Lastly, note that in case of a match, if a receiver has identity ρ and specifies a policy snd, it can automatically infer that $\sigma = \mathsf{snd}$ and $\mathsf{rcv} = \rho$. For this reason, the privacy game does not consider any match condition.

Finally, we note that the above security definition does not guarantee that the message m remains secret with respect to an *honest receiver* that chooses the "wrong" target identity snd. The latter is, however, a desirable feature that our practical scheme will satisfy (cf. Remark 1).

Authenticity of IB-ME. Turning to unforgeability in the identity-based setting, the forgery (c, ρ, snd) is considered valid if for all encryption keys ek_σ obtained by the adversary it holds that $\sigma \neq \mathsf{snd}$, and moreover the identity ρ is not held by the adversary (i.e., the adversary cannot "forge to itself").

Security definitions. The definitions below capture the very same correctness and security requirements of an ME, but translated to the identity-based case.

Definition 21 (Correctness of IB-ME)**.** *An IB-ME* $\Pi = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{RKGen}, \mathsf{Enc}, \mathsf{Dec})$ *is correct if* $\forall \lambda \in \mathbb{N}$, $\forall(\mathsf{mpk}, \mathsf{msk})$ *output by* $\mathsf{Setup}(1^\lambda)$*,* $\forall m \in \mathcal{M}$*,* $\forall \sigma, \rho, \mathsf{rcv}, \mathsf{snd} \in \{0,1\}^*$*:*

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

whenever $\sigma = \mathsf{snd}$ *and* $\rho = \mathsf{rcv}$*, and otherwise*

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

where ek_σ*,* dk_ρ *are generated by* $\mathsf{SKGen}(\mathsf{msk}, \sigma)$*, and* $\mathsf{RKGen}(\mathsf{msk}, \rho)$*.*

[5] This attack can be generalized to show that privacy does not hold if the PolGen algorithm (and thus the policy key kpol) is made public.

```
$\mathbf{G}^{\textsf{ib-priv}}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathsf{rcv}_0, \mathsf{rcv}_1, \sigma_0, \sigma_1, \alpha) \leftarrow\$ \mathsf{A}_1^{\mathsf{O}_1,\mathsf{O}_2}(1^\lambda, \mathsf{mpk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{ek}_{\sigma_b} \leftarrow\$ \mathsf{SKGen}(\mathsf{msk}, \sigma_b)$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{ek}_{\sigma_b}, \mathsf{rcv}_b, m_b)$
$b' \leftarrow\$ \mathsf{A}_2^{\mathsf{O}_1,\mathsf{O}_2}(1^\lambda, c, \alpha)$
If $(b' = b)$ return 1
Else return 0
```

```
$\mathbf{G}^{\textsf{ib-auth}}_{\Pi,\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(c, \rho, \mathsf{snd}) \leftarrow\$ \mathsf{A}^{\mathsf{O}_1,\mathsf{O}_2}(1^\lambda, \mathsf{mpk})$
$\mathsf{dk}_\rho \leftarrow\$ \mathsf{RKGen}(\mathsf{msk}, \rho)$
$m = \mathsf{Dec}(\mathsf{dk}_\rho, \mathsf{snd}, c)$
If $\forall \sigma \in \mathcal{Q}_{\mathsf{O}_1} : (\sigma \neq \mathsf{snd}) \wedge (\rho \notin \mathcal{Q}_{\mathsf{O}_2}) \wedge$
  $(m \neq \perp)$
  return 1
Else return 0
```

Figure 3: Games defining privacy and authenticity security of IB-ME. Oracles O_1, O_2 are implemented by $\mathsf{SKGen}(\mathsf{msk}, \cdot)$, $\mathsf{RKGen}(\mathsf{msk}, \cdot)$.

Definition 22 (Privacy of IB-ME)**.** *We say that an IB-ME* Π *satisfies* privacy *if for all valid PPT adversaries* A*:*

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

where game $\mathbf{G}^{\textsf{ib-priv}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig 3. Adversary* A *is called valid if* $\forall \rho \in \mathcal{Q}_{\mathsf{O}_2}$ *it satisfies the following invariant:*

- ***(Mismatch condition).*** $\rho \neq \mathsf{rcv}_0 \wedge \rho \neq \mathsf{rcv}_1$

Definition 23 (Authenticity of IB-ME)**.** *We say that an IB-ME* Π *satisfies* authenticity *if for all PPT adversaries* A*:*

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

where game $\mathbf{G}^{\textsf{ib-auth}}_{\Pi,\mathsf{A}}(\lambda)$ *is depicted in Fig.3.*

Definition 24 (Secure IB-ME)**.** *We say that an IB-ME* Π *is secure if it satisfies privacy (Def. 22) and authenticity (Def. 23).*

5.2 The Scheme

We are now ready to present our practical IB-ME scheme.

Construction 4 (IB-ME)**.** *The construction works as follows.*

$\mathsf{Setup}(1^\lambda)$**:** *Let* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *be a symmetric pairing, and* P *a generator of* $\mathbb{G}$*, with* $\mathbb{G}$*, and* $\mathbb{G}_T$ *of an order* q *that depends on* λ*. We also have three hash functions* $H : \{0,1\}^* \to \mathbb{G}$*,* $H' : \{0,1\}^* \to \mathbb{G}$*,* $\hat{H} : \mathbb{G}_T \to \{0,1\}^\ell$*, modeled as random oracles, and a polynomial-time computable padding function* $\Phi : \{0,1\}^n \to \{0,1\}^\ell$*. We require that for all* $m \in \{0,1\}^n$ *one can verify in polynomial time if* m *has been padded correctly, and moreover that* $\Phi(m)$ *is efficiently invertible. On input the security parameter* 1^λ*, the setup algorithm samples two random* $r, s \in \mathbb{Z}_q$*, and sets* $P_0 = P^r$*. Finally, it outputs the master public key* $\mathsf{mpk} = (e, \mathbb{G}, \mathbb{G}_T, q, P, P_0, H, H', \hat{H}, \Phi)$ *and the master secret key is* $\mathsf{msk} = (r, s)$*. Recall that all other algorithms are implicitly given* mpk *as input.*

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$**:** *On input the master secret key* msk*, and identity* σ*, the algorithm outputs* $\mathsf{ek}_\sigma = H'(\sigma)^s$*.*

$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho)$**:** *On input the master secret key* msk*, and identity* ρ*, the algorithm outputs* $\mathsf{dk}_\rho = (\mathsf{dk}^1_\rho, \mathsf{dk}^2_\rho, \mathsf{dk}^3_\rho) = (H(\rho)^r, H(\rho)^s, H(\rho))$*.*

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, m)$**:** *On input an encryption key* ek_σ*, a target identity* $\mathsf{rcv} = \rho$*, and a message* $m \in \{0,1\}^n$*, the algorithm proceeds as follows:*

1. *Sample random* $u, t \in \mathbb{Z}_q$*.*
2. *Compute* $T = P^t$ *and* $U = P^u$*.*
3. *Compute* $k_R = e(H(\rho), P_0^u)$ *and* $k_S = e(H(\rho), T \cdot \mathsf{ek}_\sigma)$*.*
4. *Compute* $V = \Phi(m) \oplus \hat{H}(k_R) \oplus \hat{H}(k_S)$*.*
5. *Output ciphertext* $C = (T, U, V)$*.*

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, c)$**:** *On input the master public key* mpk*, a decryption key* dk_ρ*, a target identity* $\mathsf{snd} = \sigma$*, and a message* m*, the algorithm proceeds as follows:*

1. *Parse* c *as* (T, U, V)*.*
2. *Compute* $k_R = e(\mathsf{dk}_\rho^1, U)$ *and* $k_S = e(\mathsf{dk}_\rho^2, H'(\sigma)) \cdot e(\mathsf{dk}_\rho^3, T)$*.*
3. *Compute* $\Phi(m) = V \oplus \hat{H}(k_R) \oplus \hat{H}(k_S)$
4. *If the padding is valid, return* m*. Otherwise, return* $\perp$*.*

Correctness. The correctness of the scheme only depends on the computation of k_R and k_S as evaluated by the decryption algorithm. Here, we require that the padding function Φ satisfies the property that a random string in $\{0,1\}^\ell$ has only a negligible probability to form a valid padding w.r.t. the function Φ.[6] Let k_R, k_S be the keys computed during encryption, and k'_R, k'_S the ones computed during decryption. The scheme is correct since $\forall \sigma, \rho, \mathsf{rcv}, \mathsf{snd} \in \{0,1\}^*$, $\mathsf{ek}_\sigma \leftarrow\!\!\$\ \mathsf{SKGen}(\mathsf{msk}, \sigma)$, $\mathsf{dk}_\rho \leftarrow\!\!\$\ \mathsf{RKGen}(\mathsf{msk}, \rho)$:

1. If $\sigma = \mathsf{snd}$ and $\rho = \mathsf{rcv}$:

$$k_R = e(H(\rho), P_0^u) = e(H(\rho)^r, P^u) = e(\mathsf{dk}_\rho^1, U) = k'_R,$$

and

$$\begin{aligned} k_S &= e(H(\rho), T \cdot \mathsf{ek}_\sigma) = e(H(\rho), T \cdot H'(\sigma)^s) = \\ &= e(H(\rho), T) \cdot e(H(\rho)^s, H'(\sigma)) = e(\mathsf{dk}_\rho^3, T) \cdot e(\mathsf{dk}_\rho^2, H'(\sigma)) = k'_S \end{aligned}$$

2. Otherwise, if $\rho \neq \mathsf{rcv} = \rho'$ or $\sigma \neq \mathsf{snd} = \sigma$

$$k_R = e(H(\rho'), P_0^u) \neq e(H(\rho)^r, P^u) = e(\mathsf{dk}_\rho^1, U) = k'_R,$$

or

$$\begin{aligned} k_S &= e(H(\rho), T \cdot \mathsf{ek}_\sigma) = e(H(\rho), T \cdot H'(\sigma)^s) = e(\mathsf{dk}_\rho^3, T) \cdot e(\mathsf{dk}_\rho^2, H'(\sigma)) \neq \\ &= e(\mathsf{dk}_\rho^3, T) \cdot e(\mathsf{dk}_\rho^2, H'(\sigma')) = k'_S. \end{aligned}$$

Since k'_R (resp. k'_S) is hashed by the random oracle $\hat{H}$, then $\hat{H}(k'_R)$ (resp. $\hat{H}(k'_S)$) is statistically close to a random string of length ℓ. Hence, with overwhelming probability, $V \oplus \hat{H}(k_R) \oplus \hat{H}(k'_S)$, where either $k_R \neq k'_R$ or $k_S \neq k'_S$, will produce an invalid padding, and the decryption algorithm returns $\perp$.

[6] This can be achieved, e.g., by setting $\ell = n + \lambda + 1$, and by appending to each message the string $1||0^\lambda$.

Table 2: Performance of high- and low-level cryptographic operations of IB-ME

Operation	**Minimum (ms)**	**Average (ms)**
Setup	2.197	2.213
RKGen	2.200	2.225
SKGen	3.400	3.429
Encryption	6.942	7.012
Decryption	4.344	4.385

Remark 1 (Plaintext secrecy w.r.t. unauthorized-but-honest receivers)**.** *We note that the plaintext is information-theoretically hidden from the point of view of an honest receiver which specifies a target identity that does not match the sender's identity. Moreover, the latter holds even given the internal state of the receiver at the end of the decryption procedure. In fact, since $\hat{H}(k_S)$ is statistically close to uniform, and $|\hat{H}(k_S)| = |\Phi(m)| = \ell$, the decryption algorithm will compute a symmetric key k_S different to the one generated during encryption.*[7]

Security. As for security, we establish the following result, whose proof appears in §A.4 of the appendix.

Theorem 4. *Let $\mathbb{G}$, $\mathbb{G}_T$ be two groups of prime order q, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an admissible bilinear map. If the BDH problem is hard in $(\mathbb{G}, \mathbb{G}_T, e)$ (Def. 10), then the IB-ME scheme Π from Construction 4 is secure (Def. 24) in the random oracle model.*

6 IB-ME Performance Evaluation and Application to Tor

In this section, we demonstrate that our IB-ME is practical and we use it to implement a novel system for anonymous but authentic communication. We first show in §6.1 the performance evaluation of our IB-ME implementation. We then describe in §6.2 an application for IB-ME built on top of our implementation. The proposed application is a *bulletin board hidden service* that allows parties to collect or exchange anonymous messages that have an expected format and come from authentic sources. It allows users to exchange IB-ME messages over the Tor network, specifically, using the Tor Hidden Services feature (cf. §6.2.1). Our bulletin board prototype can be used for covert communication by journalists or activists under authoritarian governments. It improves upon systems such as SecurePost ([39]) for verified group anonymity by providing much stronger privacy guarantees since ciphertexts can be vetted *before* decryption.

6.1 Implementation and Evaluation of the IB-ME cryptosystem

We provide an experimental evaluation of the IB-ME cryptosystem. To this end, we implemented a proof of concept in Python 3.6.5 using Charm 0.50 [2], a framework for prototyping pairing-based cryptosystems (among others). Since our IB-ME is defined using symmetric pairings (also called Type-I pairings), we instantiate it with a supersingular curve with a 512-bit base field (curve `SS512` in Charm), which gives approximately 80 bits of security [43]. The execution environment is an Intel NUC7i7BNH with an Intel Core i7-7567U CPU @ 3.50GHz and 16 GB of RAM, running Ubuntu 18.04 LTS.

Table 2 shows the cost in milliseconds associated to the main high- and low-level cryptographic operations of IB-ME. We executed these experiments in 50 different runs of 10 times

[7] It is important to recall that a similar guarantee does not hold in the identity-based setting, when the receiver is semi-honest (cf. §5.1).

Table 3: Space costs of IB-ME elements.

Element	Theoretical cost	Size (in bits)
Encryption key	$\lvert\mathbb{G}\rvert$	512
Decryption key	$3\lvert\mathbb{G}\rvert$	1536
Message	n	1024
Ciphertext	$2\lvert\mathbb{G}\rvert + \ell$	2129
Ciphertext expansion	$\frac{\ell}{n} + \frac{2\lvert\mathbb{G}\rvert}{n}$	≈2

each and both the minimum and average timing was taken for each operation; we use the Python module `timeit` for these measurements. It can be seen that the average timings for the main high-level operations of IB-ME, namely Encryption and Decryption, are 7.012 ms and 4.385 ms, respectively. These results show that the scheme is highly practical.

It is worth mentioning that there is room for improvement in the implementation if we use optimizations such as pre-computation of some pairing operations when one of the arguments is fixed (which occurs in the two pairings during decryption since one argument is a decryption key) or is reused (the two pairings in the encryption function have $H(\rho)$ as an argument), which can lead to speeds-up around 30%, as reported in [18]. Another potential optimization is the use of multipairings in the decryption operation. A promising direction would be to redefine the scheme in a Type-III pairing setting, which allows for more performant curves [22].

Finally, Table 3 shows a summary of the space costs associated to different elements of our IB-ME. We analyze both the theoretical cost and the actual values with the parameters of the experiment. In addition to the use of Charm's curve `SS512` (which implies that the size of $|\mathbb{G}| = 512$ bits and $|\mathbb{G}_T| = 1024$), we use for the size of identity bitstrings $|\mathbb{G}|$, for the size of messages $n = |\mathbb{G}_T|$, and for the padding output size $\ell = n + \lambda + 1 = 1105$.

6.2 An Anonymous Bulletin Board

Here, we describe the implementation of a bulletin board hidden service that is powered by our IB-ME scheme (cf. §5). In a nutshell, our application allows senders to post encrypted messages to an anonymous bulletin board, hosted by a Tor hidden service [49]. To this end, senders specify a target identity string that acts as the receiver's access policy, as well as the encryption key corresponding to their own identity. Conversely, receivers can fetch encrypted messages from the bulletin board, and try to decrypt them with their own decryption keys (associated with their identity) and the expected identity of the sender. Only those encrypted messages where there is a match between sender and receiver can be decrypted correctly.

Our system protects every party's privacy in several aspects. First of all, thanks to the nature of Tor hidden services, the IP addresses of each party and the connection between the client and the server remain hidden. Secondly, if decryption fails nothing is revealed to the parties.

Next, we will give a brief overview of Tor Hidden Services.

6.2.1 Tor and Hidden Services

Tor [47] is the most prominent P2P anonymous system, totaling more than 2 million users and $6,000$ relays. It allows clients to access the Internet anonymously by hiding the final destination of their connections. It achieves this by creating random circuits between the client and the destination (e.g., web server), where every relay is aware only of its incoming and outgoing links.

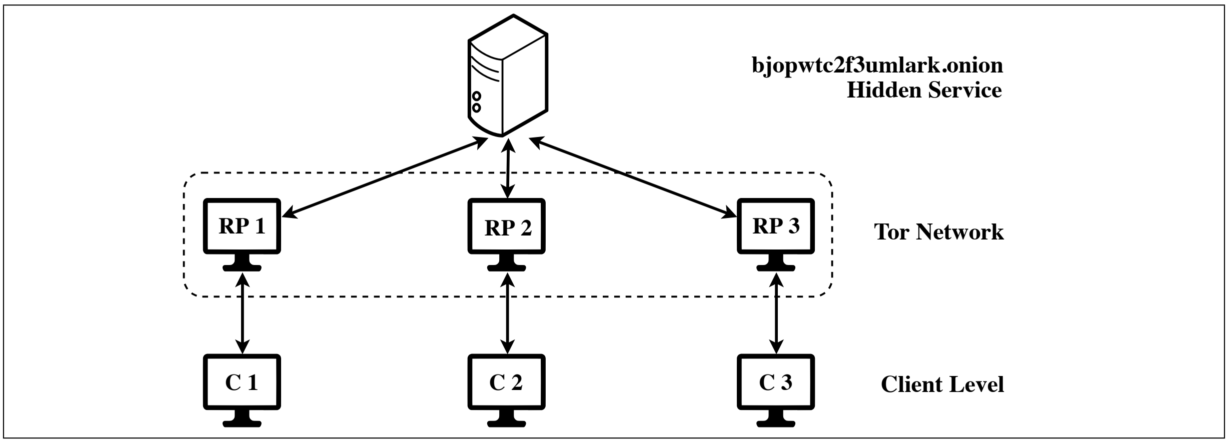

Figure 4: Example of interaction between three clients C1, C2, C3 and the anonymous bullentin board (http://bjopwtc2f3umlark.onion) using Tor. The relays RP1, RP2, and RP3 are the rendezvous points shared between the service and the respective clients. Each party communicates with the respective RP using a Tor circuit.

Various services can be set up so that they are accessible only within the Tor network. These Tor *Hidden Services* [49], or HS, are run without revealing their IP addresses and can be reached with no prior information. In order to deploy an HS, the owner needs to initialize the service by choosing some relays that will act as introduction points (IPs). The service will keep an open Tor circuit to each IP that will be used as the entry points to access the HS. The IPs' identities are communicated to Tor by creating a service descriptor entry. This entry contains all the information needed to access the service (e.g., description ID, list of IPs, etc.). Then, the entry is uploaded to the hidden service directory (HSDir) which stores the description entries of all available HSs. A node that wants to connect to an HS will (1) retrieve from HSDirs the correct description entry, (2) establish a Tor circuit to a random relay known as the *rendezvous point*, RP in short, and (3) reveal to one of the hidden service's IP (contained in the description entry) the address of the RP. The HS can now open a Tor circuit to the RP, so that the node and the HS can communicate without revealing their respective IP addresses.

6.2.2 Our Anonymous Bulletin Board

Our application is composed of two parts: a web server implemented as a Tor hidden service, and a command line client that is used to upload and download data from the server.

A user that wants to post a message to the bulletin board can use the client to encrypt it (using their IB-ME encryption key ek_σ and an identity string policy rcv for the intended receiver), and upload the ciphertext to the web server through the Tor network. These ciphertexts are publicly available.

A receiver can now use the client to download all the ciphertexts and try to decrypt each of them, using the receiver's decryption key dk_ρ and the sender's identity policy snd (given as input to the client). The client will report to the user the outcome of the decryption phase, showing all the successfully decrypted messages. The role of the web server is to store encrypted messages and to offer a simple REST API that allows clients to post and read these messages. In our prototype, we do not include any additional security measure, but in a real-world deployment, specific countermeasures should be taken in order to protect against potential denial of service attacks from clients (e.g., by requiring a proof-of-work along with the request) and/or include some authentication mechanisms. We refer the reader to Fig. 4 for an overview of the system.

As in any identity-based cryptosystem, key management requires a key generation service that generates and distributes encryption and decryption keys. This service could be implemented as another Tor hidden service, or even integrated with an existing HSDir, that automatically converts email addresses or phone numbers into keys. Another possibility is to assume the existence of an off-line authority so that users of the application obtain their keys through an out-of-band channel. In our prototype, we assume the latter option for simplicity.

Finally, note that the performance cost of our Tor application is dominated by the network latency of the Tor relays. Since the main focus of the paper is the new cryptographic primitive, we report only the performance evaluation of our IB-ME scheme (cf. §6.1).

7 Conclusions

We have proposed a new form of encryption, dubbed matchmaking encryption (ME), where both the sender and the receiver, described by their own attributes, can specify fine-grained access policies to encrypted data. ME enables several applications, e.g., communication between spies, social matchmaking, and more.

On the theoretical side, we put forward formal security definitions for ME and established the feasibility of ME supporting arbitrary policies by leveraging FE for randomized functionalities in conjunction with other more standard cryptographic tools. On the practical side, we constructed and implemented practical ME for the identity-based setting, with provable security in the random oracle model under the BDH assumption. We also showcased the utility of IB-ME to realize an anonymous bulletin board using the Tor network.

Our work leaves open several important questions. First, it would be interesting to construct ME from simpler assumptions. Second, it is conceivable that our black-box construction could be instantiated based on better assumptions since we only need secure rFE w.r.t. honest encryptors; unfortunately, the only definition that is specifically tailored for this setting [3] has some circularity problems [25, 1]. Third, a natural direction is to come up with efficient ME schemes for the identity-based setting without relying on random oracles, or to extend our scheme to the case of fuzzy matching [5]. Further extensions include the setting of chosen-ciphertext security, ME with multiple authorities, and creating an efficient infrastructure for key management and revocation.

References

[1] Shashank Agrawal and David J Wu. Functional encryption: Deterministic to randomized functions from simple assumptions. In *EUROCRYPT*, pages 30–61, 2017.

[2] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[3] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In *International Conference on Cryptography and Coding*, pages 65–84, 2013.

[4] Prabhanjan Ananth, Aayush Jain, Dakshita Khurana, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018.

[5] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *NDSS*, volume 7, pages 1–19, 2007.

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

[7] Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *PKC*, volume 6571, pages 90–108, 2011.

[8] Nuttapong Attrapadung and Shota Yamada. Duality in ABE: Converting attribute based encryption for dual predicate and dual policy via computational encodings. In *CT-RSA*, pages 87–105, 2015.

[9] 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.

[10] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, pages 566–582, 2001.

[11] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE S&P*, pages 321–334, 2007.

[12] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001.

[13] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[14] Claude Castelluccia, Stanisław Jarecki, and Gene Tsudik. Secret handshakes from CA-oblivious encryption. In *ASIACRYPT*, pages 293–307, 2004.

[15] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

[16] Melissa Chase and Sherman SM Chow. Improving privacy and security in multi-authority attribute-based encryption. In *CCS*, pages 121–130, 2009.

[17] Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. In *CCS*, pages 456–465, 2007.

[18] Craig Costello and Douglas Stebila. Fixed argument pairings. In *LATINCRYPT*, pages 92–108, 2010.

[19] Ivan Damgård, Helene Haagh, and Claudio Orlandi. Access control encryption: Enforcing information flow with cryptography. In *TCC*, pages 547–576, 2016.

[20] Ben Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. Iron: Functional encryption using intel SGX. In *CCS*, pages 765–782, 2017.

[21] Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi. Access control encryption for equality, comparison, and more. In *PKC*, pages 88–118, 2017.

[22] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[23] Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *EUROCRYPT*, pages 578–602, 2014.

[24] M Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Attribute-based authenticated key exchange. In *ACISP*, pages 300–317, 2010.

[25] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. In *TCC*, pages 325–351, 2015.

[26] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP*, pages 579–591, 2008.

[27] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, pages 89–98, 2006.

[28] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. *PKC*, pages 19–34, 2010.

[29] M. Jason Hinek, Shaoquan Jiang, Reihaneh Safavi-Naini, and Siamak F. Shahandashti. Attribute-based encryption without key cloning. *International Journal of Applied Cryptography*, 2(3):250–270, 2012.

[30] Lin Hou, Junzuo Lai, and Lixian Liu. Secret handshakes with dynamic expressive matching policy. In *ACISP*, pages 461–476, 2016.

[31] Stanisław Jarecki, Jihye Kim, and Gene Tsudik. Authentication for paranoids: Multi-party secret handshakes. In *ACNS*, pages 325–339, 2006.

[32] Stanisław Jarecki, Jihye Kim, and Gene Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In *CT-RSA*, pages 352–369, 2008.

[33] Stanisław Jarecki and Xiaomin Liu. Unlinkable secret handshakes and key-private group key management schemes. In *ACNS*, pages 270–287, 2007.

[34] Apu Kapadia, Patrick P Tsang, and Sean W Smith. Attribute-based publishing with hidden credentials and hidden policies. In *NDSS*, volume 7, pages 179–192, 2007.

[35] Sam Kim and David J Wu. Access control encryption for general policies from standard assumptions. In *ASIACRYPT*, pages 471–501, 2017.

[36] Vladimir Kolesnikov, Hugo Krawczyk, Yehuda Lindell, Alex Malozemoff, and Tal Rabin. Attribute-based key exchange with general policies. In *CCS*, pages 1451–1463, 2016.

[37] Jin Li, Kui Ren, and Kwangjo Kim. A2be: Accountable attribute-based encryption for abuse free access control. *IACR Cryptology ePrint Archive*, 2009:118, 2009.

[38] Jin Li, Kui Ren, Bo Zhu, and Zhiguo Wan. Privacy-aware attribute-based encryption with user accountability. In *ISC*, volume 9, pages 347–362, 2009.

[39] Michael Nekrasov, Danny Iland, Miriam Metzger, Lisa Parks, and Elizabeth Belding. A user-driven free speech application for anonymous and verified online, public group discourse. *Journal of Internet Services and Applications*, 9(1):21, 2018.

[40] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In *ACNS*, pages 111–129, 2008.

[41] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *CCS*, pages 195–203, 2007.

[42] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. *Journal of Computer Security*, 18(5):799–837, 2010.

[43] Yannis Rouselakis and Brent Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In *International Conference on Financial Cryptography and Data Security*, pages 315–332, 2015.

[44] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494, pages 457–473, 2005.

[45] Alessandro Sorniotti and Refik Molva. Secret handshakes with revocation support. In *ICISC*, pages 274–299, 2009.

[46] Alessandro Sorniotti and Refik Molva. A provably secure secret handshake with dynamic controlled matching. *Computers & Security*, 29(5):619–627, 2010.

[47] Paul Syverson, R Dingledine, and N Mathewson. Tor: The second generation onion router. In *Usenix Security*, 2004.

[48] Gaosheng Tan, Rui Zhang, Hui Ma, and Yang Tao. Access control encryption based on lwe. In *International Workshop on ASIA Public-Key Cryptography*, pages 43–50, 2017.

[49] Tor. Onion service protocol, 2018. `https://www.torproject.org/docs/onion-services.html.en`.

[50] Gene Tsudik and Shouhuai Xu. A flexible framework for secret handshakes. In *PETS*, pages 295–315, 2006.

[51] Damien Vergnaud. Rsa-based secret handshakes. In *Coding and Cryptography*, pages 252–274. 2006.

[52] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, volume 6571, pages 53–70, 2011.

[53] Shouhuai Xu and Moti Yung. K-anonymous secret handshakes with reusable credentials. In *CCS*, pages 158–167, 2004.

[54] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *PKC*, pages 71–89, 2011.

[55] Shucheng Yu, Kui Ren, and Wenjing Lou. Attribute-based content distribution with hidden policy. In *Secure Network Protocols*, pages 39–44, 2008.

[56] Shucheng Yu, Kui Ren, and Wenjing Lou. Attribute-based on-demand multicast group setup with membership anonymity. *Computer Networks*, 54(3):377–386, 2010.

[57] Shucheng Yu, Kui Ren, Wenjing Lou, and Jin Li. Defending against key abuse attacks in kp-abe enabled broadcast systems. In *SecureComm*, pages 311–329, 2009.

A Supporting Proofs

A.1 Proof of Theorem 1

We use $\mathcal{Q}^{i}_{\mathsf{RKGen}}$ to denote the queries submitted by adversary A_i. We start with showing (q_1, q'_1, q_2, q'_2)-privacy.

Lemma 1. *If* rFE *is* $(q_1, 1, q_2)$*-NA-SIM-secure (Def.4),* FE *is* (q'_1, q_1, q'_2)*-SIM-secure, and* NIZK *is adaptive multi-theorem zero knowledge (Def.11), then the ME scheme* Π *from Construction 1 satisfies* (q_1, q'_1, q_2, q'_2)*-privacy.*

Proof. We use a hybrid argument. Consider the following hybrid experiments:

$\mathsf{Hyb}_0(\lambda)$**:** This is exactly the experiment $\mathbf{G}^{\mathsf{priv}}_{\Pi,\mathsf{A}}(\lambda)$.

$\mathsf{Hyb}_1(\lambda)$**:** Same as Hyb_0, except that the challenger uses the zero-knowledge simulator $\mathsf{Z} = (\mathsf{Z}_0, \mathsf{Z}_1)$ to generate the CRS ω and the proof π contained in the challenge ciphertext. Formally, the challenger runs $(\omega, \zeta) \leftarrow\!\!\$\ \mathsf{Z}_0(1^\lambda)$ at the beginning of the experiment. Thus, when A_1 outputs the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$, the challenger flips a bit $b \leftarrow\!\!\$\ \{0,1\}$ and runs $c \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{rFE}}(\mathsf{mpk}_{\mathsf{rFE}}, (\mathbb{R}_b, \sigma_b, m_b))$, $\pi \leftarrow\!\!\$\ \mathsf{Z}_1(\zeta, (c, \mathsf{pk}, \mathsf{mpk}_{\mathsf{rFE}}))$. Finally, it sets the challenge ciphertext to (c, π).

$\mathsf{Hyb}_2(\lambda)$**:** Same as Hyb_1, except that the challenger uses the simulators $\mathsf{S}^{\mathsf{rFE}} = (\mathsf{S}^{\mathsf{rFE}}_1, \mathsf{S}^{\mathsf{rFE}}_2, \mathsf{S}^{\mathsf{rFE}}_3, \mathsf{S}^{\mathsf{rFE}}_4)$ to generate $\mathsf{mpk}_{\mathsf{rFE}}$ and implement the oracle RKGen. Formally, the challenger runs $(\mathsf{mpk}_{\mathsf{rFE}}, \alpha'_{\mathsf{rFE}}) \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{rFE}}_1(1^\lambda)$ to generate the master public key of rFE, and uses $\mathsf{S}^{\mathsf{rFE}}_2(\alpha'_{\mathsf{rFE}}, \cdot)$ and $\mathsf{S}^{\mathsf{rFE}}_4(\alpha'_{\mathsf{rFE}}, \cdot)$ to answer the queries submitted to RKGen by A_1 and A_2. Finally, when A_1 outputs the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$, the challenger flips a bit $b \leftarrow\!\!\$\ \{0,1\}$ and proceeds as follows:

- For all $\rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}$ such that $\mathbb{R}_b(\rho_i) = 0$, compute c'_i by running $\mathsf{Enc}_{\mathsf{FE}}(\mathsf{mpk}_{\mathsf{FE}}, (\bot, \bot))$
- For all $\rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}$ such that $\mathbb{R}_b(\rho_i) = 1$, compute c'_i by running $\mathsf{Enc}_{\mathsf{FE}}(\mathsf{mpk}_{\mathsf{FE}}, (\sigma_b, m_b))$.

Finally, it returns (c, π) where $c \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{rFE}}_3(\alpha'_{\mathsf{rFE}}, \{c'_i\}_{i\in[q_1]})$ and $\pi \leftarrow\!\!\$\ \mathsf{Z}_1(\zeta, (c, \mathsf{pk}, \mathsf{mpk}_{\mathsf{rFE}}))$.

$\mathsf{Hyb}_3(\lambda)$**:** Same as Hyb_2 except that the challenger uses the FE simulator $\mathsf{S}^{\mathsf{FE}} = (\mathsf{S}^{\mathsf{FE}}_1, \mathsf{S}^{\mathsf{FE}}_2, \mathsf{S}^{\mathsf{FE}}_3, \mathsf{S}^{\mathsf{FE}}_4)$ to generate $\mathsf{mpk}_{\mathsf{FE}}$ and implement the oracle PolGen. Formally, the challenger runs $(\mathsf{mpk}_{\mathsf{FE}}, \alpha'_{\mathsf{FE}}) \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{FE}}_1(1^\lambda)$ to generate the master public key of FE, and uses $\mathsf{S}^{\mathsf{FE}}_2(\alpha'_{\mathsf{FE}}, \cdot)$ and $\mathsf{S}^{\mathsf{FE}}_4(\alpha'_{\mathsf{FE}}, \cdot)$ to answer the queries submitted to RKGen by A_1 and A_2. When A_1 outputs the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$, the challenger flips a bit $b \leftarrow\!\!\$\ \{0,1\}$ and proceeds as follows:

- For all $\rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}, \mathbb{S}_j \in \mathcal{Q}^1_{\mathsf{PolGen}}$ such that $\mathbb{R}_b(\rho_i) = 0 \lor \mathbb{S}_j(\sigma_b) = 0$, set $y_{i,j} = \bot$.
- For all $\rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}, \mathbb{S}_j \in \mathcal{Q}^1_{\mathsf{PolGen}}$ such that $\mathbb{R}_b(\rho_i) = 1 \land \mathbb{S}_j(\sigma_b) = 1$, set $y_{i,j} = m_b$.

Finally, it runs $c'_i \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{FE}}_3(\alpha'_{\mathsf{FE}}, \{y_{i,j}\}_{j\in[q'_1]})$ for all $i \in [q_1]$, $c \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{rFE}}_3(\alpha'_{\mathsf{rFE}}, \{c'_i\}_{i\in[q_1]})$ and $\pi \leftarrow\!\!\$\ \mathsf{Z}_1(\zeta, (c, \mathsf{pk}, \mathsf{mpk}_{\mathsf{rFE}}))$, and returns (c, π) as the challenge ciphertext.

Claim 1. $\{\mathsf{Hyb}_0(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Hyb}_1(\lambda)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the adaptive multi-theorem zero-knowledge property of NIZK. The reduction is standard, so we omit it here. □

Claim 2. $\{\mathsf{Hyb}_1(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Hyb}_2(\lambda)\}_{\lambda\in\mathbb{N}}$.

Proof. Suppose there exists an adversary A that distinguishes between $\mathsf{Hyb}_0(\lambda)$ and $\mathsf{Hyb}_1(\lambda)$ with non-negligible (in λ) probability. We build a distinguisher A' for experiment $\mathbf{REAL}_{\mathsf{rFE},\mathsf{A}'}(\lambda)$ and experiment $\mathbf{IDEAL}_{\mathsf{rFE},\mathsf{A}'}(\lambda)$ with the same probability.

1. At the beginning, A' receives the master public key mpk^*. Then, it runs $(\mathsf{mpk}_{\mathsf{FE}}, \mathsf{msk}_{\mathsf{FE}}) \leftarrow\$ \mathsf{Setup}_{\mathsf{FE}}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$, and $(\omega, \zeta) \leftarrow\$ \mathsf{Z}_0(1^\lambda)$. Finally, A' sends $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{mpk}_{\mathsf{FE}}, \mathsf{mpk}^*)$ to A.

2. A' answers oracle queries in the following way:

   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, compute $s = \mathsf{Sign}(\mathsf{sk}, \sigma)$ and return $\mathsf{ek}_\sigma = (\sigma, s)$.
   - Upon input $\rho \in \{0,1\}^*$ for RKGen, send $(\rho, \mathsf{mpk}_{\mathsf{FE}})$ to the key generation oracle $\mathsf{O}_1^{\mathsf{rFE}}$ and return the output.
   - Upon input $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for PolGen, return $\mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, \mathbb{S})$.

3. Receive the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$ chosen by A. Set $m_0^* = (m_0, \mathbb{R}_0, \sigma_0)$ and $m_1^* = (m_1, \mathbb{R}_1, \sigma_1)$. Finally, it sends to the challenger m_b^*, where $b \leftarrow\$ \{0,1\}$.

4. Receive c and return (c, π) where $\pi \leftarrow\$ \mathsf{Z}_1(\zeta, (c, \mathsf{pk}, \mathsf{mpk}^*))$.

5. Answer the incoming queries as in step 2.

6. Finally, A' outputs whatever A outputs.

First, note that A submits q_1 and q_2 queries to RKGen (because, she is playing an hybrid version of game (q_1, q_1', q_2, q_2')-privacy). Hence, A' is a valid adversary for (q_1, q_2)-NA-SIM security game. Second, we claim that if A' is playing, respectively, the experiment $\mathbf{REAL}_{\mathsf{rFE},\mathsf{A}'}(\lambda)$ and $\mathbf{IDEAL}_{\mathsf{rFE},\mathsf{A}'}(\lambda)$, then the reduction perfectly simulates game $\mathsf{Hyb}_1(\lambda)$ and $\mathsf{Hyb}_2(\lambda)$. The latter is because, in experiment $\mathbf{REAL}_{\mathsf{rFE},\mathsf{A}'}(\lambda)$, the attacker A' answers to the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$ with (c, π) where π is the NIZK proof simulated by $\mathsf{Z}(\zeta, (c, \mathsf{pk}, \mathsf{mpk}^*))$, and c is the output of $\mathsf{S}_3^{\mathsf{rFE}}(\alpha'_{\mathsf{rFE}}, \{c_i'\})$ where the ciphertexts $\{c_i'\}$ are distributed exactly as in $\mathsf{Hyb}_2(\lambda)$. Additionally, mpk^* is generated by $\mathsf{S}_1^{\mathsf{rFE}}$ and the outputs of oracle RKGen are simulated using $\mathsf{S}_2^{\mathsf{rFE}}$ and $\mathsf{S}_4^{\mathsf{rFE}}$, as it happens in $\mathsf{Hyb}_2(\lambda)$. On the other hand, in the experiment $\mathbf{IDEAL}_{\mathsf{rFE},\mathsf{A}'}(\lambda)$, the attacker A' answers using the real rFE algorithms. This concludes the proof. □

Claim 3. $\{\mathsf{Hyb}_2(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Hyb}_3(\lambda)\}_{\lambda\in\mathbb{N}}$.

Proof. Suppose there exists an adversary A that distinguishes between $\mathsf{Hyb}_2(\lambda)$ and $\mathsf{Hyb}_3(\lambda)$ with non-negligible (in λ) probability. Then, we build a distinguisher A' from experiments $\mathbf{REAL}_{\mathsf{FE},\mathsf{A}'}(\lambda)$ and $\mathbf{IDEAL}_{\mathsf{FE},\mathsf{A}'}(\lambda)$ with the same probability.

1. At the beginning, A' receives the master public key mpk^*. Then, it runs $(\mathsf{mpk}_{\mathsf{rFE}}, \alpha'_{\mathsf{rFE}}) \leftarrow\$ \mathsf{S}_1^{\mathsf{rFE}}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$, and $(\omega, \zeta) \leftarrow\$ \mathsf{Z}_0(1^\lambda)$. Finally, A' sends $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{mpk}^*, \mathsf{mpk}_{\mathsf{rFE}})$ to A.

2. A' answers oracle queries in the following way:

   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, compute $s = \mathsf{Sign}(\mathsf{sk}, \sigma)$ and return $\mathsf{ek}_\sigma = (\sigma, s)$.
   - Upon input $\rho \in \{0,1\}^*$ for RKGen, answer with $\mathsf{S}_2^{\mathsf{rFE}}(\alpha'_{\mathsf{rFE}}, (\rho, \mathsf{mpk}^*))$.
   - Upon input $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for PolGen, send $\mathbb{S}$ to the key generation oracle $\mathsf{O}_1^{\mathsf{FE}}$ and return the output.

3. Receive the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$. A' flips a bit $b \leftarrow\!\!\$\ \{0,1\}$ and proceeds as follows: $\forall \rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}$, $\mathbb{R}_b(\rho_i) = 0$, set $m^*_i = (\bot, \bot)$. Otherwise, set $m^*_i = (\sigma_b, m_b)$. Send $(m^*_i)_{i\in[q_1]}$ to the challenger.

4. Receive $\{c^*_i\}_{i\in[q_1]}$ and return (c, π) where $c \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{rFE}}_3(\alpha'_{\mathsf{rFE}}, \{c^*_i\}_{i\in[q_1]})$, and $\pi \leftarrow\!\!\$\ \mathsf{Z}_1(\zeta, (c, \mathsf{pk}, \mathsf{mpk}^*))$.

5. Answer the incoming queries as in step 2.

6. Finally, A' outputs whatever A outputs.

Note that A submits $q'_1 + q'_2$ queries to PolGen (q'_1 before seeing the challenge ciphertext and q'_2 afterwards). Hence, A' is a valid adversary for the (q'_1, q_1, q'_2)-SIM security game. Thus, a similar analysis as in the proof of Claim 2 allows us to conclude that $\mathsf{Hyb}_2(\lambda)$ and $\mathsf{Hyb}_3(\lambda)$ are computationally indistinguishable. □

Let $\mathsf{Hyb}_3(\lambda, b)$ be the hybrid game $\mathsf{Hyb}_3(\lambda)$ conditioned on the challenge bit being equal to b. We claim that $\mathsf{Hyb}_3(\lambda, 0) \equiv \mathsf{Hyb}_3(\lambda, 1)$. To see this, first note that since A is valid (cf. Def. 14), it must be the case that for all $\rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}$, $\mathbb{S}_j \in \mathcal{Q}^1_{\mathsf{PolGen}}$

$$(\mathbb{R}_0(\rho_i) = 0 \vee \mathbb{S}_j(\sigma_0) = 0) = (\mathbb{R}_1(\rho_i) = 0 \vee \mathbb{S}_j(\sigma_1) = 0), \text{ and} \tag{3}$$

$$(\mathbb{R}_0(\rho_i) = 1 \wedge \mathbb{S}_j(\sigma_0) = 1) = (\mathbb{R}_1(\rho_i) = 1 \wedge \mathbb{S}_j(\sigma_1) = 1). \tag{4}$$

In fact, from the definition of valid adversary, either for all $\rho_i \in \mathcal{Q}^1_{\mathsf{RKGen}}$, $\mathbb{S}_j \in \mathcal{Q}^1_{\mathsf{PolGen}}$, Eq. (1) is satisfied—in which case also Eq. (3) and Eq. (4) hold true— or if at least one match occurs (i.e., if there exists $\hat{\rho}_i \in \mathcal{Q}^1_{\mathsf{RKGen}}, \hat{\mathbb{S}}_j \in \mathcal{Q}^1_{\mathsf{PolGen}}$ such that Eq. (1) does not hold), we know that $m_0 = m_1$, $\mathbb{R}_0(\rho_i) = \mathbb{R}_1(\rho_i)$ and $\mathbb{S}_j(\sigma_0) = \mathbb{S}_j(\sigma_1)$, which again implies Eq. (3) and Eq. (4).

Finally, note that the only dependency on the bit b in hybrid $\mathsf{Hyb}_3(\lambda, b)$ comes from the definitions of the values $y_{i,j}$. However, by Eq. (3) and Eq. (4), we observe that:

- Whenever $\mathbb{R}_b(\rho_i) = 0 \vee \mathbb{S}_j(\sigma_b) = 0$ it also holds that $\mathbb{R}_{1-b}(\rho_i) = 0 \vee \mathbb{S}_j(\sigma_{1-b}) = 0$, so that both hybrids set $y_{i,j} = \bot$;
- Whenever $\mathbb{R}_b(\rho_i) = 1 \wedge \mathbb{S}_j(\sigma_b) = 1$ it also holds that $\mathbb{R}_{1-b}(\rho_i) = 1 \wedge \mathbb{S}_j(\sigma_{1-b}) = 1$ and $m_0 = m_1$, so that both hybrids set $y_{i,j} = m_0 = m_1$.

Hence, $\mathsf{Hyb}_3(\lambda, 0)$ and $\mathsf{Hyb}_3(\lambda, 1)$ are identically distributed.

Combining Claims 1–3, we conclude that Construction 1 satisfies (q_1, q'_1, q_2, q'_2)-privacy. □

Lemma 2. *If* SS *is EUF-CMA (Def. 2), and* NIZK *has knowledge soundness (Def. 12), then the ME scheme* Π *from Construction 1 satisfies authenticity (Def. 15).*

Proof. By contradiction, assume Construction 1 does not satisfies authenticity, i.e., there exists an attacker A that has a non negligible advantage in experiment $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{A}}(\lambda)$. We build an attacker A' that breaks unforgeability of SS. A proceeds as follows:

1. Receive pk^* from the challenger.

2. Run $(\mathsf{mpk}_{\mathsf{rFE}}, \mathsf{msk}_{\mathsf{rFE}}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathsf{rFE}}(1^\lambda)$, $(\mathsf{mpk}_{\mathsf{FE}}, \mathsf{msk}_{\mathsf{FE}}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathsf{FE}}(1^\lambda)$, $(\omega, \xi) \leftarrow\!\!\$\ \mathsf{K}_0(1^\lambda)$, and send $\mathsf{mpk} = (\mathsf{pk}^*, \omega, \mathsf{mpk}_{\mathsf{FE}}, \mathsf{mpk}_{\mathsf{rFE}})$ to A.

3. Answer the incoming A's queries in the following way:
   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, forward the query to oracle Sign obtaining s as answer, and return (σ, s).

- Upon input $\rho \in \{0,1\}^*$ for RKGen, compute and return the decryption key $\mathsf{dk}_\rho \leftarrow\$ \mathsf{KGen}_{\mathsf{rFE}}(\mathsf{msk}_{\mathsf{rFE}}, (\rho, \mathsf{mpk}_{\mathsf{FE}}))$.
- Upon input $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for PolGen, compute and return the policy key $\mathsf{dk}_{\mathbb{S}} \leftarrow\$ \mathsf{Enc}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, \mathbb{S})$.

4. Upon the forgery $((c^*, \pi^*), \rho^*, \mathbb{S}^*)$ from A, compute dk_{ρ^*} and $\mathsf{dk}_{\mathbb{S}^*}$ by running $\mathsf{KGen}_{\mathsf{rFE}}(\mathsf{msk}_{\mathsf{rFE}}, (\rho^*, \mathsf{mpk}_{\mathsf{FE}}))$ and $\mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, \mathbb{S}^*)$. If either $\mathsf{V}(\omega, (c^*, \mathsf{pk}^*, \mathsf{mpk}_{\mathsf{rFE}}), \pi^*) = 0$, or $\mathsf{Dec}_{\mathsf{rFE}}(\mathsf{dk}_{\mathbb{S}^*}, \mathsf{Dec}_{\mathsf{rFE}}(\mathsf{dk}_{\rho^*}, c^*)) = \perp$, it abort. Else, run and return $(\sigma^*, s^*) \leftarrow\$ \mathsf{K}_1(\xi, (c^*, \mathsf{pk}^*, \mathsf{mpk}_{\mathsf{rFE}}), \pi^*)$ as forgery to the challenger.

We now show that the simulation is perfect, except with negligibly small probability. First, note that conditioned on $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{mpk}_{\mathsf{rFE}}, \mathsf{msk}_{\mathsf{rFE}}, \mathsf{mpk}_{\mathsf{FE}}, \mathsf{msk}_{\mathsf{FE}}, \omega)$, then $\forall \sigma, \rho \in \{0,1\}^*, \forall \mathbb{S} : \{0,1\}^* \to \{0,1\}$ the oracle queries of A are perfectly simulated by A', and the only difference is that the CRS ω is computed via K_0 in the reduction, but this distribution is computationally close to that of an honestly generated CRS. This means that with non-negligible probability the ciphertext (c^*, π^*) returned by A is valid, which implies that the proof π^* verifies correctly, and moreover $\mathsf{Dec}_{\mathsf{rFE}}(\mathsf{dk}_{\mathbb{S}^*}, \mathsf{Dec}_{\mathsf{rFE}}(\mathsf{dk}_{\rho^*}, c^*)) \neq \perp$ (so c^* is also a valid ciphertext).

Now, by knowledge soundness of the NIZK proof, except with negligible probability, we must have that the witness (σ^*, s^*) computed by the extractor is such that $((c^*, \mathsf{pk}^*, \mathsf{mpk}_{\mathsf{rFE}}), (\sigma^*, s^*)) \in R_2$, which implies that s^* is a valid signature on σ^* w.r.t. public key pk^*. Finally, in $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{A}}(\lambda)$ none of the query in $\mathcal{Q}_{\mathsf{SKGen}}$ satisfies the policy $\mathbb{S}^*$. Thus, σ^* has not been queried to Sign (i.e., $\sigma^* \notin \mathcal{Q}_{\mathsf{Sign}}$), and A' wins with non-negligible probability. □

Finally, combining Lemmas 1–2, we conclude that Construction 1 is (q_1, q_1', q_2, q_2')-secure.

A.2 Proof of Theorem 2

Proof. We prove privacy and authenticity separately.

Lemma 3. *If* $\mathsf{2FE}$ *is indistinguishably secure in the* 1*-semiprivate setting (Def. 9), and* NIZK *is adaptive multi-theorem zero knowledge (Def. 11), then the ME scheme* Π *from Construction 2 satisfies privacy (Def. 14).*

Proof. We use a hybrid argument. Consider the following hybrid experiments:

$\mathsf{Hyb}_0(\lambda)$**:** This is exactly the experiment $\mathbf{G}^{\mathsf{priv}}_{\Pi,\mathsf{A}}(\lambda)$.

$\mathsf{Hyb}_1(\lambda)$**:** Same as $\mathsf{Hyb}_0(\lambda)$, except that the challenger uses the zero-knowledge simulator $\mathsf{Z} = (\mathsf{Z}_0, \mathsf{Z}_1)$ to generate the CRS ω and the proof π contained in the challenge ciphertext. Formally, the challenger runs $(\omega, \zeta) \leftarrow\$ \mathsf{Z}_0(1^\lambda)$ at the beginning of the experiment. Thus, when A_1 outputs the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$, the challenger computes $c_0 \leftarrow\$ \mathsf{Enc}_{\mathsf{2FE}}(\mathsf{ek}_0, (\mathbb{R}_b, \sigma_b, m_b))$, $\pi \leftarrow\$ \mathsf{Z}_1(\zeta, (c_0, \mathsf{pk}, \mathsf{ek}_0))$, and sets the challenge ciphertext to (c_0, π).

Claim 4. $\{\mathsf{Hyb}_0(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Hyb}_1(\lambda)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the adaptive multi-theorem zero-knowledge property of NIZK. The reduction is standard, so we omit it here. □

Claim 5. *If* $\mathsf{2FE}$ *is indistinguishably secure in the* 1*-semi-private model (Def. 9), then for all PPT valid adversaries* A*:* $|\Pr[\mathsf{Hyb}_1(\lambda) = 1] - 1/2| \leq \mathsf{negl}(\lambda)$.

Proof. Suppose that there exists a valid adversary A that has non-negligible advantage in $\mathsf{Hyb}_1(\lambda)$. We build an attacker A' that breaks security of experiment $\mathbf{G}^{\mathsf{spriv}}_{\mathsf{2FE},\mathsf{A}'}(\lambda, 1)$. A' proceeds as follows:

1. At the beginning, receive ek_0^* sampled by the challenger. Then, it runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\, \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$ and $(\omega, \zeta) \leftarrow\!\!\$\, \mathsf{Z}_0(1^\lambda)$. Finally, A' sends $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{ek}_0^*)$ to A.
2. A' answers the incoming oracle queries from A in the following way:
   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, return (σ, s) where $s \leftarrow\!\!\$\, \mathsf{Sign}(\mathsf{sk}, \sigma)$.
   - Upon input $\rho \in \{0,1\}^*$ for RKGen, send ρ to oracle $\mathsf{KGen}_{\mathsf{2FE}}$ and return the corresponding output.
   - Upon input $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for PolGen, send $\mathbb{S}$ to oracle $\mathsf{Enc}_{\mathsf{2FE}}$ and return the corresponding output.
3. Receive the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$ chosen by A. Select any policy $\mathbb{S}^* \in \mathcal{Q}_{\mathsf{PolGen}}$, i.e., any policy submitted by A to oracle PolGen. Set $m_0^0 = (\mathbb{R}_0, \sigma_0, m_0)$, $m_0^1 = (\mathbb{R}_1, \sigma_1, m_1)$ and $m_1^0 = m_1^1 = \mathbb{S}^*$. Send the challenge $(m_0^0, m_1^0), (m_0^1, m_1^1)$ to the challenger.
4. After receiving the ciphertexts (c_0^*, c_1^*) from the challenger, A computes $\pi \leftarrow\!\!\$\, \mathsf{Z}_1(\zeta, (c_0^*, \mathsf{pk}, \mathsf{ek}_0))$. Finally, send (c_0^*, π) to A.
5. Simulate oracle queries as in step 3.
6. Return the output of A.

We now show that the simulation is perfect. Conditioned on msk^*, ek_0^*, ek_1^* sampled by the challenger and sk, pk, ω, ζ generated by A', $\forall \sigma, \rho \in \{0,1\}^*$, $\forall \mathbb{S} : \{0,1\}^* \to \{0,1\}$ the following holds:

$$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma) = (\sigma, \mathsf{Sign}(\mathsf{sk}, \sigma))$$

$$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho) = \mathsf{KGen}_{\mathsf{2FE}}(\mathsf{msk}^*, \rho)$$

$$\mathsf{PolGen}(\mathsf{mpk}, \mathsf{kpol}, \mathbb{S}) = \mathsf{Enc}_{\mathsf{2FE}}(\mathsf{ek}_1^*, \mathbb{S})$$

where $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{ek}_0^*)$, $\mathsf{kpol} = \mathsf{ek}_1^*$, $\mathsf{msk} = (\mathsf{msk}^*, \mathsf{sk})$. This suffices to conclude that the queries' answers have the same distribution of what A expects to receive. Moreover, A is a valid adversary (cf. Def. 14) in $\mathsf{Hyb}_1(\lambda)$. This allows us to conclude that $\forall \mathbb{S}' \in \mathcal{Q}_{\mathsf{Enc}_{\mathsf{2FE}}} \cup \{m_1^0\}$ (recall $m_1^0 = m_1^1 = \mathbb{S}^* \in \mathcal{Q}_{\mathsf{PolGen}}$), and $\forall \rho \in \mathcal{Q}_{\mathsf{KGen}_{\mathsf{2FE}}}$, the following invariant is maintained:

- **(Mismatch condition)** Either $f_\rho(m_0^0, \mathbb{S}') = f_\rho(m_0^1, \mathbb{S}') = \perp$;
- **(Match condition)** Or $f_\rho(m_0^0, \mathbb{S}') = f_\rho(m_0^1, \mathbb{S}') \in \{\perp, m_0\}$ (recall $m_0 = m_1$ in this case).

Finally, it is clear that there does not exist any 1-st position input such that $f_\rho(\cdot, m_1^0) \neq f_\rho(\cdot, m_1^1)$ (since $m_1^0 = m_1^1 = \mathbb{S}^*$). Thus, A' is a valid adversary for $\mathbf{G}^{\mathsf{spriv}}_{\mathsf{2FE},\mathsf{A}'}(\lambda, 1)$ and has the same advantage of A. □

Combining Claims 4–5, we obtain that Construction 2 satisfies privacy. □

Lemma 4. *If* SS *is EUF-CMA (Def. 2), and* NIZK *has knowledge soundness (Def. 12), then the ME scheme* Π *from Construction 2 satisfies authenticity (Def. 15).*

Proof. By contradiction, assume Construction 2 does not satisfies authenticity, i.e., there exists an attacker A that has a non negligible advantage in experiment $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{A}}(\lambda)$. We build an attacker A' that breaks unforgeability of SS. A proceeds as follows:

1. Receive pk^* from the challenger.

2. Execute $(\mathsf{msk_{2FE}}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow\$ \mathsf{Setup_{2FE}}(1^\lambda)$, $(\omega, \xi) \leftarrow\$ \mathsf{K}_0(1^\lambda)$, and send $\mathsf{mpk} = (\mathsf{pk}^*, \omega, \mathsf{ek}_0)$ to A.

3. Answer the incoming A's queries in the following way:
   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, forward the query to oracle Sign obtaining s as answer, and return (σ, s).
   - Upon input $\rho \in \{0,1\}^*$ for RKGen, compute and return the decryption key $\mathsf{dk}_\rho \leftarrow\$ \mathsf{KGen_{2FE}}(\mathsf{msk_{2FE}}, \rho)$.
   - Upon input $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for PolGen, compute and return the policy key $\mathsf{dk}_\mathbb{S} \leftarrow\$ \mathsf{Enc_{2FE}}(\mathsf{ek}_1, \mathbb{S})$.

4. Receive the forgery $((c_0^*, \pi^*), \rho^*, \mathbb{S}^*)$ from A and proceed as follows:
   - Compute $\mathsf{dk}_{\rho^*} \leftarrow\$ \mathsf{KGen_{2FE}}(\mathsf{msk_{2FE}}, \rho^*)$ and $c_1^* \leftarrow\$ \mathsf{Enc_{2FE}}(\mathsf{ek}_1, \mathbb{S}^*)$.
   - If either $\mathsf{V}(\omega, (c_0^*, \mathsf{pk}^*, \mathsf{ek}_0), \pi^*) = 0$, or $\mathsf{Dec_{2FE}}(\mathsf{dk}_{\rho^*}, c_0^*, c_1^*) = \bot$, abort.
   - Else, run $(\sigma^*, s^*) \leftarrow\$ \mathsf{K}_1(\xi, (c_0^*, \mathsf{pk}^*, \mathsf{ek}_0), \pi^*)$ and return (σ^*, s^*) as forgery to the challenger.

We now show that the simulation is perfect, except with negligibly small probability. First, note that conditioned on $(\mathsf{pk}^*, \mathsf{sk}^*)$ chosen by the challenger, and $(\mathsf{ek}_0, \mathsf{ek}_1, \mathsf{msk_{2FE}}, \omega)$ generated by A', the following holds $\forall \sigma, \rho \in \{0,1\}^*$:

$$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma) = (\sigma, \mathsf{Sign}(\mathsf{sk}^*, \sigma))$$

$$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho) = \mathsf{KGen_{2FE}}(\mathsf{msk_{2FE}}, \rho)$$

$$\mathsf{PolGen}(\mathsf{mpk}, \mathsf{kpol}, \mathbb{S}) = \mathsf{Enc_{2FE}}(\mathsf{ek}_1, \mathbb{S}).$$

Hence, the oracle queries of A are perfectly simulated by A', and the only difference is that the CRS ω is computed via K_0 in the reduction, but this distribution is computationally close to that of an honestly generated CRS. This means that with non-negligible probability the ciphertext (c_0^*, π^*) returned by A is valid, which implies that the proof π^* verifies correctly, and moreover $\mathsf{Dec_{2FE}}(\mathsf{dk}_{\rho^*}, c_0^*, c_1^*) \neq \bot$ (so c_0^* is also a valid ciphertext).

Now, by knowledge soundness of the NIZK proof, except with negligible probability, we must have that the witness (σ^*, s^*) computed by the extractor is such that $((c_0^*, \mathsf{pk}^*, \mathsf{ek}_0), (\sigma^*, s^*)) \in R_2$, which implies that s^* is a valid signature on σ^* w.r.t. public key pk^*. Finally, in $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{auth}}(\lambda)$ none of the query in $\mathcal{Q}_{\mathsf{SKGen}}$ satisfies the policy $\mathbb{S}^*$. Thus, σ^* has not been queried to Sign (i.e., $\sigma^* \notin \mathcal{Q}_{\mathsf{Sign}}$), and A' wins with non-negligible probability. □

By combining Lemmas 3–4 we obtain that Construction 2 is secure. □

A.3 Proof of Theorem 3

Proof. As usual, we prove privacy and authenticity separately.

Lemma 5. *If* FE *is secure (Def. 7), and* NIZK *is adaptive multi-theorem zero knowledge (Def. 11), then the A-ME scheme* Π *from Construction 3 satisfies privacy (Def. 18).*

Proof. We use a hybrid argument. Consider the following hybrid experiments:

$\mathsf{Hyb}_0(\lambda)$**:** This is exactly the experiment $\mathbf{G}_{\Pi,\mathsf{A}}^{\mathsf{arr\text{-}priv}}(\lambda)$.

$\mathsf{Hyb}_1(\lambda)$**:** Same as $\mathsf{Hyb}_0(\lambda)$, except that the challenger uses the zero-knowledge simulator $\mathsf{Z} = (\mathsf{Z}_0, \mathsf{Z}_1)$ to generate the CRS ω and the proof π contained in the challenge ciphertext. Formally, the challenger runs $(\omega, \zeta) \leftarrow\!\!\$\ \mathsf{Z}_0(1^\lambda)$ at the beginning of the experiment. Thus, when A_1 outputs the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$, the challenger sets the challenge ciphertext to (c, π) where $c \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{FE}}(\mathsf{mpk}_{\mathsf{FE}}, (\mathbb{R}_b, \sigma_b, m_b))$, and $\pi \leftarrow\!\!\$\ \mathsf{Z}_1(\zeta, (c, \mathsf{pk}, \mathsf{mpk}_{\mathsf{FE}}))$.

Claim 6. $\{\mathsf{Hyb}_0(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Hyb}_1(\lambda)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the adaptive multi-theorem zero-knowledge property of NIZK. The reduction is standard, so we omit it here. □

Claim 7. *If* FE *is secure (Def. 7), then for all PPT adversaries* A*:* $|\Pr[\mathsf{Hyb}_1(\lambda) = 1] - 1/2| \leq \mathsf{negl}(\lambda)$*.*

Proof. Suppose that there exists an adversary A that has non-negligible advantage in $\mathsf{Hyb}_1(\lambda)$. We build an attacker A′ that breaks security of experiment $\mathbf{G}^{\mathsf{fe}}_{\mathsf{FE},\mathsf{A}'}(\lambda)$. A′ proceeds as follows:

1. At the beginning, receive mpk^* sampled by the challenger. Then, it runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SS}}(1^\lambda)$ and $(\omega, \zeta) \leftarrow\!\!\$\ \mathsf{Z}_0(1^\lambda)$. Finally, A′ sends $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{mpk}^*)$ to A.
2. A′ answers the incoming oracle queries from A in the following way:
   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, return (σ, s) where $s = \mathsf{Sign}(\mathsf{sk}, \sigma)$.
   - Upon input $\rho \in \{0,1\}^*$ and $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for RKGen, send $(\rho, \mathbb{S})$ to oracle $\mathsf{KGen}_{\mathsf{FE}}$ and return the corresponding output.
3. Receive the challenge $(m_0, m_1, \mathbb{R}_0, \mathbb{R}_1, \sigma_0, \sigma_1)$ chosen by A Set $m_0^* = (\mathbb{R}_0, \sigma_0, m_0)$, $m_0^* = (\mathbb{R}_1, \sigma_1, m_1)$. Send the challenge (m_0^*, m_1^*) to the challenger.
4. After receiving the ciphertext c^* from the challenger, A computes $\pi \leftarrow\!\!\$\ \mathsf{Z}_1(\zeta, (c^*, \mathsf{pk}, \mathsf{mpk}))$. Finally, send (c^*, π) to A.
5. Simulate oracle queries as in step 3.
6. Return the output of A.

We now show that the simulation is perfect. Conditioned on msk^*, msk_0^* sampled by the challenger and sk, pk, ω, ζ generated by A′, $\forall \sigma, \rho \in \{0,1\}^*$, $\forall \mathbb{S} : \{0,1\}^* \to \{0,1\}$ the following holds:

$$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma) = (\sigma, \mathsf{Sign}(\mathsf{sk}, \sigma))$$

$$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho, \mathbb{S}) = \mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}^*, (\rho, \mathbb{S}))$$

where $\mathsf{mpk} = (\mathsf{pk}, \omega, \mathsf{mpk}^*)$, $\mathsf{msk} = (\mathsf{msk}^*, \mathsf{sk})$. This suffices to conclude that the queries' answers have the same distribution of what A expects to receive. Moreover, A is a valid adversary (cf. Def. 18) in $\mathsf{Hyb}_1(\lambda)$. This allows us to conclude that $\forall (\rho, \mathbb{S}) \in \mathcal{Q}_{\mathsf{KGen}_{\mathsf{FE}}}$, the following invariant is maintained:

- **(Mismatch condition)** Either $f_{(\rho,\mathbb{S})}(m_0^*) = f_{(\rho,\mathbb{S})}(m_1^*) = \bot$;
- **(Match condition)** Or $f_{(\rho,\mathbb{S})}(m_0^*) = f_{(\rho,\mathbb{S})}(m_1^*) \in \{\bot, m_0\}$ (recall $m_0 = m_1$ in this case).

Thus, A′ has the same advantage of A and is a valid adversary for $\mathbf{G}^{\mathsf{fe}}_{\mathsf{FE},\mathsf{A}'}(\lambda)$. □

Combining Claim 6– 7, we obtain that Construction 3 satisfies privacy. □

Lemma 6. *If* SS *is EUF-CMA (Def. 2), and* NIZK *has knowledge soundness (Def. 12), then the A-ME scheme* Π *from Construction 3 satisfies authenticity (Def. 19).*

Proof. By contradiction, assume that Construction 3 does not satisfies authenticity, i.e. there exists an attacker A that has a non-negligible advantage in experiment $\mathbf{G}^{\mathsf{arr\text{-}auth}}_{\Pi,\mathsf{A}}(\lambda)$. We build an attacker A' that breaks unforgeability of SS as follows:

1. Receive pk^* from the challenger.
2. Execute $(\mathsf{msk}_{\mathsf{FE}}, \mathsf{mpk}_{\mathsf{FE}}) \leftarrow\$\ \mathsf{Setup}_{\mathsf{FE}}(1^\lambda)$, $(\omega, \xi) \leftarrow\$\ \mathsf{K}_0(1^\lambda)$, and send $\mathsf{mpk} = (\mathsf{pk}^*, \omega, \mathsf{mpk}_{\mathsf{FE}})$ to A.
3. Answer the incoming A's queries in the following way:
   - Upon input $\sigma \in \{0,1\}^*$ for SKGen, forward the query to oracle Sign obtaining s as answer, and return (σ, s).
   - Upon input $\rho \in \{0,1\}^*$ and $\mathbb{S} : \{0,1\}^* \to \{0,1\}$ for RKGen, compute and return the decryption key $\mathsf{dk}_{(\rho,\mathbb{S})} \leftarrow\$\ \mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, (\rho, \mathbb{S}))$.
4. Receive the forgery $((c^*, \pi^*), \rho^*, \mathbb{S}^*)$ from A and proceed as follows:
   - Compute $\mathsf{dk}_{\rho^*,\mathbb{S}^*} \leftarrow\$\ \mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, (\rho^*, \mathbb{S}^*))$.
   - If either $\mathsf{V}(\omega, (c^*, \mathsf{pk}^*, \mathsf{mpk}_{\mathsf{FE}}), \pi^*) = 0$, or $\mathsf{Dec}_{\mathsf{FE}}(\mathsf{dk}_{\rho^*,\mathbb{S}^*}, c^*) = \bot$, abort.
   - Else, run $(\sigma^*, s^*) \leftarrow\$\ \mathsf{K}_1(\xi, (c^*, \mathsf{pk}^*, \mathsf{mpk}_{\mathsf{FE}}), \pi^*)$ and return (σ^*, s^*) as forgery to the challenger.

We now show that the simulation is perfect, except with negligibly small probability. First, note that conditioned on $(\mathsf{pk}^*, \mathsf{sk}^*)$ chosen by the challenger, and $(\mathsf{mpk}_{\mathsf{FE}}\mathsf{msk}_{\mathsf{FE}}, \omega)$ generated by A', the following holds $\forall \sigma, \rho \in \{0,1\}^*$:

$$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma) = (\sigma, \mathsf{Sign}(\mathsf{sk}^*, \sigma))$$

$$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho, \mathbb{S}) = \mathsf{KGen}_{\mathsf{FE}}(\mathsf{msk}_{\mathsf{FE}}, (\rho, \mathbb{S}))$$

Hence, the oracle queries of A are perfectly simulated by A', and the only difference is that the CRS ω is computed via K_0 in the reduction, but this distribution is computationally close to that of an honestly generated CRS. This means that with non-negligible probability the ciphertext (c^*, π^*) returned by A is valid, which implies that the proof π^* verifies correctly, and moreover $\mathsf{Dec}_{\mathsf{FE}}(\mathsf{dk}_{\rho^*,\mathbb{S}^*}, c) \neq \bot$ (so c^* is also a valid ciphertext).

Now, by knowledge soundness of the NIZK proof, except with negligible probability, we must have that the witness (σ^*, s^*) computed by the extractor is such that $((c^*, \mathsf{pk}^*, \mathsf{mpk}_{\mathsf{FE}}), (\sigma^*, s^*)) \in R_3$, which implies that s^* is a valid signature on σ^* w.r.t. public key pk^*. Finally, in $\mathbf{G}^{\mathsf{arr\text{-}auth}}_{\Pi,\mathsf{A}}(\lambda)$ none of the query in $\mathcal{Q}_{\mathsf{SKGen}}$ satisfies the policy $\mathbb{S}^*$. Thus, σ^* has not been queried to Sign (i.e., $\sigma^* \notin \mathcal{Q}_{\mathsf{Sign}}$), and A' wins with non-negligible probability. □

By combining Lemmas 5–6 we obtain that Construction 3 is secure. □

A.4 Proof of Theorem 4

We start with showing privacy.

Lemma 7. *Let* A *be an adversary that breaks the privacy property of Construction 4 with advantage* ε*, while asking at most* q_R *queries to the decryption key oracle* RKGen *and* $q_{\hat{H}}$ *queries to the random oracle* $\hat{H}$*. Then, there is an algorithm that solves the BDH problem with advantage* $8\varepsilon/e^2(q_R+2)^2 q_{\hat{H}}$.

There are many similarities between our scheme and the Boneh-Franklin CPA-secure IBE [12]. Their proof uses an intermediate PKE scheme called $\mathsf{BasicPub}$ to simplify the security analysis. More in details, they first show that their IBE scheme is CPA-secure if $\mathsf{BasicPub}$ is CPA-secure [12, Lemma 4.2], and then, they show that if the BDH assumption holds then $\mathsf{BasicPub}$ is CPA-secure [12, Lemma 4.3].

We will follow a similar tactic, defining two games that in the end prove that our IB-ME satisfies privacy under the BDH assumption. First, we define $\mathsf{BasicPub}^+$, a variant of $\mathsf{BasicPub}$ more suitable for our needs. $\mathsf{BasicPub}^+$ is composed of the following algorithms:

$\mathsf{Setup}(1^\lambda)$**:** Generate a symmetric pairing $e : \mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$, with $\mathbb{G}$, and $\mathbb{G}_T$ of an order q that depends on λ. Choose a random generator P of $\mathbb{G}$. Sample a random $r\in\mathbb{Z}_q$ and set $P_0 = P^r$. Choose a key derivation function $\hat{H} : \mathbb{G}\to\{0,1\}^n$, for some n. The master public key is the tuple $\mathsf{mpk} = (q,\mathbb{G},\mathbb{G}_T,e,n,P,P_0,\hat{H})$. The master secret key is $\mathsf{msk} = r$.

$\mathsf{KGen}(\mathsf{mpk},\mathsf{msk})$**:** Choose a random $G\in\mathbb{G}$. The public key is $\mathsf{pk} = G$. The private key is $\mathsf{sk} = G^r$.

$\mathsf{Enc}(\mathsf{mpk},\mathsf{pk},m)$**:** To encrypt a message m under public key $\mathsf{pk} = G$, choose a random $x\in\mathbb{Z}_q$ and output $c = (U,V) = (P^x, m\oplus\hat{H}(e(G,P_0)^x))$.

$\mathsf{Dec}(\mathsf{mpk},\mathsf{sk},c)$**:** Let $c = (U,V)$ be a ciphertext for public key pk, then the algorithm returns $m = V\oplus\hat{H}(e(\mathsf{sk},U))$.

In the first game, described in Claim 8, we show that if $\mathsf{BasicPub}^+$ is IND-CPA$^+$-secure, then our scheme satisfies privacy. In order to be compatible with our definition of privacy, we define IND-CPA$^+$ security as a variant of traditional IND-CPA where the adversary not only inputs a pair of messages m_0 and m_1, but also a pair of (honestly generated) public keys pk_{j_0} and pk_{j_1}. Thus, the experiment can be seen as a hybrid between the typical IND-CPA game and the key-privacy game for PKE defined by Bellare *et al.* [10].

Definition 25 (IND-CPA$^+$)**.** *A public-key encryption scheme* $\Pi = (\mathsf{Setup},\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$ *is IND-CPA*$^+$ *secure if for all PPT adversaries* $\mathsf{A} = (\mathsf{A}_1,\mathsf{A}_2)$*, we have that*

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

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

1. $(\mathsf{msk},\mathsf{mpk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
2. $(m_0, m_1, \mathsf{pk}_{j,0}, \mathsf{pk}_{j,1}, \alpha) \leftarrow\!\!\$\ \mathsf{A}_1^{\mathsf{KGen}(\mathsf{msk},\cdot),}(1^\lambda,\mathsf{mpk})$.
3. $c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk},\mathsf{pk}_{j,b},m_b)$ *where* $b \leftarrow\!\!\$\ \{0,1\}$.
4. $b' \leftarrow\!\!\$\ \mathsf{A}_2^{\mathsf{KGen}(\mathsf{msk},\cdot)}(1^\lambda, c, \alpha)$.

5. *If $b = b'$ and $\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1} \in \mathcal{Q}_{\mathsf{KGen}}$ then output 1, and otherwise output 0.*

In the game above, the oracle KGen *generates pairs* $(\mathsf{pk}, \mathsf{sk})$*, but only outputs the public key* pk.

Claim 8. *Let* A *be an adversary that breaks privacy of Construction 4 with advantage ε, while asking at most q_R queries to the oracle* RKGen. *Then, there is an algorithm* A′ *with advantage $4\varepsilon/e^2(q_R + 2)^2$ against IND-CPA$^+$ security of* BasicPub$^+$.

Proof. This proof is similar to the proof of [12, Lemma 4.2]. The challenger starts the game by running the Setup algorithm of BasicPub$^+$ and sends the public parameters $(q, \mathbb{G}, \mathbb{G}_T, e, n, P, P_0, \hat{H})$ to A′. Note that the master secret key $\mathsf{msk} = r$ remains unknown to A′. Now, A′ interacts with the adversary A in the following way:

Setup: A′ samples a secret value $s \leftarrow\$ \mathbb{Z}_q$ and gives A the public parameters defined above, plus two random oracles H and H' under its control and the padding function Φ.

H **queries:** A′ performs the following steps:

1. If query ρ_i is in a tuple $(\rho_i, Q_i, \beta_i, d_i) \in \mathcal{L}_1$, then return Q_i. Otherwise, generate a random coin $d_i \in \{0, 1\}$ so that $Pr[d_i = 0] = \delta$
2. If $d_i = 0$, then sample a random $\beta_i \in \mathbb{Z}_q$, compute $Q_i = P^{\beta_i}$, and add the tuple $(\rho_i, Q_i, \beta_i, 0)$ to $\mathcal{L}_1$. Otherwise, run the public key generation oracle of BasicPub to obtain pk_i, set $Q_i = \mathsf{pk}_i$, and add the tuple $(\rho_i, Q_i, \perp, 1)$ to $\mathcal{L}_1$.
3. Return Q_i.

H' **queries:** A′ maintains a list $\mathcal{L}_2$ that stores tuples of the form (σ_i, Z_i) with the history of calls to H'. If the query σ_i was already done, the challenger returns the value Z_i. If not, it samples a random $Z_i \in \mathbb{G}$, adds (σ_i, Z_i) to the list, and returns Z_i.

SKGen **queries:** Let σ_i be the input to oracle SKGen. A′ obtains $H'(\sigma_i) = Z_i$, where (σ_i, Z_i) is the corresponding tuple in $\mathcal{L}_2$, and returns Z_i^s.

RKGen **queries:** Let ρ_i be the input to oracle RKGen. A′ obtains $H(\rho_i) = Q_i$, where $(\rho_i, Q_i, \beta_i, d_i)$ is the corresponding tuple in $\mathcal{L}_1$. If $d_i = 1$, A′ aborts; otherwise, returns $\mathsf{dk}_{\rho_i} = (P_0^{\beta_i}, Q_i^s, Q_i = P^{\beta_i})$. Note that, since $P_0 = P^r$, we have that $\mathsf{dk}^1_{\rho_i} = (P^{\beta_i})^r = Q_i^r$.

Challenge: At this moment, A sends $(m_0, m_1, \mathsf{rcv}_0, \mathsf{rcv}_1, \sigma_0, \sigma_1)$ to A′. Now A′ performs the following steps:

1. Let $\mathsf{rcv}_0 = \rho_0$ and $\mathsf{rcv}_1 = \rho_1$. A′ queries $H(\rho_0) = Q_0$ and $H(\rho_1) = Q_1$. If both tuples $(\rho_0, Q_0, b_0, 1)$ and $(\rho_1, Q_1, b_1, 1)$ do not belong to $\mathcal{L}_1$ (i.e., $d_i = 1$ in both tuples), A′ aborts. Otherwise, we know that $d_0 = 1$ and $d_1 = 1$, which means that $Q_0 = \mathsf{pk}_0$ and $Q_1 = \mathsf{pk}_1$.
2. A′ computes $T = P^t$, for a random $t \in \mathbb{Z}_q$ and queries $H'(\sigma_0) = Z_0$ and $H'(\sigma_1) = Z_1$. It uses them to obtain $m_0^* = \Phi(m_0) \oplus \hat{H}(e(Q_0, T \cdot Z_0^s))$ and $m_1^* = \Phi(m_1) \oplus \hat{H}(e(Q_1, T \cdot Z_1^s))$. Note that $\mathsf{ek}_{\sigma_i} = Z_i^s$.
3. A′ sends (m_0^*, m_1^*, Q_0, Q_1) to its challenger and receives $C = (U, V)$ as response.
4. A′ computes $C' = (T, U, V)$ and sends it to A. Note that this is a proper encryption of m_b under policy $\mathsf{rcv}_b = \rho_b$ and sender's identity σ_b.

Second query phase: A′ answers all the queries as in the first phase.

Guess: A outputs a guess b' and A′ responds its challenger with the same guess.

Assuming that the adversary makes at most q_R queries to oracle RKGen, then the probability that A' does not abort for any of these calls is δ^{q_R}. Similarly, A' does not abort in the challenge with probability $(1-\delta)^2$. Hence, the overall probability of A' not aborting is $\delta^{q_R}(1-\delta)^2$, which is maximized at $\delta_{opt} = q_R/(q_R+2)$. If we use δ_{opt} as the probability for obtaining coins $d_i = 0$ in H queries, we have that the probability of A' not aborting is at least $4/e^2(q_R+2)^2$. □

Claim 9. *Let* A *be an adversary that breaks IND-CPA*$^+$*-security of* $\mathsf{BasicPub}^+$ *with advantage* ε*, and asks at most* $q_{\hat{H}}$ *queries to the random oracle* $\hat{H}$*. Then, there is an algorithm* A' *that solves the BDH problem with advantage* $2\varepsilon/q_{\hat{H}}$*.*

Proof. The proof follows the strategy of [12, Lemma 4.2]: A' receives a BDH tuple (P, P^a, P^b, P^c), whose correct solution is $D = e(P,P)^{abc}$. During setup, the A' sends the master public key to A where $P_0 = P^a$. This implies that the master secret key $\mathsf{msk} = a$, although this remains unknown to A'. Then, A' proceeds in the following way:

KGen **queries:** A' samples a random $x_i \in \mathbb{Z}_q$ and sets $\mathsf{pk}_i = (P^b)^{x_i}$. Note that the associated secret key is $\mathsf{sk}_i = P^{abx_i}$, although it remains unknown to A'.

$\hat{H}$ **oracle:** A' maintains a list $\hat{\mathcal{L}}$ that stores tuples of the form $(X_i, \hat{h}_i)$ with the history of calls to $\hat{H}$. If the query X_i was already done, the A' returns the value $\hat{h}_i$. If not, it samples a random $h_i \in \{0,1\}^n$, adds $(X_i, \hat{h}_i)$ to the list, and returns $\hat{h}_i$.

Challenge: A sends a tuple $(m_0, m_1, \mathsf{pk}_{j_0}, \mathsf{pk}_{j_1})$. A' samples a random string $Z \in \{0,1\}^n$, defines the challenge ciphertext as $c = (P^c, Z)$, and sends c to A. Note that the decryption of c is $Z \oplus \hat{H}(e(P^c, \mathsf{sk}_{j_\beta}))$, for some $\beta \in \{0,1\}$, which is equal to $Z \oplus \hat{H}(D^{x_{j_\beta}})$, where x_{j_β} is the secret key associated to public key pk_{j_β}.

Guess: The A' receives the guess β' from the A, sets $z = 1/x_{j_{\beta'}}$, takes a random tuple $(X_i, \hat{H}_i) \in \hat{\mathcal{L}}$ and outputs X_i^z as the solution to the received instance of BDH.

A' outputs the correct solution D with probability at least $2\varepsilon/q_{\hat{H}}$. The analysis that gives this bound is the same than the one provided in [12, Lemma 4.2]. Thus we will omit it here. □

of Lemma 7. By composing the reductions in Claim 8 and Claim 9, we can conclude that if there exists an adversary that breaks privacy with advantage ε, then there exists an algorithm that solves the BDH problem with advantage $8\varepsilon/e^2(q_R+2)^2 q_{\hat{H}}$. □

Lemma 8. *Let* A *be an adversary that breaks authenticity of Construction 4 with advantage* ε*, while asking at most* q_R*,* q_S*,* $q_{\hat{H}}$ *queries, respectively, to oracles* RKGen*,* SKGen*, and to the random oracle* $\hat{H}$*. Then, there is an algorithm* A' *that solves the BDH problem with advantage* $8\varepsilon/e^2(q_R+q_S+2)^2 q_{\hat{H}}$*.*

Proof. A' receives the challenge (P, P^a, P^b, P^c). The solution is $D = e(P,P)^{abc}$. A' decides that the master secret key is $\mathsf{msk} = (a, b, H')$ (although a, b are unknown). Now, A' interacts with the adversary A in the following way:

Setup: A' gives A the public parameters $(q, \mathbb{G}, \mathbb{G}_T, e, n, P, P^a = P_0, H, H', \hat{H}, \Phi)$ where H, H', $\hat{H}$ are three random oracles controlled by A'.

H **queries:** A' performs the following steps:

1. If query ρ_i is in a tuple $(\rho_i, Q_i, \beta_i, d_i) \in \mathcal{L}_1$, then return Q_i. Otherwise, generate a random $\beta_i \in \mathbb{Z}_q$, and random coin $d_i \in \{0,1\}$ so that $Pr[d_i = 0] = \delta$.

2. If $d_i = 0$, then set $Q_i = P^{\beta_i}$. Otherwise, set $Q_i = P^{c\beta_i}$.
3. Finally, add $(\rho_i, Q_i, \beta_i, d_i)$ to $\mathcal{L}_1$ and send Q_i to A.

H' queries: A' performs the following steps:

1. If query σ_i is in a tuple $(\sigma_i, Q_i, \beta_i, d_i) \in \mathcal{L}_2$, then return Q_i. Otherwise, generate a random $\beta_i \in \mathbb{Z}_q$, and random coin $d_i \in \{0, 1\}$ so that $Pr[d_i = 0] = \delta$.
2. If $d_i = 0$, then set $Q_i = P^{\beta_i}$. Otherwise, set $Q_i = P^{a\beta_i}$.
3. Finally, add $(\sigma_i, Q_i, \beta_i, d_i)$ to $\mathcal{L}_2$ and send Q_i to A.

$\hat{H}$ queries: A' maintains a list $\hat{\mathcal{L}}$ that stores tuples of the form $(X_i, \hat{h}_i)$ with the history of calls to $\hat{H}$. If the query X_i was already done, the challenger returns the value $\hat{h}_i$. If not, it samples a random $h_i \in \{0, 1\}^n$, adds $(X_i, \hat{h}_i)$ to the list, and returns $\hat{h}_i$.

SKGen **queries:** Let σ_i be the input to oracle SKGen. A' obtains $H'(\sigma_i) = Q_i$, where $(\sigma_i, Q_i, \beta_i, d_i)$ is the corresponding tuple in $\mathcal{L}_2$. If $d_i = 1$, A' aborts; otherwise, returns $\mathsf{ek}_{\sigma_i} = P^{b\beta_i}$.

RKGen **queries:** Let ρ_i be the input to oracle RKGen. A' obtains $H(\rho_i) = Q_i$, where $(\rho_i, Q_i, \beta_i, d_i)$ is the corresponding tuple in $\mathcal{L}_1$. If $d_i = 1$, A' aborts; otherwise, returns $\mathsf{dk}_{\rho_i} = (P^{a\beta_i}, P^{b\beta_i}, Q_i = P^{\beta_i})$.

Forgery: At this moment, A sends (c, ρ, snd) to A'. Let $\mathsf{snd} = \sigma$. Now A' performs the following steps:

1. Compute $H(\rho) = Q$ and $H'(\sigma) = Q'$. If both the tuples $(\rho, Q, \beta, d) \in \mathcal{L}_1$ and $(\sigma, Q', \beta', d') \in \mathcal{L}_2$ do not have coins d, d' equal to 1, A' aborts. If not, we know that $\mathsf{dk}^2_\rho = P^{cb\beta}$ and $H'(\sigma) = P^{a\beta'}$. Hence, $\hat{H}(k_S) = \hat{H}(e(\mathsf{dk}^2_\rho, H'(\sigma))e(\mathsf{dk}^3_\rho, T))$, where: $e(\mathsf{dk}^2_\rho, H'(\sigma)) = e(P^{cb\beta}, P^{a\beta'}) = D^{\beta\beta'}$, and $Q = \mathsf{dk}^3_\rho$.
2. Parse c as (T, U, V). Compute $z = 1/(\beta\beta')$ and take a random tuple $(X_i, \hat{h}_i)$. Return $D' = (X_i \cdot e(Q, T)^{-1})^z$.

First of all, note that the simulation is perfect since in the authenticity game we require that the challenge $(c, \rho, \mathsf{snd} = \sigma)$ satisfies $\rho \notin \mathcal{Q}_{\mathsf{RKGen}}$ and $\forall \sigma' \in \mathcal{Q}_{\mathsf{SKGen}}, \sigma' \neq \sigma$. Assuming that the adversary makes at most q_R and q_S queries to oracle RKGen and SKGen, then the probability that A' does not abort for any of these calls is $\delta^{q_R+q_S}$. Similarly, A' does not abort in the forgery phase with probability $(1-\delta)^2$. Hence, the overall probability of A' not aborting is $\delta^{q_R+q_S}(1-\delta)^2$, which is maximized at $\delta_{opt} = (q_R + q_S)/(q_R + q_S + 2)$. If we use δ_{opt} as the probability for obtaining coins $d_i = 0$ in H and H' queries, we have that the probability of A' not aborting is at least $4/e^2(q_R + q_S + 2)^2$.

If A' does not abort, it outputs the correct solution D' with probability at least $2\varepsilon/q_{\hat{H}}$. Hence, A' solves the BDH problem with advantage $8\varepsilon/e^2(q_R + q_S + 2)^2 q_{\hat{H}}$. □

By setting $\varepsilon \geq \frac{1}{\mathsf{poly}(\lambda)}$, $q_R = \mathsf{poly}(\lambda)$, $q_S = \mathsf{poly}(\lambda)$, $q_{\hat{H}} = \mathsf{poly}(\lambda)$ in Lemmas 7–8, we obtain that Construction 4 is secure.