What Did Come Out of It? Analysis and Improvements of DIDComm Messaging

Christian Badertscher[1], Fabio Banfi[2], and Jesus Diaz[3]

[1]IOG & Zurich University of Applied Sciences, Switzerland
[2]Zühlke Engineering AG, Switzerland
[3]IOG, Spain

August 29, 2024

Abstract

Self-Sovereign Identity (SSI) empowers individuals and organizations with full control over their data. Decentralized identifiers (DIDs) are at its center, where a DID contains a collection of public keys associated with an entity, and further information to enable entities to engage via secure and private messaging across different platforms. A crucial stepping stone is DIDComm, a cryptographic communication layer that is in production with version 2. Due to its widespread and active deployment, a formal study of DIDComm is highly overdue.

We present the first formal analysis of DIDComm's cryptography, and formalize its goal of (sender-) anonymity and authenticity. We follow a composable approach to capture its security over a generic network, formulating the goal of DIDComm as a strong ideal communication resource. We prove that the proposed encryption modes reach the expected level of privacy and authenticity, but leak beyond the leakage induced by an underlying network (captured by a parameterizable resource).

We further use our formalism to propose enhancements and prove their security: first, we present an optimized algorithm that achieves simultaneously anonymity and authenticity, conforming to the DIDComm message format, and which outperforms the current DIDComm proposal in both ciphertext size and computation time by almost a factor of 2. Second, we present a novel DIDComm mode that fulfills the notion of *anonymity preservation*, in that it does never leak more than the leakage induced by the network it is executed over. We finally show how to merge this new mode into our improved algorithm, obtaining an efficient all-in-one mode for full anonymity and authenticity.

Contents

1 Introduction

1.1 Background and Motivation

The Self-Sovereign Identity (SSI) ecosystem is mostly being designed around Decentralized Identifiers (DIDs), which are now a recommendation within the W3C [W3C22]. Via DIDs, entities publish information for running cryptographic protocols with them, like signing or key exchanges. However, the DID stack —and in some sense its philosophy—deviates from previous PKI-based approaches, and new frameworks are being proposed that implement cryptographic protocols leveraging the cryptographic material inside a DID. While it could be argued that existing and well studied standards could be used for this, DID-based frameworks keep gaining traction. DIDComm [Fou23] is one of such frameworks, aimed at establishing secure communication across parties equipped with DIDs.

1.1.1 DIDComm.

DIDComm's v1 was shaped as a set of RFCs for connectionless and secure communications within Hyperledger Aries [Hyp19]. Via Hyperledger Aries, DIDComm v1 was adopted by the British Columbia government [Gov23b, Gov23a], and by credit unions in the US [Tec23], among others. Currently, DIDComm v2 is being incubated by the Decentralized Identity Foundation [Ide23]. While it is backwards incompatible with DIDComm v1, several projects are building on top of it, like Veramo [Ver23] (used for example by the Metamask plugin), or the Atala PRIM SSI solution [Inp23].

The main usage of DIDComm is in setting up a secure communication route between a sender and one or more recipients. We depict the typical operation in Figure 1. Recipients first register their public key information in shape of DIDs (step 0), usually containing information about how to reach them via (a sequence of) chosen mediators. Senders fetch this information from the DID-based PKI (step 1), and build the DIDComm envelope (step 2), an onion-like encrypted package with one encryption per mediator, so that each decrypted layer reveals the next hop. The payload in each hop is symmetrically encrypted with a random key that is further encrypted with keys derived from non-interactive key exchanges between sender and recipients (or mediators, for intermediate hops). Mediators can not only decrypt and resend messages but also, e.g., choose to add new hops, or change the transport medium. The DIDComm envelope is sent via the chosen network (steps 3-5). Finally, the recipient receives the final layer of encryption, rederives the negotiated keys, and decrypts the plaintext message (step 6).

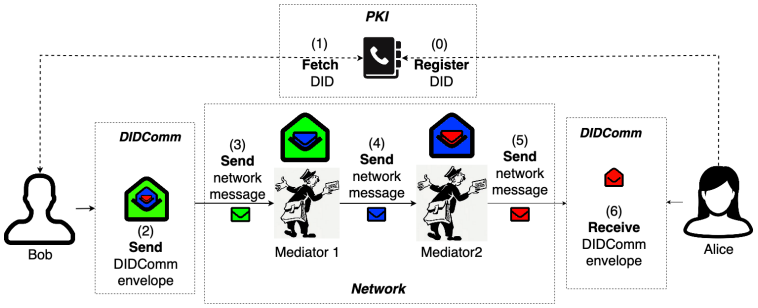

Figure 1: Establishing a secure communication route with DIDComm.

1.1.2 Related Work.

Given DIDComm's increasing adoption and future prospects, it is of highest importance to develop a formal understanding of its security goals, and to propose secure improvements if needed. As we have learned with other communication protocols, like TLS [PvdM16], formal foundations are a key enabler for benchmarking

improvement proposals and to guarantee a sustainable development and well-founded discussions, as evidenced by the impressive list of references in the RFC of TLS 1.3 [Res18].

Built on top of DIDs, DIDComm is best suited for Web3 settings. However, it is natural to ask about its sensitivity to Web2 risks. We overview some of the main attacks suffered in the past by TLS, as a means to illustrate whether or not they apply to DIDComm:

- Attacks to cryptographic algorithms, exploiting weak cryptography, or the incorrect use of cryptographic modes (e.g., POODLE [M̈14]). This risk also applies to DIDComm. As stated next, we identify a subtle, easy to fix, attack of this type.
- Downgrade and renegotiation attacks (e.g., FREAK [BBD+15], Logjam [ABD+15] or DROWN [ASS+16]). TLS is session-oriented, and begins by client and server (re-)negotiating parameters for cryptographic operation. Due to this, a well-known type of attacks to TLS focuses on forcing the client/server to pick a vulnerable set of parameters. Since DIDComm is not session-oriented and, in particular, does not include negotiation of cryptographic parameters, these attacks do not apply.
- Implementation attacks, focusing on implementation flaws that may even depend on complex combinations, like cookie data compression (e.g., CRIME [RD12]). These attacks can also affect DIDComm, although it is out of the scope of this work.

Another prominent example of successful communications protocol with emphasis in security and privacy, is Tor [AFNV19, DMS04], which counts with a vast literature on attacks and countermeasures – see e.g. [ESG22, SFEB21, Syv17, SB16, JBHD12]. DIDComm shares with Tor its onion-like encryption approach, and although strong anonymity is not a primary goal in DIDComm, so is to operate on top of arbitrary communication networks, which may include anonymizing ones. However, there are also differences. First, Tor is a session (circuit) oriented protocol, whereas DIDComm is message-oriented. Second, while both Tor and DIDComm employ multiplexing techniques, they are applied at different levels: Tor allows wrapping multiple circuits into one TLS connection, yet each circuit has its own data encryption key; whereas DIDComm uses a single data encryption key, which then further wraps with keys exchanged with multiple recipients.

1.2 Contribution

Towards the formal understanding of the security of DIDComm messaging, focusing on its v2, we make the following contributions.

Formal model for secure communication with DIDs. We give a faithful abstraction of the general setting where DIDs are used over potentially any network – the model put forth by DIDComm in Figure 1 – where we cast the network as parameterizable so that it is possible to steer the anonymity it already guarantees (as different networks give different guarantees and we want to prove one statement for all of them). We cast the ideal guarantees that DIDComm messaging encryption modes aim to achieve by means of a strong ideal communication resource. We put forth a composable treatment in Section 4. This simplifies capturing privacy and anonymity guarantees as it is a simulation-based security notion, while also admits a natural modular approach to protocol design. We not only abstract the network interface into its own module such that our statement about the cryptographic layer holds for any underlying network, but also can rely on the composition theorem to argue that higher-level applications can use the achieved communication resource (e.g. to implement non-deniable authentication based on signatures of plaintexts) and be assured that composition with the actual protocol retains all security guarantees. This composable approach paves the way for analyzing higher-level protocols in the DIDComm framework. While we focus on DIDComm's encryption modes, the task of secure messaging with DIDs is just one module in larger infrastructures around DIDs as we outline in Section 8.

The obtained communication resource exposes the ideal guarantees of the three core modes of DIDComm, denoted by: `anon`, for sending anonymous, private, but unauthenticated messages to a set of recipients; `auth`, for sending authenticated and private messages to a set of recipients; and `a-auth`, for sending anonymous,

private, but authenticated messages to a set of recipients, which appears to be a central feature, hiding the sender toward the network while authenticating the message to the recipients.

Security proof of DIDComm and extensions We introduce DIDComm's main encryption modes via pseudocode in Section 3, to facilitate subsequent understanding of the claimed security properties. This pseudocode has been extracted via careful analysis of the DIDComm v2 spec [CLT22], the standards upon which it is built [Jon15, Mad21], and the official code repository [sd21, Lab23]. Thus, we consider our description to accurately capture DIDComm's operation. This is further endorsed by our prototypical code – upon which we report in Section 7 – which is built directly from DIDComm's repositories, or from the cryptographic libraries upon which DIDComm builds.

We formally cast DIDComm in Section 5 which allows us to reason about its provable security. We show that the ideal communication resource is realized, and expose two subtleties. First, the combined `a-auth` mode does not guarantee that ciphertexts are committed to a message, although authentication alone has it. This might be easily overlooked in practice and become a possible source of confusion in connection with message franking [GLR17]. We suggest to always use committing authenticated encryption to avoid such confusion, and in this paper we always assume that this is the case. Second, since our theorems are very specific about the leakage induced by the cryptographic layer, we observe that the current DIDComm modes leak substantial information. Notably, the set of recipients. While not a primary design goal, it is desirable to avoid this leakage, as otherwise using DIDComm with an anonymous network (e.g. a mixnet) is a clear mismatch as the cryptographic scheme reveals information that the network wanted to hide.

In Section 6 we propose three extensions that we prove secure.

1. We show an optimized `a-auth` mode that combines authenticity and anonymity. Our proposal is much more efficient—about a 2x improvement—for runtime and ciphertext size compared to DIDComm's original proposal. To avoid confusion, when needed we refer to the original combined mode as `naive-a-auth`, and to our proposal as `merge-a-auth`. As we report below, the overhead incurred by `merge-a-auth` is low, when compared to the basic `auth` mode.

2. We introduce the notion of sender-anonymous messages with receiver-anonymity – which we refer to as `ra-anon`– a desirable property for complementing the cryptographic layer by a mode that does not disclose the set of recipients to any nodes along the network path (in contrast to the current implementation which does not intend to hide it). Such a protocol can achieve full anonymity when used alongside private bulletin boards or Tor [AFNV19, DMS04]. In fact, we prove something stronger: this mode is anonymity-preserving in that it does not add any more leakage than what is leaked by any underlying network. Since ciphertexts do not leave a trace on the intended recipients, such a mode is useful to achieve plausible deniability and reducing the attack vector against censorship. We give an algorithm for this mode by borrowing techniques from fuzzy message detection.

3. We show how to incorporate receiver-anonymity into our optimized combined mode of operation, mode to which we refer as `ra-a-auth` to again get an optimized all-in-one algorithm. This algorithm keeps all sender and receiver identities hidden, still the sender can authenticate itself toward all recipients. It is worth noting that even this all-in-one mode is still more efficient than the `naive-a-auth` mode currently available in DIDComm, which shows that we can get more security with practical performance.

We emphasize that our proposed modes introducing receiver-anonymity can be highly desirable in real world use cases. For instance, consider a user who buys online a ticket to some event. In order for the user to receive the ticket, the vendor has to send it. But the user does not want their email provider (or other intermediaries) to learn that they have transacted with the event vendor. With our receiver anonymity mode, the user can specify a generic inbox shared by thousands of users, and periodically (and privately) check for messages aimed at them. This type of interaction (i.e., delivering credentials) is actually a main use case of DIDComm.

Benchmarks We evaluate in Section 7 the computational, space, and memory costs of our proposal by prototyping our extensions, using as reference the existing modes. `merge-a-auth` almost improves computational and space costs by a factor of 2, compared with `naive-a-auth`. Space- and computationally-wise, our new modes (`ra-anon` and `ra-a-auth`) do not induce too much overhead – and, concretely, are more efficient than `naive-a-auth`. Memory-wise, our new modes add about 1.5% extra cost, which is negligible in practice – and can be due to unoptimized coding.

Interest beyond DIDComm We observe that the `auth` and `anon` modes in DIDComm are direct applications of existing RFCs in the domain of JSON Web Algorithms [Jon15, Mad21] (in `auth`, senders use a well-known static key pair; in `anon`, they don't). `naive-a-auth` simply applies them sequentially, but there is no specific mode in [Jon15, Mad21] explicitly covering this combination. Thus, our `merge-a-auth` mode is of independent interest, as it achieves "the best of both worlds" almost for free, that is, almost as fast as authentication alone with ciphertexts of similar sizes. On the other hand, receiver anonymity introduces a privacy enhancement not covered by the existing standards, although it can be made RFC-compliant by leveraging headers defined in the RFCs. Given that –as we report– the overhead of adding this property is low, we consider that this may also be of interest in privacy-demanding scenarios as described above.

2 Preliminaries

2.1 Notation

For a scheme S defining some algorithm $\mathcal{A}$, we sometimes use the notation $\mathsf{S}.\mathcal{A}$, and if $\mathcal{X}$ is a set representing either the domain or the range of some algorithm defined by S, we sometimes use the notation $\mathsf{S}.\mathcal{X}$. In our pseudocode, some functions receive ID arguments – which identify public keys – and then internally use pk_{ID} straight away. For readability, we introduce ID as a shorthand for $(\mathsf{ID}, pk_{\mathsf{ID}})$. Note that, in a PKI setting (like DIDs), knowing the identifier of a public key allows fetching that key from the PKI.

2.2 Key-Derivation Function (KDF)

We define KDF syntax and security following [Kra10]. Intuitively, a KDF is a function that takes as input a sample σ from a *source of keying material* (e.g., the output g^{xy} of a Diffie-Hellman key exchange) and a length value L, and outputs a string of length L. The source of keying material Σ, or simply *source*, is a two-valued probability distribution that outputs (σ, α), where α represents some auxiliary knowledge about σ (or its distribution) that is available to the attacker (e.g., the public values g^x and g^y of a Diffie-Hellman key exchange). We next provide a formal definition of a KDF, which also defines additional inputs, required for example in key-exchange with identities. The security properties are provided in Appendix E.

Definition 1 *A* key-derivation function (KDF) *is an efficiently computable algorithm* $\mathsf{KDF} : \{0,1\}^+ \times \mathbb{N} \times \{0,1\}^* \to \{0,1\}^+$ *that on input a non-empty bitstring* σ *sampled from a source, a length value* L*, and an optional context value* c*, outputs a non-empty bitstring of length* L*, that is,* $\mathsf{KDF}(\sigma, L, c) \in \{0,1\}^L$.

2.3 AE with Associated Data (AEAD)

We follow [Rog02] in defining authenticated encryption with associated data (AEAD) which is compactly committing [GLR17]. Note that we assume AEAD schemes where the ciphertext can be split into a core ciphertext C and an authentication tag T. We state here the syntax and defer the security game to Appendix E.

Definition 2 *For key space* $\mathcal{K}$*, nonce space* $\mathcal{N}$*, header space* $\mathcal{H}$*, message space* $\mathcal{M}$*, ciphertext space* $\mathcal{C}$*, and authentication tag space* $\mathcal{T}$*, an* authenticated encryption with associated data (AEAD) scheme *is a tuple* $\mathsf{AEAD} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ *where:*

- $\mathcal{G}$ *is the uniform distribution over* $\mathcal{K}$*;*

- $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$ *is the deterministic* encryption algorithm;
- $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \cup \{\perp\}$ *is the deterministic* decryption algorithm.

For correctness, it is required that for any key $K \in \mathcal{K}$, any nonce $N \in \mathcal{N}$, any header $H \in \mathcal{H}$, and any message $M \in \mathcal{M}$, letting $(C, T) \leftarrow \mathcal{E}(K, N, H, M)$, results in $\mathcal{D}(K, N, H, C, T) = M$. Moreover, we assume that $|C| = |M|$, and $|T| = \tau$, where τ is a constant defined by the scheme AEAD.[1]

2.4 Deterministic AE (DAE)

Since DIDComm uses key-wrapping, we need a tailored version of authenticated encryption, namely *deterministic authenticated encryption* (DAE) [RS06], which is suitable for encrypting (authentically) keying material. Even though [RS06] defines DAE with associated data, since DIDComm does not take advantage of this feature in the key-wrapping phase, we do not consider this in our formalism. We state here the syntax and defer the security game to Appendix E.

Definition 3 *For key space $\mathcal{K}$, message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, a* deterministic authenticated encryption (DAE) scheme *is a tuple* $\mathsf{DAE} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ *where:*

- $\mathcal{G}$ *is the uniform distribution over $\mathcal{K}$;*
- $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ *is the deterministic* encryption algorithm;
- $\mathcal{D} : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *is the deterministic* decryption algorithm.

For correctness, it is required that for any key $K \in \mathcal{K}$ and any message $M \in \mathcal{M}$, $\mathcal{D}(K, \mathcal{E}(K, M)) = M$.

3 Introduction to DIDComm's Encryption Modes

DIDComm specifies two encryption modes [CLT22, Section 5.1]: *anonymous sender encryption* (AnonCrypt), which aims at providing confidentiality and integrity without revealing the identity of the sender, and *authenticated sender encryption* (AuthCrypt), which aims at additionally proving the identity of the sender, but *exclusively* to the recipient. DIDComm also specifies sequentially composing AuthCrypt and AnonCrypt in order to combine their properties.

We describe the two modes on a high level. Both AnonCrypt and AuthCrypt employ elliptic curve Diffie-Hellman (ECDH) key agreement to protect a randomly generated content encryption key (CEK) using key-wrapping [RS06]. In order to protect the CEK, a key encryption key (KEK) is obtained by applying a key-derivation function (KDF). The CEK is then used for an authenticated encryption scheme with associated data (AEAD) such as AES in Galois/Counter Mode (AES256-GCM) or CBC mode (AES256-CBC) with HMAC with SHA512 (HMAC-SHA512) for authentication.

The reason for generating a fresh random key for encryption, rather than using the one derived from ECDH, is to allow for more compact ciphertexts in the case of multiple receivers: rather than encrypting the whole message with a distinct key generated for each receiver, encrypting the message with the same key for each receiver and then encrypting (the shorter) key separately for each receiver, significantly reduces the ciphertext size. DIDComm enforces this approach even in the case of a single receiver so as to support both single and multi receiver flows, and has been motivated as a trade-off to reduce code paths [HC22].

The main difference between AuthCrypt and AnonCrypt is how the ECDH key agreement is defined: AnonCrypt uses an ephemeral key-pair for the sender, and the receiver's static key-pair to derive a shared secret, while AuthCrypt derives an additional shared secret using both sender and receiver's static key-pairs. In addition, in AnonCrypt, only the identifiers of the recipients' keys are fed to the KDF; whereas AuthCrypt also feeds the KDF with the sender's key identifier. Both AnonCrypt and Authcrypt rely on existing standards. We overview them in Appendix A. We next revisit both methods, detailing their further differences, and appropriately abstract them to analyze DIDComm's claimed security guarantees.

[1] This is the case for the GCM and CBC-HMAC AEAD schemes used by DIDComm.

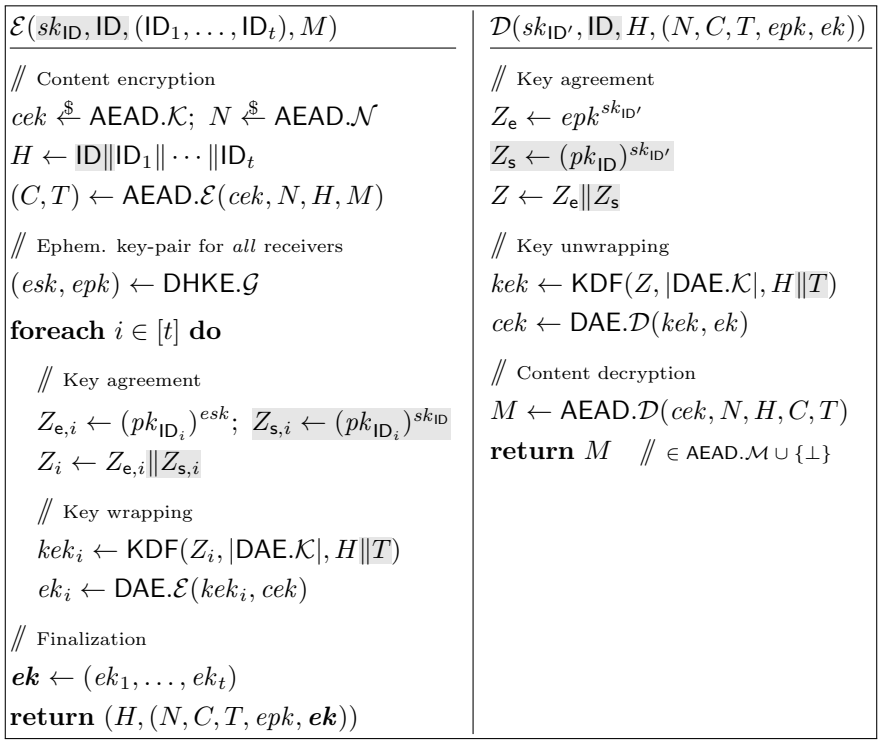

Figure 2: Encryption and decryption algorithms of `anon`, and `auth`, with the additions to build `auth` from `anon` highlighted in gray. ID denotes the sender identity, and $\mathsf{ID}', \mathsf{ID}_1, \ldots, \mathsf{ID}_t$ denote the identities of the receivers. Recall that given ID, the key pk_{ID} is always available. To distinguish the modes, we refer to the algorithms by $\mathcal{E}_{\text{anon}}/\mathcal{D}_{\text{anon}}$ and $\mathcal{E}_{\text{auth}}/\mathcal{D}_{\text{auth}}$, respectively.

3.1 Abstracting AnonCrypt and AuthCrypt

We formally define both the (multi-receiver) scheme `anon` (AnonCrypt) and the (multi-receiver) scheme `auth` (AuthCrypt) as the composition of three primitives: a key-derivation function scheme KDF, an AE scheme AEAD with associated data, and a deterministic AE scheme DAE. Suppose a sender with identity ID wants to send a message M to a list of receivers with identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$. First, both `anon` and `auth` algorithms sample a fresh CEK cek and nonce N from the respective spaces defined by Definition 2. Then, they both define the list of receiver identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$ as the additional data H, and `auth` additionally includes the sender identity ID to H. Encrypting M with $\mathsf{AEAD}.\mathcal{E}$ results in ciphertext-tag pair (C, T), which will be part of the final ciphertext.

The next step is to encrypt the CEK for each receiver. For this, both `anon` and `auth` use the *same* ephemeral public-key for each receiver [Mad21], which is motivated by a line of works such as [Kur02, BBS03, BBKS07]. More precisely, for both schemes, the sender first generates an ephemeral Diffie-Hellman key-pair $(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$, with $epk = g^{esk}$, and then derives for each receiver with identity ID_i an (ephemeral) shared secret $Z_{\mathsf{e},i} = (pk_{\mathsf{ID}_i})^{esk}$ using the ephemeral secret key esk and receiver ID_i's public key pk_{ID_i}. Then, only if using `auth`, the sender derives an additional (static) shared secret $Z_{\mathsf{s},i} = (pk_{\mathsf{ID}_i})^{sk_{\mathsf{ID}}}$ using its own secret key sk_{ID} and the receiver ID_i's public key pk_{ID_i}. If using `anon`, the sender sets Z_i to $Z_{\mathsf{e},i}$, while if using `auth`, it sets Z_i to $Z_{\mathsf{e},i} \| Z_{\mathsf{s},i}$. Next, according to Definition 1, the sender derives the KEK kek_i by feeding Z_i as the source sample to KDF, specifying length $L = |\mathsf{DAE}.\mathcal{K}|$. In case of `anon`, the context value is set to H, but in case of `auth`, it is set to $H \| T$. Note that setting the context to $H \| T$ is crucial in order to achieve authenticity as, otherwise, the encrypted content (the AEAD ciphertext, C) and the CEK are decoupled, enabling an attack that replaces the message and this would break authenticity and the ideal resource we

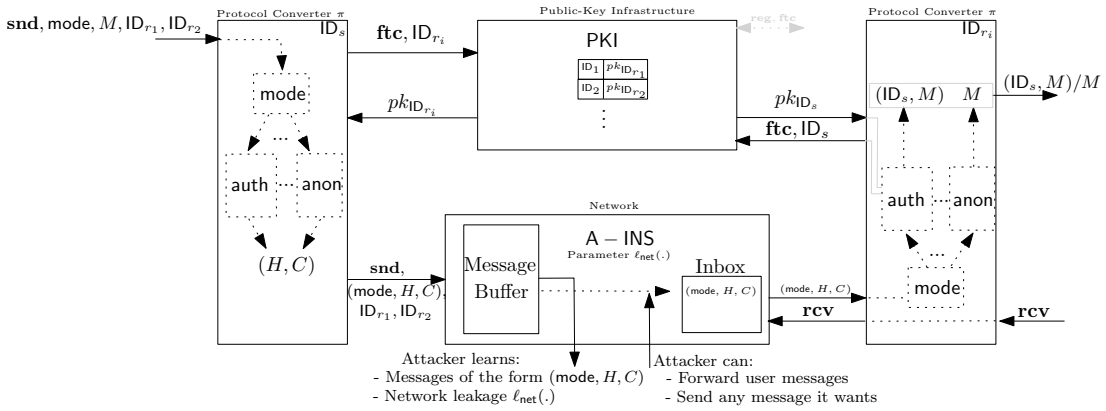

Figure 3: The direct modeling of the DIDComm setting of Figure 1 as resources and converters in Constructive Cryptography. We depict only two honest parties (U) as a left and right interface for sender and receive, respectively (but note that in the general protocol, a party play both roles), and the attacker's interface E is drawn at the bottom of the resources, where the attacker learns all information contained in the message buffer and can steer delivery of messages.

INIT()	$\mathsf{U}_i(\mathbf{reg}; \mathsf{ID}, pk)$ // ID is not registered yet	$\mathsf{U}_i(\mathbf{ftc}; \mathsf{ID})$
$\mathbf{P} \leftarrow []$	$\mathbf{P}[\mathsf{ID}] \leftarrow pk$	**return** $\mathbf{P}[\mathsf{ID}]$
	$\mathsf{E}(\mathbf{reg}; \mathsf{ID}, pk)$ // ID is not registered yet	$\mathsf{E}(\mathbf{ftc}; \mathsf{ID})$
	$\mathbf{P}[\mathsf{ID}] \leftarrow pk$	**return** $\mathbf{P}[\mathsf{ID}]$

Figure 4: Public-key infrastructure resource PKI_n, for $i \in [n]$ for maintaining a unique association $\mathbf{P}$ between identities and keys.

introduce later could not be achieved.

Finally, the sender encrypts *cek* using a DAE scheme DAE according to Definition 3 with kek_i as key, obtaining the encrypted key ek_i. The resulting ciphertext is defined to be $(H, (N, C, T, epk, \boldsymbol{ek}))$, where $\boldsymbol{ek} := (ek_1, \ldots, ek_t)$, and decryption is the natural inversion of this process. We provide a formal description of encryption and decryption of both `auth` and `anon` in Figure 2, where the corresponding functions are denoted by $\mathcal{E}_{\text{anon}}/\mathcal{D}_{\text{anon}}$ and $\mathcal{E}_{\text{auth}}/\mathcal{D}_{\text{auth}}$.

For the combined mode `a-auth`, DIDComm puts forth the straightforward combination $(H, C) \leftarrow \mathcal{E}_{\text{a-auth}}(\mathsf{ID}, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t), M) := \mathcal{E}_{\text{anon}}((\mathsf{ID}_1, \ldots, \mathsf{ID}_t), (H', C'))$ with $(H', C') \leftarrow \mathcal{E}_{\text{auth}}(\mathsf{ID}, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t), M)$. And, tor the decryption counterpart, $(\mathsf{ID}, M) \leftarrow \mathcal{D}_{\text{a-auth}}((\mathsf{ID}_1, \ldots, \mathsf{ID}_t), (H, C)) := \mathcal{D}_{\text{auth}}(\mathsf{ID}, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t), (H', C'))$, where $(H' = (\mathsf{ID} \| \mathsf{ID}_1 \| \ldots \| \mathsf{ID}_t), C') := \mathcal{D}_{\text{anon}}((\mathsf{ID}_1, \ldots, \mathsf{ID}_t), (H, C))$. Note that we augment the output of $\mathcal{D}_{\text{a-auth}}$ with the identity of the sender, ID, even though this is not strictly part of the encrypted plaintext. This slight abuse of notation is needed to explicitly capture that, in `a-auth`, from an a priori sender-unauthenticated ciphertext, the recipient learns the sender's identity.

4 Formalizing DIDComm's Goals

We model DIDComm's security in the *constructive cryptography* (CC) framework [MR11, Mau12] introduced by Maurer and Renner. In Appendix D we provide an introduction to the framework. CC allows to define the security of a cryptographic protocol as a statement capturing the construction of an ideal functionality

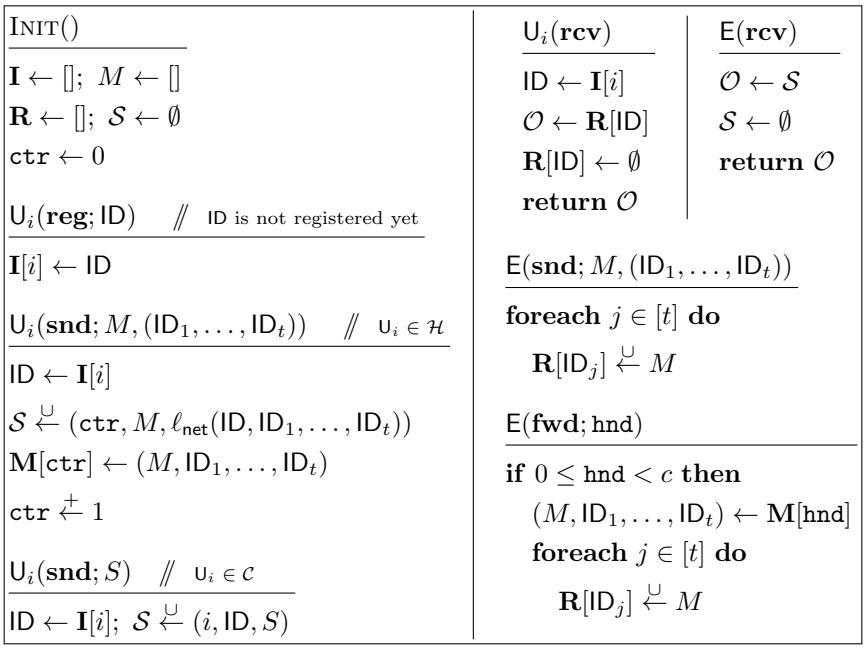

Figure 5: Assumed anonymous insecure network resource $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}}}$, for $i \in [n]$. The honest interface allows to specify a message and a target set of recipients, where the final delivery is under the power of the network adversary (interface E). Furthermore, a dishonest sender can formally insert any message it wants into the sender buffer.

from a set of assumed functionalities, and such a notion of security is naturally *composable*. We model functionalities (ideal and assumed) as *resources*, defined by their input-output behavior, and the protocol interacting with the assumed resources as a list of *converters*. Rather than defining the explicit behavior, resources and converters are described in words or pseudo-code. A resource specifies a set $\mathcal{I}$ of *interfaces*, which should be thought of as being assigned to parties, and attaching a converter π to a resource $\mathbf{R}$ at a particular party interface $\mathsf{U} \in \mathcal{I}$, denoted $\pi^{\mathsf{U}}\mathbf{R}$, results in a new resource.

We use *distinguishers* to define the distance (in terms of behavior) between two resources $\mathbf{R}$ and $\mathbf{S}$, and write $\mathbf{R} \approx_\varepsilon \mathbf{S}$ to denote the fact that no *efficient* (i.e., polynomial-time) distinguisher can distinguish between the two with advantage greater than ε when interacting through all available interfaces. In this work we consider the setting with $n \geq 1$ (potentially corrupted) party interfaces $\mathcal{U} := \{\mathsf{U}_1, \ldots, \mathsf{U}_n\}$ and an adversary interface E. For a list of converters $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_n)$ (also called a *protocol*), a resource with interface set $\mathcal{I} = \{\mathsf{U}_1, \ldots, \mathsf{U}_n, \mathsf{E}\}$, and a set of corrupted parties $\mathcal{C} \subseteq \mathcal{U}$ and corresponding honest party set $\mathcal{H} := \mathcal{U} \setminus \mathcal{C} := \{\mathsf{U}_{i_1}, \ldots, \mathsf{U}_{i_t}\}$, for $t \leq n$ and $i_j \in [n]$, we use the notation $\boldsymbol{\pi}^{\mathcal{H}}\mathbf{R} := \pi_{i_1}^{\mathsf{U}_{i_1}} \cdots \pi_{i_t}^{\mathsf{U}_{i_t}}\mathbf{R}$. We define security of a protocol as follows.

Definition 4 *For two resources $\mathbf{R}$ and $\mathbf{S}$ with interface set $\mathcal{I} = \mathcal{U} \cup \{\mathsf{E}\}$, for user set $\mathcal{U} = \{\mathsf{U}_1, \ldots, \mathsf{U}_n\}$, and a protocol $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_n)$, we say that $\boldsymbol{\pi}$ (securely) constructs $\mathbf{S}$ from $\mathbf{R}$ within ε, denoted $\mathbf{R} \xrightarrow{\boldsymbol{\pi}, \varepsilon} \mathbf{S}$, if for any corrupt set $\mathcal{C} \subseteq \mathcal{U}$ and honest set $\mathcal{H} := \mathcal{U} \setminus \mathcal{C}$, there exists a simulator $\boldsymbol{\sigma}$ such that $\boldsymbol{\pi}^{\mathcal{H}}\mathbf{R} \approx_\varepsilon \boldsymbol{\sigma}^{\mathcal{C} \cup \{\mathsf{E}\}}\mathbf{S}$.*

The advantage of a composable security notion is that it guarantees that the ideal resource can be replaced by the assumed resource with the protocol in any context: if $\boldsymbol{\pi}_1$ realizes $\mathbf{S}$ from $\mathbf{R}$, and $\boldsymbol{\pi}_2$ realizes $\mathbf{T}$ from $\mathbf{S}$, then their composition $\boldsymbol{\pi}_2\boldsymbol{\pi}_1$ realizes $\mathbf{T}$ from $\mathbf{R}$.

4.1 What Should DIDComm Achieve?

We now give our formal model for DIDComm's encryption modes, which, at the same time, is a general model of communication in the DID space, whenever sender and receiver have only minimal cryptographic setup available in the form of a public key and a lookup table (which we frame as a PKI, but it could also be a blockchain) for finding the association. We use Constructive Cryptography to capture the topology of Figure 1 directly into a corresponding formal model, as shown graphically in Figure 3. This modelling captures in particular sender and receiver(s) of DIDComm messages, and will allow to precisely define expected confidentiality, authenticity, and privacy properties via an ideal resource graphically depicted in Figure 6. This being a complex task on its own, we leave out of scope capturing other capabilities of the larger framework, such as routing via intermediate proxies. We also leave out of scope capturing the security achieved via digital signatures, which DIDComm does support. Note however that DIDComm's encryption modes are the core component of the framework, as they are mandatory whenever any type of confidentiality and privacy is needed and that the cryptographic guarantees we prove hold with respect to a very broad family of networks. In fact the real network resource we introduce below is parameterized and can model completely insecure networks (where all participants are fully identifiable), to a fully anonymous network. This means, as an upside, that the guarantees we prove for the cryptographic protocols are very strong.

4.1.1 DIDComm's Setup

We now describe the resources that we use to model the setup assumed by DIDComm: a public-key infrastructure PKI_n and an insecure network $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}}}$ with an explicit leakage function ℓ_{net} that captures how much information about the communication endpoints could be inferred by a network attacker due to the users using this particular network. On a high level, PKI_n allows any user U_i, for $i \in [n]$, as well as the adversary E, to register a public key pk under ID via the command **reg**, which is internally stored in table $\mathbf{P}$. Any user and the adversary can then fetch the public key pk registered under some ID via the command **ftc**. The resource ensures a unique association, that is, an ID can only appear once in the table. We assume implicitly that registering an already registered identity yields an observable error and require protocols to check before registering an association. The formal description of PKI_n is given in Figure 4.

On a high level, $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}}}$ allows any user U_i, for $i \in [n]$, who registered ID via the command **reg** to the resource, to send a message M to a list of users with registered identities $(\mathsf{ID}_1, \ldots, \mathsf{ID}_t)$ via the command **snd**. The message, along with the network leakage $\ell_{\mathsf{net}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$, is not directly sent, but stored in buffer $\mathcal{S}$ for the adversary E to read via the command **rcv**. Each message is also associated with a unique handle (counter) `hnd`, which the adversary can use to forward M to all users with identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$ via the command **fwd**. Rather than being delivered directly to the user with registered identity ID_j, the message is stored in buffer $\mathbf{R}[\mathsf{ID}_j]$, for each $j \in [t]$. The adversary can also directly send a message M to a list $(\mathsf{ID}_1, \ldots, \mathsf{ID}_t)$ via the command **snd**, which means M is also stored in $\mathbf{R}[\mathsf{ID}_j]$, for each $j \in [t]$. Finally, each user U_i, which registered ID, will be able to retrieve all messages in $\mathbf{R}[\mathsf{ID}]$ via the command **rcv**. We distinguish honest users $\mathcal{H}$ and dishonest or corrupted users $\mathcal{C}$ and usually make this explicit in the notation, i.e., we write $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}},\mathcal{C}}$ to denote the resource where the interfaces specified in $\mathcal{C}$ are formally assigned to dishonest users, and only omit the explicit parameter for sake of readability. Those users are modeled to be able to freely communicate to E (concretely, by putting any value S in the send buffer via the command **snd**) which models that all dishonest entities can be seen as a monolithic (and even more powerful) network attacker controlling the network and corrupted users. The formal description is given in Figure 5.

On realizing DIDComm's setup We now provide some justification as to why the ideal setup resources we describe reasonably follow the setting in a typical DIDComm deployment. A more detailed analysis is an interesting future research direction but outside the scope of this paper that focuses on the cryptographic layer.

For the PKI, note that DIDComm builds on top of DIDs as a source of public key material. Indeed, it can be argued that DIDs are a type of PKI. To see this, observe that, as per [W3C22], any compliant DID method needs to define the operation that allows to associate an identifier – the DID string, e.g. `did:example:id` –

to a (set of) public key(s). Conversely, given an identifier, any compliant DID method needs to define how to fetch the associated (set of) public key(s). It is clear that the former corresponds to the **reg** operation in PKI_n, and the latter corresponds to the **ftc** operation of PKI_n – both also available to adversarial users in the DID setting. Finally, the typical use case is that `id` is the hash of a master public key of the DID, making sure that no one else can claim and perform cryptographic operations using the same DID string.

For the network, our generic ideal resource $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}}}$ is also generic enough to capture a wide range of communication networks. Indeed, in any networking system, any party must be able first join the network and get some address or identifier to which others can send messages. This is captured by the **reg** interface. Then, in any network, any party can send a message to a set of recipients, who must be able to receive them. Both capabilities are captured by the **snd** and **rcv** interfaces. The network is subject to a network attacker, who has far reaching capabilities as how to inject or read messages. The network is parameterized by a leakage function ℓ_{net}, which is made available to the adversary alongside any sent message, and can be set to any desired function depending on the real network that we want to capture. For instance, in the case of a strong anonymizing network, such as Tor [DMS04], the leakage function could be modeled as outputting the empty string, whereas when modeling basic TCP/IP networks, one must assume that identifiers of sender and recipient(s) can be inferred from network traffic.

4.1.2 DIDComm's Goals as an Ideal Resource

In Figure 7 we define $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\mathcal{C}}$ – the ideal resource for DIDComm's encryption modes – with leakage function ℓ_{rcv} and corrupt users set $\mathcal{C}$, which functionally integrates the PKI_n resource. This means that the ideal resource can again be used as a PKI (in future constructions), in addition to the ideal communication functionalities it offers. This is an important feature as it makes sure that the PKI is not consumed but preserved by the construction. $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\mathcal{C}}$ allows any honest user $\mathsf{U}_i \notin \mathcal{C}$, for $i \in [n]$, who registered ID via command **reg** to the resource, to send a message M to a list of users with registered identities $(\mathsf{ID}_1, \ldots, \mathsf{ID}_t)$ via command **snd**, additionally specifying a mode $\mathtt{mode} \in \{\mathtt{anon}, \mathtt{auth}, \mathtt{a\text{-}auth}\}$. As for $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}},\mathcal{C}}$, a buffer $\mathcal{S}$ models what is leaked to the adversary. Together with the leakage $\ell_{\mathsf{rcv}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$, for each of the honest receivers the length $|M|$ of the message is leaked, while for every corrupt user the full message M is leaked. Additionally, if $\mathtt{mode} = \mathtt{auth}$, also the ID of the sending user U_i is leaked. The adversary can read buffer $\mathcal{S}$ via command **rcv**. Each message is associated with a unique handle `hnd`, which the adversary can use to forward M to all users with identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$ via command **fwd**. Rather than being delivered directly to the user with registered identity ID_j, the message is stored in buffer $\mathbf{R}[\mathsf{ID}_j]$, for each $j \in [t]$, together with the sender's ID in case $\mathtt{mode} = \mathtt{auth}$ or $\mathtt{mode} = \mathtt{a\text{-}auth}$. The adversary can also specify an ID together with the handle `hnd` when issuing the command **fwd**, in case it knows the associated ID. If ID is not specified ($\mathsf{ID} = \bot$), the resource will find the receiver ID associated with this handle `hnd`, if such ID exists. This feature is essential: consider the case of no leakage at all (as we achieve in our new modes). Then, messages can be sent among honest users in the desired oblivious way, hiding content and communication endpoints.

The adversary can also directly send a message M, optionally associated to a sending ID, to two lists $\mathcal{R}, \mathcal{R}'$ of IDs via command **snd**. This means that M (or (ID, M)) is also stored in $\mathbf{R}[\mathsf{ID}']$, for each $\mathsf{ID}' \in \mathcal{R}$. For each $\mathsf{ID}' \in \mathcal{R}'$, $\bot$ is instead stored in $\mathbf{R}[\mathsf{ID}']$ – i.e., $\mathcal{R}'$ models the set of recipients whose message the adversary decides to tamper with. Also, the adversary can communicate directly to corrupt users via command **inj**. Finally, each user U_i, which registered ID, can retrieve all messages in $\mathbf{R}[\mathsf{ID}]$ via command **rcv**.

Remark 1 *For brevity, we include in Figure 7 code (within dashed boxes) augmenting the DIDComm model described above. The corresponding improvements are defined in Section 6.*

5 Security of DIDComm

We next formalize DIDComm's protocol as a CC converter in Figure 8. The protocol dispatches the message (resp. the ciphertext) to the correct encryption (resp. decryption) algorithm for each mode. With this, we

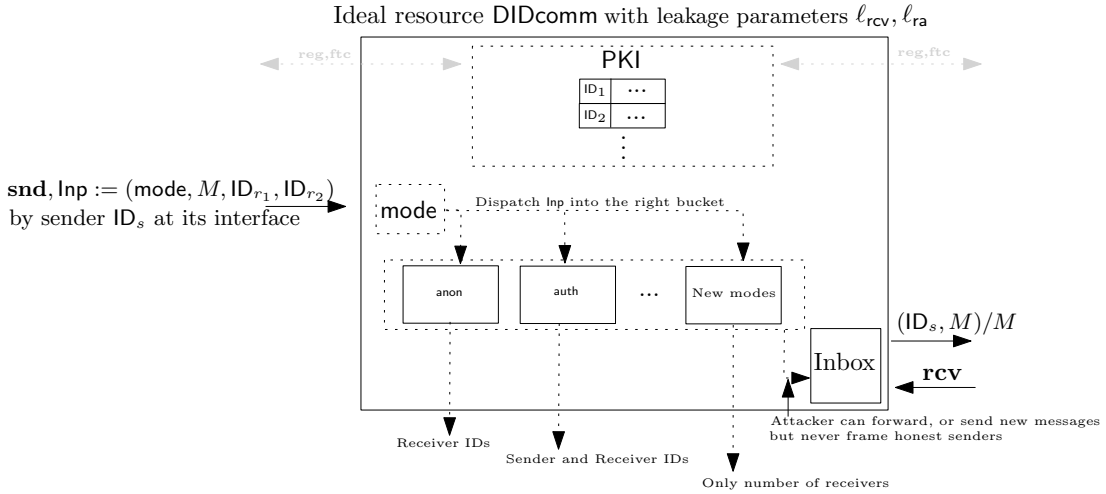

Figure 6: The ideal resource that the cryptographic protocols of DIDComm and our new modes achieve, where we specifically focus on what the cryptographic layer inherently leaks about the involved identities (bottom). Again, depict two honest parties on the left and right, respectively to illustrate the flow of the messages from sender to receiver. The adversary does not learn the contents of the message buffers, only the length of the message (privacy) and, depending on the mode, leakage about the involved identities. The ideal resource further functionally integrates a PKI (top) whose operations are depicted in grey.

are ready to present our first main result, which states that the DIDComm protocol realizes the resource $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\mathcal{C}}$ from PKI_n and $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}},\mathcal{C}}$. Herein, the resulting leakage function

$$\ell_{\mathsf{rcv}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t) := (\ell_{\mathsf{net}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t), (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$$

is the one that leaks the set of recipients in addition to what the underlying network leaks. We generally assume that $t \geq 1$.

Theorem 1 *For any $n \geq 1$ and any assumed leakage function ℓ_{net}, consider the real resources PKI_n from Figure 4 and $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}}}$ from Figure 5, the ideal resource $\mathsf{DIDComm}_{n,\ell_{\mathsf{rcv}},\mathcal{C}}$ depicted in Figure 7 with the leakage function ℓ_{rcv} defined above, and the DIDComm protocol $\boldsymbol{\pi}_{\mathsf{DIDComm}} := (\pi_1, \ldots, \pi_n)$ from Figure 8. We have that, for a negligible function ε, and assuming that the DDH assumption holds,*

$$[\mathsf{PKI}_n, \mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}},\mathcal{C}}] \xrightarrow{\boldsymbol{\pi}_{\mathsf{DIDComm}},\varepsilon} \mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\mathcal{C}}. \tag{1}$$

The proof is given in Appendix E, and builds on the security of the underlying KDF, DAE and AEAD schemes, where, in the former, we assume that the DDH assumption holds, given our explicit adoption of DDH-based sources of keying material for KDFs in Figure 2, following [Kra10]. We point out that due to the similarity of the above statement with Theorem 2 in Section 6.1, where we replace the implementation of mode `a-auth` by a more efficient version, we cover both variants in one security proof in Appendix E.4.

Remark 2 *Recall that we assume throughout the paper that* AEAD *is compactly committing [GLR17]. Otherwise, the slightly weaker resource described in Appendix B, would be realized instead.*

Remark 3 *Throughout this paper, the parameter n denotes the total number of users in the system, and t denotes the number of recipients of a message processed with some of the defined DIDComm modes. In the real world, n is likely to be in the order of tens to hundreds of thousands and, while technically unrestricted,*

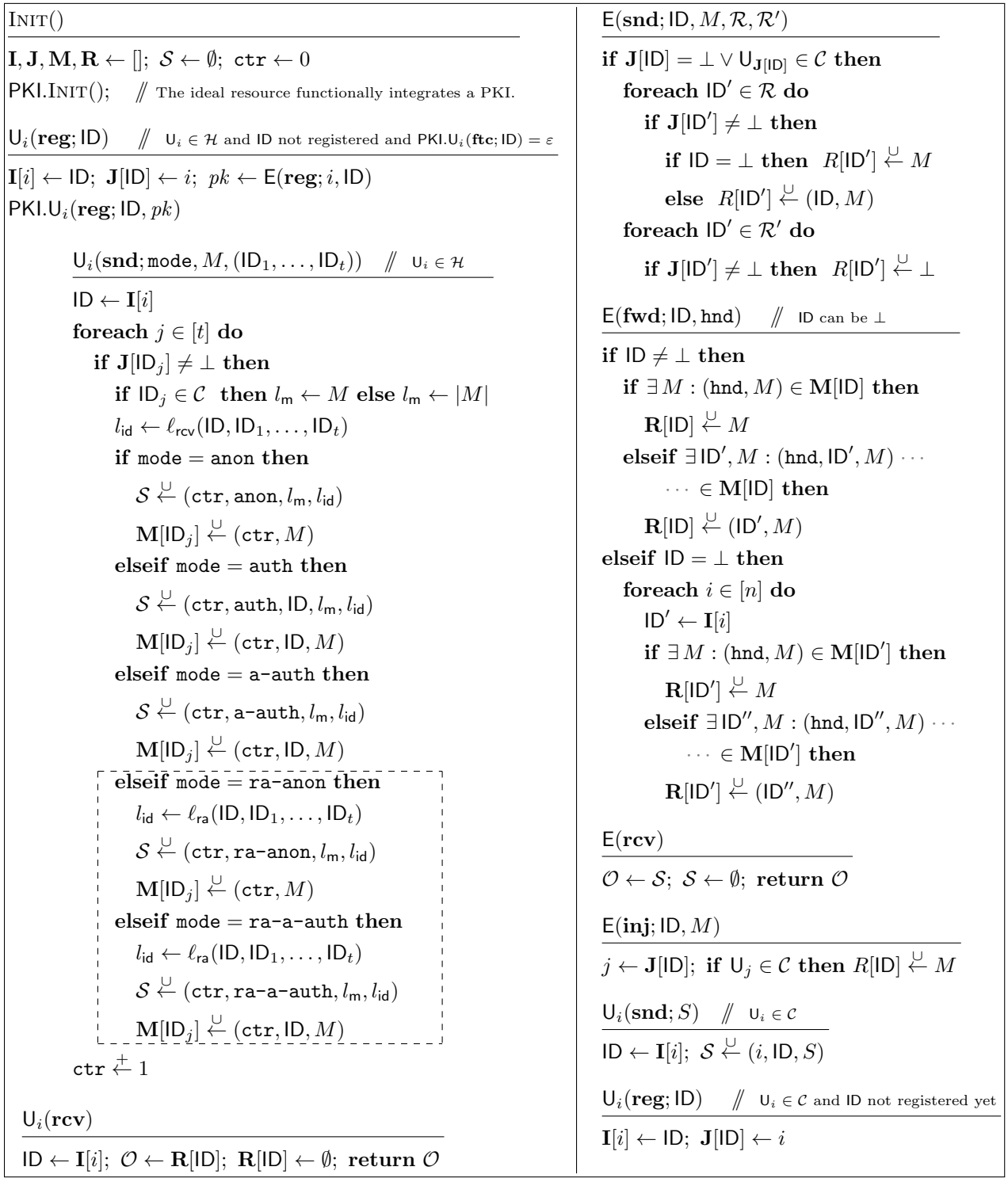

Figure 7: The basic ideal resource $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\mathcal{C}}$ and augmented resource $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\ell_{\mathsf{ra}},\mathcal{C}}$ which includes the two new modes and leakage function in the dashed box. The set of corrupt users (interfaces) is denoted by $\mathcal{C}$. For readability, we only depict the DIDComm related commands and omit explicit commands to the PKI that are also available at the parties' interfaces.

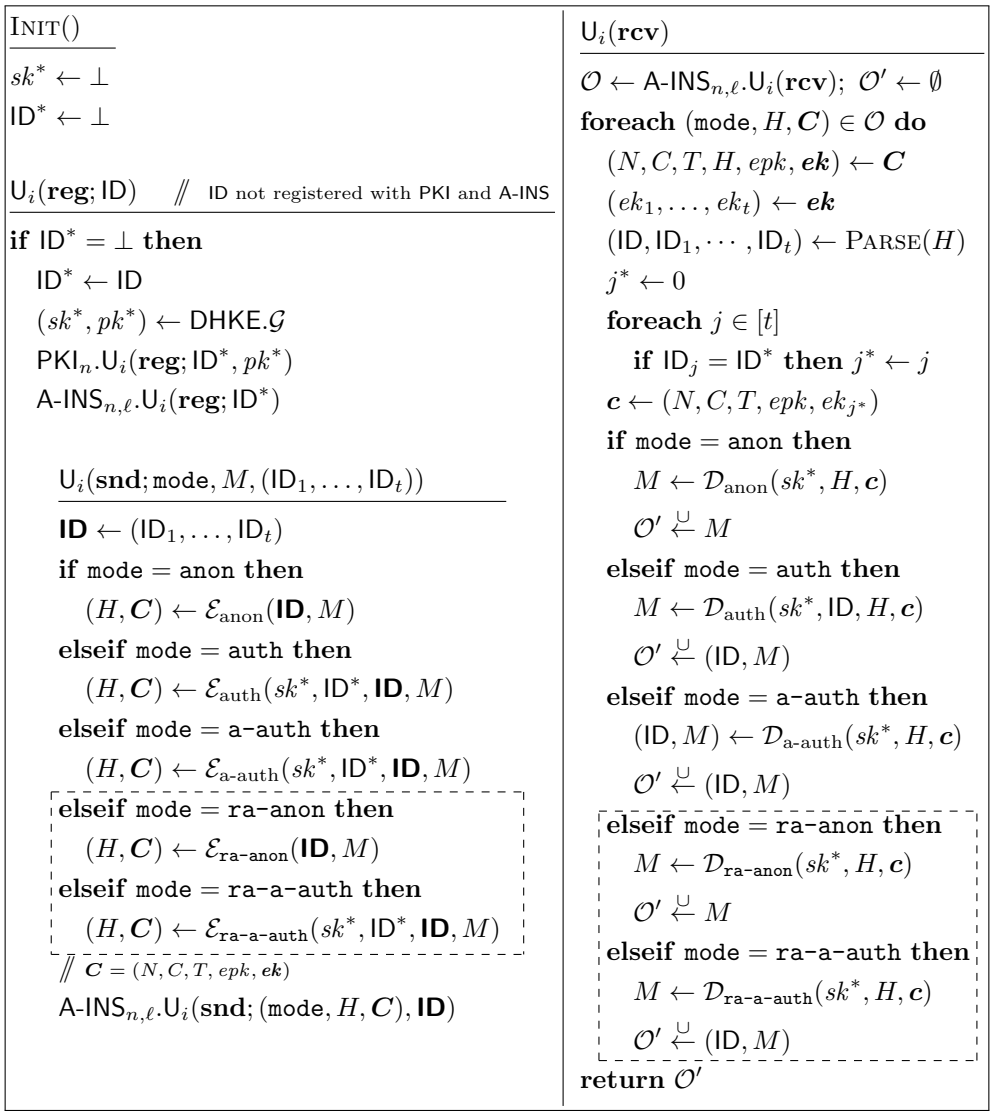

Figure 8: DIDComm's protocol converter π_i for interface U_i, for $i \in [n]$. $\text{PARSE}(H)$ returns $(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$ if $H = \mathsf{ID} \| \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$ and $(\perp, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$ if $H = \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$ or $H = \mathtt{a\text{-}auth} \| \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$, for valid IDs $\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t$. The code within dashed boxes augments the basic converter with the receiver anonymity modes.

t will probably be smaller than 10, as the most prominent use-case for $t > 1$ is when sending a message to several devices of the recipient. In this regard, efficiency-wise, n is not relevant, whereas the larger t is, the more gain can be achieved via the defined modes, compared with running the same mode independently towards t recipients. Privacy-wise, n defines the size of the maximum possible anonymity set (but does not influence encryption/decryption times in any way) and t does not play a crucial role as there is no receiver anonymity anyway in the basic DIDComm modes analyzed above. When we move to the new modes with receiver anonymity in the next section, note that in general, the number t is the only leakage admitted by the new modes, and we give a very simple and effective trick how to conceal that leakage in practice.

6 Improving DIDComm

6.1 Combining Authcrypt and Anoncrypt - Efficiently

We describe a highly optimized encryption and decryption process for `a-auth` in Figure 9. If needed for disambiguation and comparison, we refer to this mode of operation by `merge-a-auth`, in contrast to the naive variant, which we refer by `naive-a-auth`. Recall that in `naive-a-auth`, the whole output of authcrypt (the tuple $(N, C, T, H, \boldsymbol{ek}, epk)$, cf. Figure 2) is serialized and fed to anoncrypt, which again encodes the recipients' identifiers as well as additional metadata – which can require many hundreds of bytes in some DID schemes – and performs fresh new key exchanges, re-encrypting the output by authcrypt. Our merged mode, `merge-a-auth`, avoids duplicating the key exchanges, and simply adds an extra key-encryption layer that hides the sender's identity. Upon receiving the message, a legitimate recipient "peels off" this extra key-encryption layer, revealing the sender's identity, which is then used to derive the final key-encryption key. We obtain the following theorem, whose proof is covered in Appendix E, as explained in Section 5.

Theorem 2 *The construction from Theorem 1 is still achieved when using the algorithms in Figure 9 for the combined mode* `a-auth` *in converter* π_i *from Figure 8.*

$\mathcal{E}_{\text{a-auth}}(sk_{\mathsf{ID}}, \mathsf{ID}, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t), M)$

// Content encryption
$cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$; $N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
$H \leftarrow \texttt{a-auth} \| \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$
$(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$

// Ephem. key-pair for *all* receivers
$(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$

foreach $i \in [t]$ **do**

  // Key agreement
  $Z_{\mathsf{e},i} \leftarrow (pk_{\mathsf{ID}_i})^{esk}$; $Z_{\mathsf{s},i} \leftarrow (pk_{\mathsf{ID}_i})^{sk_{\mathsf{ID}}}$
  $Z_i \leftarrow Z_{\mathsf{e},i} \| Z_{\mathsf{s},i}$

  // Key wrapping
  $kek'_i \leftarrow \mathsf{KDF}(Z_i, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
  $c_i \leftarrow \mathsf{DAE}.\mathcal{E}(kek'_i, cek)$
  $kek_i \leftarrow \mathsf{KDF}(Z_{\mathsf{e},i}, |\mathsf{DAE}.\mathcal{K}|, H)$
  $ek_i \leftarrow \mathsf{DAE}.\mathcal{E}(kek_i, \texttt{a-auth} \| \mathsf{ID} \| c_i)$

// Finalization
$\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
return $(H, (N, C, T, epk, \boldsymbol{ek}))$

$\mathcal{D}_{\text{a-auth}}(sk_{\mathsf{ID}'}, H, (N, C, T, epk, ek))$

// Key agreement and first layer unwrapping
$Z_{\mathsf{e}} \leftarrow epk^{sk_{\mathsf{ID}'}}$
$kek \leftarrow \mathsf{KDF}(Z_{\mathsf{e}}, |\mathsf{DAE}.\mathcal{K}|, H)$
$m \leftarrow \mathsf{DAE}.\mathcal{D}(kek, ek)$
if $\textsc{Parse}(m) = \perp$ **then return** $\perp$
$(\mathsf{ID}, c) \leftarrow \textsc{Parse}(m)$
$Z_{\mathsf{s}} \leftarrow (pk_{\mathsf{ID}})^{sk_{\mathsf{ID}'}}$
$Z \leftarrow Z_{\mathsf{e}} \| Z_{\mathsf{s}}$

// Content-key unwrapping
$kek' \leftarrow \mathsf{KDF}(Z, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
$cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek', c)$
if $cek = \perp$ **then return** $\perp$

// Content decryption
$M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
if $M \neq \perp$ **then return** (ID, M)
else return $\perp$

Figure 9: Merging anoncrypt and authcrypt efficiently, with the major additions/differences to/from `auth` and `anon` highlighted in gray. $\textsc{Parse}(m)$ returns (ID, c) if $m = \texttt{a-auth} \| \mathsf{ID} \| c$, with ID a valid identity and c a valid ciphertext for DAE, and $\perp$ otherwise.

6.2 Receiver Anonymity

The goal is to achieve an ideal DIDComm resource where the leakage function associated to the **snd** command of interface U_i is $\ell_{\mathsf{ra}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t) = t$. Thus, our proposal starts with the (mild) assumption that it is reasonable to leak the number of devices a receiver has (similar to leaking the length of the message). Similar to message's privacy, concealing additionally the number of devices can be achieved analogously: we note that DIDComm builds on top of (publicly accessible) DIDs, making it possible to estimate the most frequent number of devices/keys a DID owns, and adding dummy keys if needed to achieve a reasonable degree of uniformity.

Essentially, we propose to augment DIDComm with a private-receiver anoncrypt mode that removes all receiver identifiers from the header field, but still allows the receiver to detect which encrypted messages are aimed at him. While message detection techniques have been studied before [BLMG21], we propose a mechanism tailored to DIDComm's specifics. To build our `ra-anon` mode, formally specified in Figure 10, we propose to add one detection flag per recipient. Each of these detection flags is an encryption of 1, under a key that only the legitimate recipient may reproduce – as it is the result of a Diffie-Hellman key-exchange with the sender. The resulting ciphertext has the same size than in the normal anoncrypt mode, although the associated header H –where we include the encrypted flags – now grows linearly with the number of recipients, but only requires a number of bytes equal to the hash length per recipient, which seems reasonable. The extra computation overhead is, on the sender's side, one extra KDF evaluation and deterministic symmetric encryption (of one bit) per recipient – both efficient operations. The computation overhead for receivers is simply one KDF evaluation and, on average, $t/2$ deterministic symmetric decryptions (of a one-bit message), where t is the number of potential recipients. Moreover, this proposal is RFC-compliant, as we are essentially proposing a new method to encode recipients' keys into the header H, which is something that RFC7518 already requires to concrete implementations [Jon15, Section 4.6.2].

Lastly, observe that the final receivers do not need to bulk download all the ciphertexts from the last mediator: it is enough with downloading the tuple $(H, epk, \boldsymbol{ek})$ associated to each encrypted message. With that, they can already detect whether a message is aimed at them, and later request the full ciphertext (and even when it is not aimed at them, in order to reduce correlation risk).

6.3 Receiver-Anonymous Combined Mode

We present as a final contribution a concrete mode of operation, `ra-a-auth`, that achieves all security properties efficiently in one mode. That is, a sender- and receiver-anonymous mode of operation (anonymous against any network attacker) but where the sender can authenticate itself to the final receiver. Its internal operation is direct, given `ra-anon` and `merge-a-auth`: one simply has to include the encrypted flags as done in `ra-anon` and, in addition, reencrypt the key encryption keys, giving them the extra structure that includes the sender's identity, as done in `merge-a-auth`. For lack of space, we defer the full specification to Appendix C.

6.4 Security Analysis of Augmented DIDComm

The two new modes that achieve receiver-anonymity presented above can be formalized as a protocol in our framework as depicted in Figure 8 with the code of the dash boxes included. Formally, we are able to show that using the new modes at most leak the number of recipients, i.e., the worst case leakage is

$$\ell_{\mathsf{ra}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t) := (\ell_{\mathsf{net}}(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_t), t).$$

This formalizes the aforementioned anonymity-preservation: nothing else is leaked, beyond what the network layer leaks.

Theorem 3 *For any $n \geq 1$, and leakage function ℓ_{net}, consider the real resources PKI_n from Figure 4 and $\mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}},\mathcal{C}}$ from Figure 5, the augmented ideal resource $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\ell_{\mathsf{ra}},\mathcal{C}}$ from Figure 7 with leakage*

$\mathcal{E}_{\text{ra-anon}}(pk_{B_1}, \ldots, pk_{B_t}, M)$

// Content encryption key and nonce
$cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$; $N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$

// Ephem. key-pair for *all* receivers
$(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$

// Header computation
$H \leftarrow$ `ra-anon`
foreach $i \in [t]$ **do**
  // Key agreement
  $Z_i \leftarrow (pk_{B_i})^{esk}$
  // Internal header value for flagging
  $fk_i \leftarrow \mathsf{KDF}(Z_i, |\mathsf{AEAD}.\mathcal{K}|)$
  $N_i \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
  $(C_i, T_i) \leftarrow \mathsf{AEAD}.\mathcal{E}(fk_i, N_i, \texttt{ra-anon}, 1)$
  // Extend header
  $H \leftarrow H \| (N_i, C_i, T_i)$
// Content encryption
$(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$
foreach $i \in [t]$ **do**
  // Key wrapping
  $kek_i \leftarrow \mathsf{KDF}(Z_i, |\mathsf{DAE}.\mathcal{K}|, H)$
  $ek_i \leftarrow \mathsf{DAE}.\mathcal{E}(kek_i, cek)$
// Finalization
$\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
return $(H, (N, C, T, epk, \boldsymbol{ek}))$

$\mathcal{D}_{\text{ra-anon}}(sk_B, H, (N, C, T, epk, \boldsymbol{ek}))$

// Key agreement
$Z \leftarrow epk^{sk_B}$
`ra-anon`$\|H_1\| \cdots \|H_t \leftarrow H$
// Flag detection
$fk \leftarrow \mathsf{KDF}(Z, |\mathsf{AEAD}.\mathcal{K}|)$
$j \leftarrow 0$
foreach $i \in [t]$ **do**
  $(N_i, C_i, T_i) \leftarrow H_i$
  if $\mathsf{AEAD}.\mathcal{D}(fk, N_i, \texttt{ra-anon}, \cdots$
    $\cdots C_i, T_i) = 1$ **then**
    $j \leftarrow i$
if $j = 0$ **then**
  abort
// Content Key unwrapping
$kek \leftarrow \mathsf{KDF}(Z, |\mathsf{DAE}.\mathcal{K}|, H_j)$
$cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek, ek_j)$
// Message decryption
$M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
return M // $\in \mathsf{AEAD}.\mathcal{M} \cup \{\perp\}$

Figure 10: An anonymous-receiver anoncrypt mode of operation.

functions ℓ_{rcv} as in Theorem 1 and ℓ_{ra} as defined above, and the augmented DIDComm protocol from Figure 8. Then, for a negligible function ε,

$$[\mathsf{PKI}_n, \mathsf{A\text{-}INS}_{n,\ell_{\mathsf{net}},\mathcal{C}}] \xrightarrow{\boldsymbol{\pi}_{\mathsf{DIDComm}},\varepsilon} \mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell_{\mathsf{rcv}},\ell_{\mathsf{ra}},\mathcal{C}}. \tag{2}$$

7 Benchmarks

We report now on the results obtained from prototypical implementations of our proposals, available in [BBD23].

As baseline, we evaluate `anon` and `auth`, as well as `naive-a-auth`. We measure encryption and decryption times, DIDComm message sizes, and maximum memory consumption of encryption (decryption is always less demanding). The results are depicted in Figure 11. All tests are run for 1 to 6 recipients (using multiplex encryption), in an Acer Swift3 laptop with 8 Intel Core i7 and 16 GB of RAM. The graphs depict the mean value over 1000 iterations. We use as DID strings sample values like "`did:example:bob#key0`". This is

relevant, as these values are explicitly encoded within the resulting DIDComm message. Indeed, some popular DID systems (e.g. `did:peer`) use as DID strings an encoding of a full list of public keys, which may take up to several hundreds of bytes. Thus, removing duplicates is especially critical and the results shown in our analysis are probably very conservative in that regard.

Computation- and communication-wise, `merge-a-auth` largely improves `naive-a-auth`, while `ra-anon` and `ra-a-auth` incur in very acceptable overhead. Memory-wise, our new modes are slightly more costly (about 1.5% than `naive-a-auth`). However, it is likely that this can still be largely improved with more efficient coding.

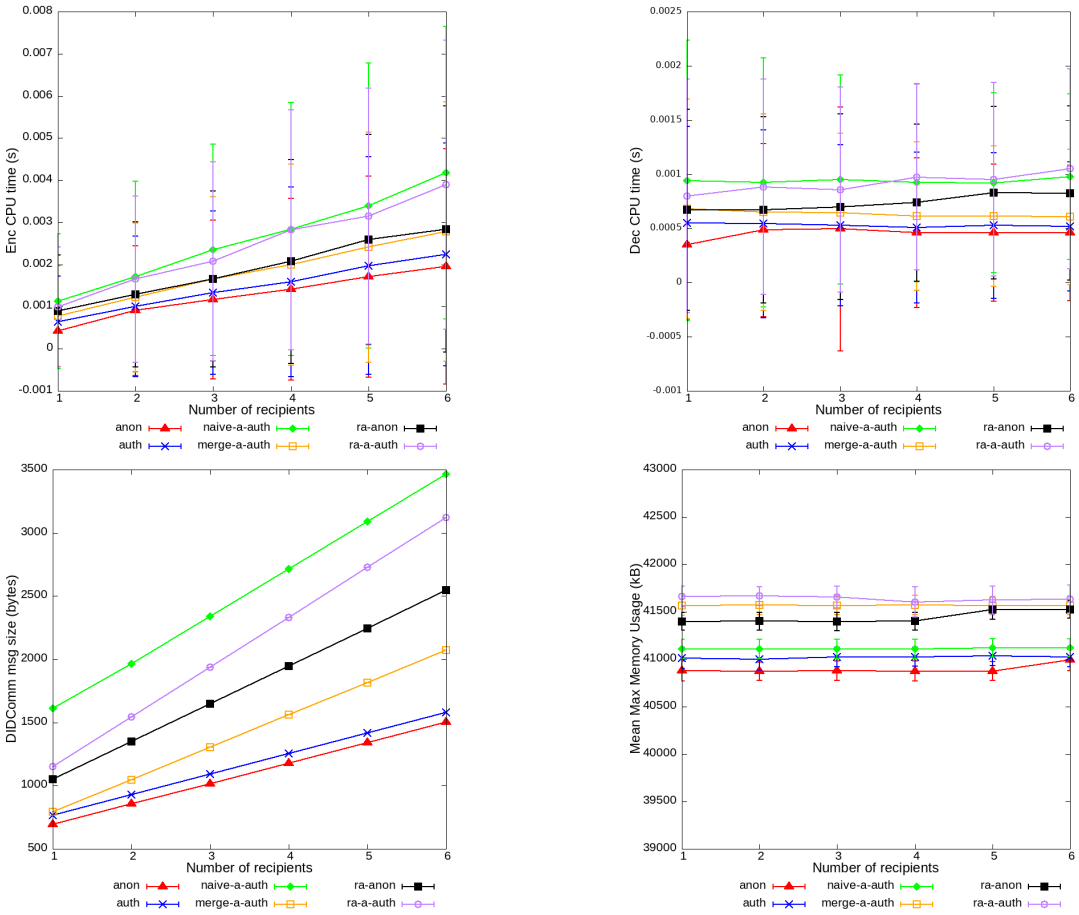

Figure 11: Benchmarks for the basic DIDComm modes and our proposals.

`merge-a-auth` mode While `naive-a-auth` adds up the costs of independent `anon` and `auth` runs, `merge-a-auth` just adds a KDF evaluation and deterministic encryption per recipient to the cost of `auth`. Space-wise, we add some structure to the encrypted keys, but the overhead is worth it. Especially, since we avoid duplicated encodings of identifiers (the B_i's in the H field). Overall, `merge-a-auth` reduces almost by a factor of 2 the costs of `naive-a-auth`, except memory-wise, for which it increases costs by 1%.

`ra-anon` mode In this mode, encryption and decryption costs have a strong dependency on the number of recipients – we add as many flag encryptions as recipients at encryption time, and to the trial decryption at decryption time. Still, the costs seem reasonable. The same applies to memory needs, and message size, where the extra linear growth is due to the additional encrypted flags.

`ra-a-auth` **mode** In this mode, we add to `merge-a-auth` the computational and space costs associated to the encrypted flags and trial decryptions. Still, it is worth noting that the final costs remain below than those of `naive-a-auth`– excluding memory needs, with only a very small increase of about 1.5%

8 Lessons Learned

DIDComm is a new communications framework focused on DIDs, aiming to uphold their decentralized philosophy. This results in significant changes in message routing and poses challenges for new developers. For instance, intermediate hops can alter the transport medium without sender or recipient awareness. Nevertheless, the security of the cryptographic layer (DIDComm messaging) and interoperability are essential. To meet interoperability requirements, DIDComm uses simple and effective technologies like JSON and cryptographic specifications such as JWA [Jon15].

In summary, when analyzing and proposing improvements on deployed protocols, one encounters inherent tradeoffs between security and other dimensions, such as deployability, code simplification, or efficiency. We summarize next some of the learnings from our work in this regard.

Deployability vs security and efficiency. Analyzing security with simple primitives allows the use of known techniques. However, proposing improvements or fixes is harder, as it requires adhering to existing technologies and standards to maximize adoption, even if it sacrifices efficiency. Candidate improvements should also avoid adding unrealistic trust assumptions. For example, using broadcast encryption (BE) [BGW05] in DIDComm messaging could optimize encryption for multiple recipients. However, to the best of our knowledge, current BE schemes are not supported by the JWA stack, requiring a significant shift in DIDComm. Additionally, BE schemes need a trusted setup for a master secret, which contradicts DIDComm's decentralized model. Efficiency gains from BE are also limited unless multi-recipient encryption is widely used.

Code simplification vs security and efficiency. Engineering-wise, reusing code reduces the risk of coding vulnerabilities, but it can be risky when combining cryptographic primitives. In DIDComm, to achieve sender anonymity while allowing the receiver to verify the sender's identity, authcrypt and anoncrypt are applied sequentially. This combination, however, does not provide the expected guarantees when using non-committing encryption in anoncrypt. Formal security analysis reveals that non-committing encryption allows adversaries to specify different messages to different recipients, resulting in a weaker resource. This issue is easily fixed by using committing encryption schemes.

Naively composing cryptographic primitives can also impact costs, and this impact may be significant or not. For our improved `merge-a-auth` mode, the computational costs, despite nearly halving, are still in the order of milliseconds, making the gains modest. However, space costs are significantly impacted, with a 2x improvement translating to hundreds of bytes, which is critical when chaining multiple hops in DIDComm. Nevertheless, in this case, since the improvements are based on existing standards, implementing the improved version does not add much complexity to the code and may be worthwhile.

Composable analysis as basis for future analyses. Another important aspect worth noting is that, having chosen a composable security framework that allows us to model a generic communication pattern is beneficial when extending our work to cover a larger set of protocols that deal with cryptographic communication protocols between DIDs that fit into the same pattern. More precisely, analyzing routing in DIDComm would boil down to showing which type of network resource, i.e., which leakage function ℓ_{net} of $\mathsf{A\text{-}INS}$ DIDComm routing actually achieves over the Internet. Thus, the final leakage of DIDComm messaging and DIDComm routing can be obtained by composition. Even more, such a routing analysis will use the same encryption modes, where the results of this paper can be invoked again thanks to our parameterizable assumed network (with appropriate parameters as the underlying network resource is then the Internet with presumably no protection).

9 Conclusion

We have given the first formal model of (the encryption modes of) DIDComm, a popular framework for secure communications using Decentralized Identifiers (DIDs). Our model follows the Constructive Cryptography paradigm, ensuring composability. While proving security, we identified a subtle (easily fixable) issue, observed that DIDComm leaks more information beyond what is leaked by the network, and proposed improved and extended encryption modes that increase efficiency and preserve privacy.

For future work, two main lines stand out. First, in our analysis we have covered DIDComm's encryption modes. However, it also supports signing – which we have not looked into. Second, DIDComm also includes higher-level protocols that build on its encryption (and signing) modes, and that we leave out of scope as well, like the routing protocol, which defines the mechanism to establish a communication path between sender and receipient.

Acknowledgements

Work by the second author was done partly while he was at ETH Zurich. The authors would like to thank Daniel Hardman and Sam Curren for their help in addressing doubts regarding DIDComm. The authors would also like to thank the anonymous CCS reviewers for their constructive feedback during the review phase.

References

[ABD+15] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *22nd ACM Conference on Computer and Communications Security*, October 2015.

[AFNV19] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 701–731. Springer, 2019. doi:10.1007/978-3-030-26951-7_24.

[ASS+16] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN: Breaking TLS with SSLv2. In *25th USENIX Security Symposium*, August 2016.

[BBD+15] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 535–552. IEEE Computer Society, 2015. doi:10.1109/SP.2015.39.

[BBD23] Christian Badertscher, Fabio Banfi, and Jesus Diaz. Supplementary material. https://github.com/jesusdiazvico/didcomm-privacy-benchmarks, September 2023.

[BBKS07] Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007. doi:10.1109/TIT.2007.907471.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000. doi:10.1007/3-540-45539-6_18.

[BBS03] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2003. doi:10.1007/3-540-36288-6_7.

[BCD20] Elaine Barker, Lily Chen, and Richard Davis. Recommendation for key-derivation methods in key-establishment schemes. Technical report, National Institute of Standards and Technology, 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf.

[BCK+18] Elaine Barker, Lily Chen, Sharon Keller, Allen Roginsky, Apostol Vassilev, and Richard Davis. Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. Technical report, National Institute of Standards and Technology, 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005. doi:10.1007/11535218_16.

[BLMG21] Gabrielle Beck, Julia Len, Ian Miers, and Matthew Green. Fuzzy message detection. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1507–1528. ACM, 2021. doi:10.1145/3460120.3484545.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. pages 136–145, 2001. doi:10.1109/SFCS.2001.959888.

[CLT22] Sam Curren, Tobias Looker, and Oliver Terbu. Didcomm messaging v2.x editor's draft. https://identity.foundation/didcomm-messaging/spec/, 2022.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX, 2004. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[ESG22] Edward Eaton, Sajin Sasy, and Ian Goldberg. Improving the privacy of tor onion services. In Giuseppe Ateniese and Daniele Venturi, editors, *Applied Cryptography and Network Security - 20th International Conference, ACNS 2022, Rome, Italy, June 20-23, 2022, Proceedings*, volume 13269 of *Lecture Notes in Computer Science*, pages 273–292. Springer, 2022. doi:10.1007/978-3-031-09234-3_14.

[FOR17] Pooya Farshim, Claudio Orlandi, and Razvan Rosie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symmetric Cryptol.*, 2017(1):449–473, 2017. URL: https://doi.org/10.13154/tosc.v2017.i1.449-473, doi:10.13154/TOSC.V2017.I1.449-473.

[Fou23] Identity Foundation. Didcomm website. https://didcomm.org/, May 2023.

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97. Springer, 2017. doi:10.1007/978-3-319-63697-9_3.

[Gov23a] Government of British Columbia. BC Wallet. https://digital.gov.bc.ca/digital-trust/projects-and-initiatives/bc-wallet-technology-overview/, May 2023.

[Gov23b] Government of British Columbia. OrgBook BC. https://digital.gov.bc.ca/digital-trust/projects-and-initiatives/projects-overview/, May 2023.

[HC22] Daniel Hardman and Sam Curren. Private communication, 2022.

[HS02] Russ Housley and Jim Schaad. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394, October 2002. URL: https://www.rfc-editor.org/info/rfc3394, doi:10.17487/RFC3394.

[HTT18] Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. The multi-user security of gcm, revisited: Tight bounds for nonce randomization. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1429–1440. ACM, 2018. doi:10.1145/3243734.3243816.

[Hyp19] Hyperledger Aries. Aries RFC 0005: DID Communication. https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0005-didcomm/README.md, November 2019.

[Ide23] Identity Foundation. DIDComm Messaging v2.x Editor's Draft. https://identity.foundation/didcomm-messaging/spec/, April 2023.

[Inp23] Input Output Global. Atala PRISM. https://atalaprism.io/, May 2023.

[JBHD12] Rob Jansen, Kevin S. Bauer, Nicholas Hopper, and Roger Dingledine. Methodically modeling the tor network. In Sean Peisert and Stephen Schwab, editors, *5th Workshop on Cyber Security Experimentation and Test, CSET '12, Bellevue, WA, USA, August 6, 2012*. USENIX Association, 2012. URL: https://www.usenix.org/conference/cset12/workshop-program/presentation/jansen.

[JH15] Michael Jones and Joe Hildebrand. JSON Web Encryption (JWE). RFC 7516, May 2015. URL: https://www.rfc-editor.org/info/rfc7516, doi:10.17487/RFC7516.

[Jon15] Michael Jones. JSON Web Algorithms (JWA). RFC 7518, May 2015. URL: https://www.rfc-editor.org/info/rfc7518, doi:10.17487/RFC7518.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, 2010. doi:10.1007/978-3-642-14623-7_34.

[Kur02] Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2002. doi:10.1007/3-540-45664-3_4.

[Lab23] SICPA Digital Lab. didcomm-jvm implementation. https://github.com/sicpa-dlab/didcomm-jvm, Mar 2023. Commit 993265c.

[lep20] lepture. authlib. https://github.com/lepture/authlib, 2020.

[MÏ14] Bodo Möller. https://security.googleblog.com/2014/10/this-poodle-bites-exploiting-ssl-30.html, October 2014.

[Mad21] Neil Madden. Public Key Authenticated Encryption for JOSE: ECDH-1PU. Internet-Draft draft-madden-jose-ecdh-1pu-04, Internet Engineering Task Force, May 2021. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-madden-jose-ecdh-1pu/04/.

[Mau12] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Theory of Security and Applications – TOSCA 2011*, pages 33–56, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *ICS 2011*, pages 1–21. Tsinghua University Press, 2011.

[PvdM16] Kenneth G. Paterson and Thyla van der Merwe. Reactive and proactive standardisation of TLS. In Lidong Chen, David A. McGrew, and Chris J. Mitchell, editors, *Security Standardisation Research - Third International Conference, SSR 2016, Gaithersburg, MD, USA, December 5-6, 2016, Proceedings*, volume 10074 of *Lecture Notes in Computer Science*, pages 160–186. Springer, 2016. doi:10.1007/978-3-319-49100-4_7.

[RD12] Juliano Rizzo and Thai Duong. https://threatpost.com/crime-attack-uses-compression-ratio-tls-requests-side-channel-hijack-secure-sessions-091312/77006/, September 2012.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: https://www.rfc-editor.org/info/rfc8446, doi:10.17487/RFC8446.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 98–107. ACM, 2002. doi:10.1145/586110.586125.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006. doi:10.1007/11761679_23.

[SB16] Paul Syverson and Griffin Boyce. Bake in .onion for tear-free and stronger website authentication. *IEEE Secur. Priv.*, 14(2):15–21, 2016. doi:10.1109/MSP.2016.33.

[sd21] sicpa dlab. didcomm-python. https://github.com/sicpa-dlab/didcomm-python, 2021.

[SFEB21] Paul Syverson, Matthew Finkel, Saba Eskandarian, and Dan Boneh. Attacks on onion discovery and remedies via self-authenticating traditional addresses. In *WPES '21: Proceedings of the 20th Workshop on Privacy in the Electronic Society, Korea, 15 November 2021*, pages 45–52. ACM, 2021. doi:10.1145/3463676.3485610.

[Syv17] Paul Syverson. The once and future onion. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I*, volume 10492 of *Lecture Notes in Computer Science*, pages 18–28. Springer, 2017. doi:10.1007/978-3-319-66402-6_3.

[Tec23] Technometria. Building an SSI Ecosystem: MemberPass and Credit Unions. https://www.windley.com/archives/2021/06/building_an_ssi_ecosystem_memberpass_and_credit_unions.shtml, May 2023.

[Ver23] Veramo. Performant and modular APIs for Verifiable Data and SSI. , May 2023.

[W3C22] W3C. Decentralized Identifiers (DIDs) v1.0. https://www.w3.org/TR/did-core/, July 2022.

A Standards Followed by AnonCrypt and AuthCrypt

AnonCrypt follows the specification of JSON web algorithms' (JWA) `ECDH-ES`, as specified in RFC 7518 [Jon15], whereas AuthCrypt follows the newer JWA's `ECDH-1PU` RFC draft [Mad21]. Both methods are based on NIST 800-56A recommendation [BCK+18]: `ECDH-1PU`, which stands for ECDH *one-pass unified model*, is defined therein in section 6.2.1.2 as "C(1e, 2s, ECC CDH)", while `ECDH-ES`, which stands for ECDH *ephemeral static*, is defined therein in section 6.2.2.2 as "C(1e, 1s, ECC CDH)".

Both AnonCrypt and AuthCrypt use the *"One-Step KDF"* (with SHA-256) specified in [BCD20] to derive the KEK from the shared secret resulting from the key agreement, following the recommendations given in Section 5.8.2.1 of [BCK+18]. Even though key-wrapping is optional as per JWA, DIDComm messaging editor's draft [CLT22, section 5.1.4] mandates it for both AuthCrypt and AnonCrypt. Specifically, the AES key wrap algorithm defined in RFC 3394 [HS02] is used to encrypt the CEK using the KEK. Additionally, [CLT22, section 5.1.3] also declares AES256-CBC with HMAC-SHA512 (`A256CBC-HS512`) as the *required* encryption scheme for both AuthCrypt and AnonCrypt, and AES256-GCM (`A256GCM`) as *recommended* for AnonCrypt only. Additionally, it declares the AEAD scheme XChaCha20Poly1305 (`XC20P`) as *optional*. Moreover, in the same section the documents states that *"implementations MUST choose nonces securely"*, and both [lep20, `authlib/jose/drafts_jwe_models.py`] and [JH15, Section 3.3] suggest this should be achieved by generating uniformly random initialization vectors.

B On the Weaker Ideal Functionality

Lack of commitment. Compared to the strong ideal resource we achieve, this one allows a malicious sender, in one go, cast different messages to different recipients. This is due to admitting in `anon` (and consequently also `anon(auth(.))`) AEAD schemes that are not (compactly) committing. This is in contrast to the stronger resource, where each send-event is always a commitment to the message, i.e., in one send event, all receivers either receive the same message or no message. Here, a malicious sender can specify different messages to different parties and as such, a party that receives as output (ID, M) cannot conclude that this can be traced back to a unique send-event where the entity ID sent the message M. Hence, the weaker resource can be obtained by replacing the adversary command **snd** by one that takes as input ID and a list $(M_1, \mathsf{ID}_1), \ldots, (M_n, \mathsf{ID}_n)$, and which to distributes n messages to n recipients in one go, capturing the worst case.

C Code for the Receiver-Anonymous Combined Mode

We give in Figure 12 the code of the receiver-anonymous combined mode, as introduced in Section Section 6.3.

D Constructive Cryptography

In the framework of *constructive cryptography* (CC) [MR11, Mau12], security of a cryptographic scheme is not defined as a game between an adversary and a challenger offering some oracles, but rather as the indistinguishability of two worlds. More precisely, one defines the two worlds as *resources*, a concept analogous to *functionalities* in universal composability [Can01]. The real world resource is made up of a set of *assumed*

$\mathcal{E}_{\texttt{ra-a-auth}}(sk_A,\, pk_{B_1}, \ldots, pk_{B_t}, M)$

// Content encryption key and nonce
$cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$
$N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$

// Ephem. key-pair for *all* receivers
$(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$

// Header computation
$H \leftarrow \texttt{ra-a-auth}$
foreach $i \in [t]$ **do**
  // Key agreement
  $Z_{\mathsf{e},i} \leftarrow (pk_{B_i})^{esk}$
  // Internal header value for flagging
  $fk_i \leftarrow \mathsf{KDF}(Z_{e,i}, |\mathsf{AEAD}.\mathcal{K}|)$
  $N_i \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
  $(C_i, T_i) \leftarrow \mathsf{AEAD}.\mathcal{E}(fk_i, N_i, \texttt{ra-a-auth}, 1)$
  // Extend header
  $H \leftarrow H \| (N_i, C_i, T_i)$
// Content encryption
$(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$
foreach $i \in [t]$ **do**
  // Key agreement
  $Z_{\mathsf{s},i} \leftarrow (pk_{B_i})^{sk_A}$
  // Key wrapping
  $kek_{0,i} \leftarrow \mathsf{KDF}(Z_{\mathsf{e},i}, |\mathsf{DAE}.\mathcal{K}|, H\|T)$
  $kek_{1,i} \leftarrow \mathsf{KDF}(Z_{\mathsf{e},i}\|Z_{\mathsf{s},i}, |\mathsf{DAE}.\mathcal{K}|, H\|T)$
  $c_i \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{1,i}, cek)$
  $ek_i \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{0,i}, \texttt{ra-a-auth}\|A\|c_i)$
// Finalization
$\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
return $(H, (N, C, T, epk, \boldsymbol{ek}))$

$\mathcal{D}_{\texttt{ra-a-auth}}(sk_B,\, H, (N, C, T, epk, \boldsymbol{ek}))$

// Key agreement and first layer unwrapping
$Z_{\mathsf{e}} \leftarrow epk_B^{sk}$
$\texttt{ra-a-auth}\|H_1\|\cdots\|H_t \leftarrow H$
// Flag detection
$fk \leftarrow \mathsf{KDF}(Z, |\mathsf{AEAD}.\mathcal{K}|)$
$j \leftarrow 0$
foreach $i \in [t]$ **do**
  $(N_i, C_i, T_i) \leftarrow H_i$
  if $\mathsf{AEAD}.\mathcal{D}(fk, N_i, \texttt{ra-a-auth}, \cdots \cdots C_i, T_i) = 1$ **then**
    $j \leftarrow i$
if $j = 0$ **then**
  abort
$kek_0 \leftarrow \mathsf{KDF}(Z_{\mathsf{e}}, |\mathsf{DAE}.\mathcal{K}|, H\|T)$
$M_0 \leftarrow \mathsf{DAE}.\mathcal{D}(kek_0, ek_j)$
if $M_0 \neq \texttt{ra-a-auth}\|A\|\cdots$ **then**
  return $\perp$
$Z_{\mathsf{s}} \leftarrow (pk_A)^{sk_B}$
// Content-Key unwrapping
$kek_1 \leftarrow \mathsf{KDF}(Z_{\mathsf{e}}\|Z_{\mathsf{s}}, |\mathsf{DAE}.\mathcal{K}|, H\|T)$
$cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek_1, c)$
if $cek = \perp$ **then**
  return $\perp$
// Content decryption
$M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
if $M \neq \perp$ **then**
  return (A, M)
else
  return $\perp$

Figure 12: A merged operation of anoncrypt and authcrypt including receiver anonymity.

resources, composed in parallel, to which a *converter* is attached. The converter implements a protocol that uses the cryptographic scheme whose security one wants to define. The ideal world resource is made up of an *ideal* resource, to which also a spacial converter is attached, called the *simulator*.

As a basic example, think of the scheme being regular symmetric-key encryption. To define its security in CC, we need to specify what the assumed resources are, what the protocol using the scheme while attached to those resources is, what the ideal resource is, and finally how the simulator behaves. For this specific

example, we think of a setting where a sender Alice wants to send messages *confidentially* to Bob.

As assumed resources, we specify an *insecure channel* INS between Alice and Bob and a shared key KEY. The insecure channel INS allows Alice to input the ciphertext, and outputs the ciphertext to Bob, but also to the *adversary* Eve. Moreover, Eve can also input a ciphertext for Bob to INS. When Bob gets the ciphertexts from INS, he does not see who the sender was. The shared key KEY simply samples a key for the symmetric-key encryption scheme, according to the specific key-generation algorithm, and outputs it to both Alice and Bob.

The protocol $\boldsymbol{\pi}$, attached to both INS and KEY, is defined as the natural way to use the scheme. Now, Alice and Bob do not have direct access to INS and KEY anymore, $\boldsymbol{\pi}$ will elaborate their inputs and outputs to and from those resources. Concretely, $\boldsymbol{\pi}$ first collects the key K from KEY. Then, on input a message M from Alice, $\boldsymbol{\pi}$ encrypts M into $C := \mathsf{Enc}_K(M)$, and sends the ciphertext C to Bob over INS. Once a ciphertext C' is obtained by $\boldsymbol{\pi}$ from INS (which could have been send by either Alice or Eve), it is decrypted into $M' := \mathsf{Dec}_K(C')$ by $\boldsymbol{\pi}$, and output to Bob. The protocol attached to the resources defines the whole real resource, denoted $\mathsf{REAL} := \boldsymbol{\pi}[\mathsf{KEY}, \mathsf{INS}]$.

As ideal resource, we specify a *confidential channel* CNF between Alice and Bob. The confidential channel CNF allows Alice to input the message, and outputs the message to Bob, but to the adversary Eve it only outputs the length of the message. On the other end, and in the insecure channel, Eve can still input a message for Bob to CNF, and Bob can not see who the sender was (we are capturing confidentiality, not authenticity).

The simulator $\boldsymbol{\sigma}$, attached to CNF, will elaborate Eve's inputs and outputs to and from CNF. Concretely, $\boldsymbol{\sigma}$ first samples a key K according to the specific key-generation algorithm.

Then, on input a ciphertext C from Eve, $\boldsymbol{\sigma}$ decrypts C into $M := \mathsf{Dec}_K(C)$, and sends the message M to Bob over CNF. Once a length ℓ of a message is obtained by $\boldsymbol{\sigma}$ from CNF, a random message $\tilde{M}$ of length ℓ is sampled and encrypted into $\tilde{C} := \mathsf{Enc}_K(\tilde{M})$, and then output to Eve. The simulator attached to the resource CNF defines the whole ideal resource, denoted $\mathsf{IDEAL} := \boldsymbol{\sigma}\,\mathsf{CNF}$.

Finally, since both REAL and IDEAL have the same interfaces, we can consider a distinguisher trying to tell them apart. If any distinguisher can succeed with only negligible probability, then we can deem the symmetric-key encryption scheme used by $\boldsymbol{\pi}$ to be secure. In this case, we say that the symmetric-key encryption scheme *constructs* CNF from INS and KEY, denoted $[\mathsf{KEY}, \mathsf{INS}] \xrightarrow{\boldsymbol{\pi}} \mathsf{CNF}$.

A nice feature of CC security definitions, is that they are *composable*. Let us explain with a slightly different version of the example above. We could also define security of a symmetric-key encryption scheme as constructing a *secure channel* SEC from an *authenticated channel* AUT and KEY via a protocol $\boldsymbol{\pi}_1$ and simulator $\boldsymbol{\sigma}_1$. The secure channel SEC is like CNF, but does not allow Eve to input anything, and the authenticated channel AUT is like INS, but also does not allow Eve to input anything. The new definition would be captured by the statement $[\mathsf{KEY}, \mathsf{AUT}] \xrightarrow{\boldsymbol{\pi}_1} \mathsf{SEC}$. On the other hand, we could define security of a MAC scheme as constructing AUT from INS and KEY via protocol $\boldsymbol{\pi}_2$, denoted $[\mathsf{KEY}, \mathsf{INS}] \xrightarrow{\boldsymbol{\pi}_1} \mathsf{AUT}$.

Now since we can construct AUT from INS using $\boldsymbol{\pi_1}$ and SEC from AUT using $\boldsymbol{\pi_2}$, it would seem natural that composing the two protocols $\boldsymbol{\pi_2}$ and $\boldsymbol{\pi_1}$ sequentially, would result in a protocol constructing SEC directly from INS. This is indeed the case, and in fact from CC's *composition theorem* we have that

$$\begin{aligned}[\mathsf{KEY}, \mathsf{INS}] \xrightarrow{\boldsymbol{\pi}_1} \mathsf{AUT} \quad \text{and} \quad [\mathsf{KEY}, \mathsf{AUT}] \xrightarrow{\boldsymbol{\pi}_1} \mathsf{SEC} \\ \Longrightarrow [\mathsf{KEY}, \mathsf{KEY}, \mathsf{INS}] \xrightarrow{\boldsymbol{\pi}_2\boldsymbol{\pi}_1} \mathsf{SEC}.\end{aligned}$$

The main advantage of composition is that, concretely for this example, proving security of the symmetric-key encryption scheme and of the MAC scheme is sufficient, and security of the *Encrypt-then-MAC* scheme requires no further proof, since it is implied.

E Additional Definitions and Proofs

E.1 Security Notions for Key-Derivation Function (KDF)

Following [Kra10], a KDF is considered secure if, on input (σ, L, c), its output is pseudorandom, for $(\sigma, \alpha) \leftarrow \Sigma$, and where only the auxiliary knowledge α is made public.[2]

Definition 5 *A KDF* KDF *is secure w.r.t. source* Σ *if no adversary* $\mathcal{A}$ *given* α*, where* $(\sigma, \alpha) \leftarrow \Sigma$*, can distinguish with non-negligible advantage between an oracle returning* $\mathsf{KDF}(\sigma, L, c)$ *on input* (c, L)*, and one returning a uniformly distributed bitstring of length* L*, where* $\mathcal{A}$ *is restricted to not repeat any input* c*. We define* $\mathcal{A}$*'s advantage as* $\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KDF}}(\mathcal{A}) := 2 \cdot \Pr[\hat{b} = b] - 1$*, where* b *is a uniform bit selecting one of the two experiments and* $\hat{b}$ *is* $\mathcal{A}$*'s output.*

E.2 Security Notions for Deterministic AE (DAE)

Following [RS06], also the notion of security for DAE combines the two properties of confidentiality and unforgeability into a single notion.

Definition 6 *A DAE scheme* DAE *is* DAE-secure *if, for any efficient adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{dae}}_{\mathsf{DAE}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathcal{E}(K,\cdot),\mathcal{D}(K,\cdot)} = 1 \mid K \leftarrow \mathcal{G}] - \Pr[\mathcal{A}^{\$(\cdot),\perp(\cdot)} = 1]$$

is negligible, if $\mathcal{A}$ *is neither allowed to repeat queries nor to query the second oracle with answers from the first (to avoid trivial wins).*

E.3 Security Notions for AE with Associated Data (AEAD)

Following [Rog02], we require an AEAD scheme to provide both confidentiality (IND$-CPA) and unforgeability (INT-CTXT). We combine these two properties into a single notion as per [Rog02, RS06], and adapt such notion to the *multi-user* setting, as introduced in [BBM00], loosely following the syntax from [HTT18]. Note that for the specific case of GCM, tight multi-user bounds were established in [HTT18].

$\mathbf{Real}_{\mathsf{AEAD}}$	$\mathbf{Rand}_{\mathsf{AEAD}}$
$K_1, K_2, \ldots \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$	**procedure** $\textsc{Enc}(i, N, H, M)$:
procedure $\textsc{Enc}(i, N, H, M)$:	$(C, T) \xleftarrow{\$} \{0,1\}^{\lvert M\rvert} \times \{0,1\}^{\tau}$
return $\mathsf{AEAD}.\mathcal{E}(K_i, N, H, M)$	**return** (C, T)
procedure $\textsc{Dec}(i, N, H, C, T)$:	**procedure** $\textsc{Dec}(i, N, H, C, T)$:
return $\mathsf{AEAD}.\mathcal{D}(K_i, N, H, C, T)$	**return** $\perp$

Figure 13: Games defining the multi-user security of an AE scheme AEAD.

Definition 7 *An AEAD scheme* AEAD *with tag length* τ *is* muAE$-secure *if, for any efficient adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathrm{muae\$}}_{\mathsf{AEAD}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{Real}_{\mathsf{AEAD}}} = 1] - \Pr[\mathcal{A}^{\mathbf{Rand}_{\mathsf{AEAD}}} = 1]$$

is negligible, where games $\mathbf{Real}_{\mathsf{AEAD}}$ *and* $\mathbf{Rand}_{\mathsf{AEAD}}$ *are defined in fig. 13. Moreover,* $\mathcal{A}$ *is neither allowed to repeat pairs* (i, N) *to its* Enc *oracle, nor to query the* Dec *with answers from* Enc *(to avoid trivial wins).*

We further require an AEAD scheme to be *committing* [FOR17], that is, an honestly generated ciphertext (C, T) only decrypts to a non-$\perp$ message M if the same values K, N, H used to encrypt are used to decrypt.

[2] In the original definition of [Kra10], a KDF also specifies a *salt*, but since the KDF used by DIDComm does not, we adapted the notation.

E.4 Security Proofs

Proof 1 (Proof of Theorem 1) *Recall that, by Definition 4, the statement from Equation* (1) *means that for any corrupt set* $\mathcal{C} \subseteq \{\mathsf{U}_1, \ldots, \mathsf{U}_n\}$ *and corresponding honest set* $\mathcal{H} = \mathcal{U} \setminus \mathcal{C}$*, there exists a simulator* $\boldsymbol{\sigma}$ *such that*

$$\boldsymbol{\pi}^{\mathcal{H}}_{\mathsf{DIDComm}}[\mathsf{PKI}_n, \mathsf{A\text{-}INS}_{n,\ell}] \approx_{\varepsilon} \boldsymbol{\sigma}^{\mathcal{C}\cup\{\mathsf{E}\}}\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell,\mathcal{C}}. \tag{3}$$

We will prove Equation (3) *for the protocol* $\boldsymbol{\pi}_{\mathsf{DIDComm}}$ *from fig. 8 obtained by using anoncrypt (*`anon`*) and authcrypt (*`auth`*) from fig. 2 and our new combined mode* `a-auth` *from fig. 9. The result also follows trivially for the naive combination of anoncrypt and authcrypt. We proceed by fixing an arbitrary corrupt set* $\mathcal{C} \subseteq \{\mathsf{U}_1, \ldots, \mathsf{U}_n\}$ *and then providing a concrete simulator* $\boldsymbol{\sigma}$ *and a sequence of hybrids* $\mathsf{H}_0, \ldots, \mathsf{H}_3$ *such that, for any distinguisher* $\mathbf{D}$*,*

- $\mathsf{H}_0 \equiv \boldsymbol{\pi}^{\mathcal{H}}_{\mathsf{DIDComm}}[\mathsf{PKI}_n, \mathsf{A\text{-}INS}_{n,\ell}]$*;*
- $\mathsf{H}_0 \approx_{\varepsilon_1} \mathsf{H}_1$*, with* $\varepsilon_1(\mathbf{D}) := \mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KDF}}(\mathcal{A}_1)$*, for* $\mathcal{A}_1 := \rho_1(\mathbf{D})$*;*
- $\mathsf{H}_1 \approx_{\varepsilon_2} \mathsf{H}_2$*, with* $\varepsilon_2(\mathbf{D}) := \mathbf{Adv}^{\mathrm{dae}}_{\mathsf{DAE}}(\mathcal{A}_2)$*, for* $\mathcal{A}_2 := \rho_2(\mathbf{D})$*;*
- $\mathsf{H}_2 \approx_{\varepsilon_3} \mathsf{H}_3$*, with* $\varepsilon_3(\mathbf{D}) := \mathbf{Adv}^{\mathrm{ae\$}}_{\mathsf{AEAD}}(\mathcal{A}_3)$*, for* $\mathcal{A}_3 := \rho_3(\mathbf{D})$*;*
- $\mathsf{H}_3 \equiv \boldsymbol{\sigma}^{\mathcal{C}\cup\{\mathsf{E}\}}\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell,\mathcal{C}}$*;*

where $\varepsilon(\mathbf{D}) = \sum_{i=1}^{3} \varepsilon_i(\mathbf{D})$*, and reductions* $\rho_1(\mathbf{D})$ *to* $\rho_4(\mathbf{D})$ *are defined below.*

Simulator $\boldsymbol{\sigma}$ *We formally describe the simulator* $\boldsymbol{\sigma}$ *in fig. 18. It is attached to the* $\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell,\mathcal{C}}$ *resource, and it needs to simulate the behavior of the corrupted parties* $\mathcal{C}$*, as well as that of the adversary* E*. Note that the uniqueness of* M *(which ensures* $\mathsf{H}_3 \equiv \boldsymbol{\sigma}^{\mathcal{C}\cup\{\mathsf{E}\}}\mathsf{DIDComm}^{\mathsf{PKI}_n}_{n,\ell,\mathcal{C}}$*), is guaranteed by the committing property of the AEAD scheme.*

Hybrid H_0 *This hybrid, formally specified in fig. 14, is the monolithic representation of the real resource resource* $\boldsymbol{\pi}^{\mathcal{H}}_{\mathsf{DIDComm}}[\mathsf{PKI}_n, \mathsf{A\text{-}INS}_{n,\ell}]$*.*

Hybrid H_1 *This hybrid, formally specified in fig. 15, is the same as* H_0*, except that instead of using* KDF *to generate KEKs for* DAE*, we sample random keys for each message and sender-receiver pairs, and store them internally in a table* $\mathbf{K}$ *in order to correctly decrypt with* DAE*. Indistinguishability between* H_0 *and* H_1 *will follow from security of the KDF, which in our case implies DDH, as we explicitly formalized DIDComm's KDF building block from the DDH-based constructions in [Kra10].*

Hybrid H_2 *This hybrid, formally specified in fig. 16, is the same as* H_1*, except that instead of using* DAE *to encrypt CEKs, we sample random ciphertexts for each message and include them in the resulting ciphertext instead of the encryptions of the CEKs. We store the actual CEKs internally in table* $\mathbf{K}$*, so that we can retrieve these upon receiving messages without having to use* DAE*. Note that now the KEK are not needed anymore. Indistinguishability follows from security of the underlying DAE scheme.*

Hybrid H_3 *This hybrid, formally specified in fig. 17, is the same as* H_2*, except that instead of using* AEAD *to encrypt messages with the CEK, we sample random ciphertexts for each message and include them in the resulting ciphertext instead of the encryptions of the messages. We store the actual messages internally in table* $\mathbf{M}$*, so that we can retrieve these upon receiving without having to use* AEAD*. Indistinguishability follows from the security of the underlying AEAD scheme.*

Finally, the reductions ρ_1 *through* ρ_4 *are trivially defined as the code wrapping the differences between the various hybrids, since we used real-or-ideal notions of security for each primitive.*

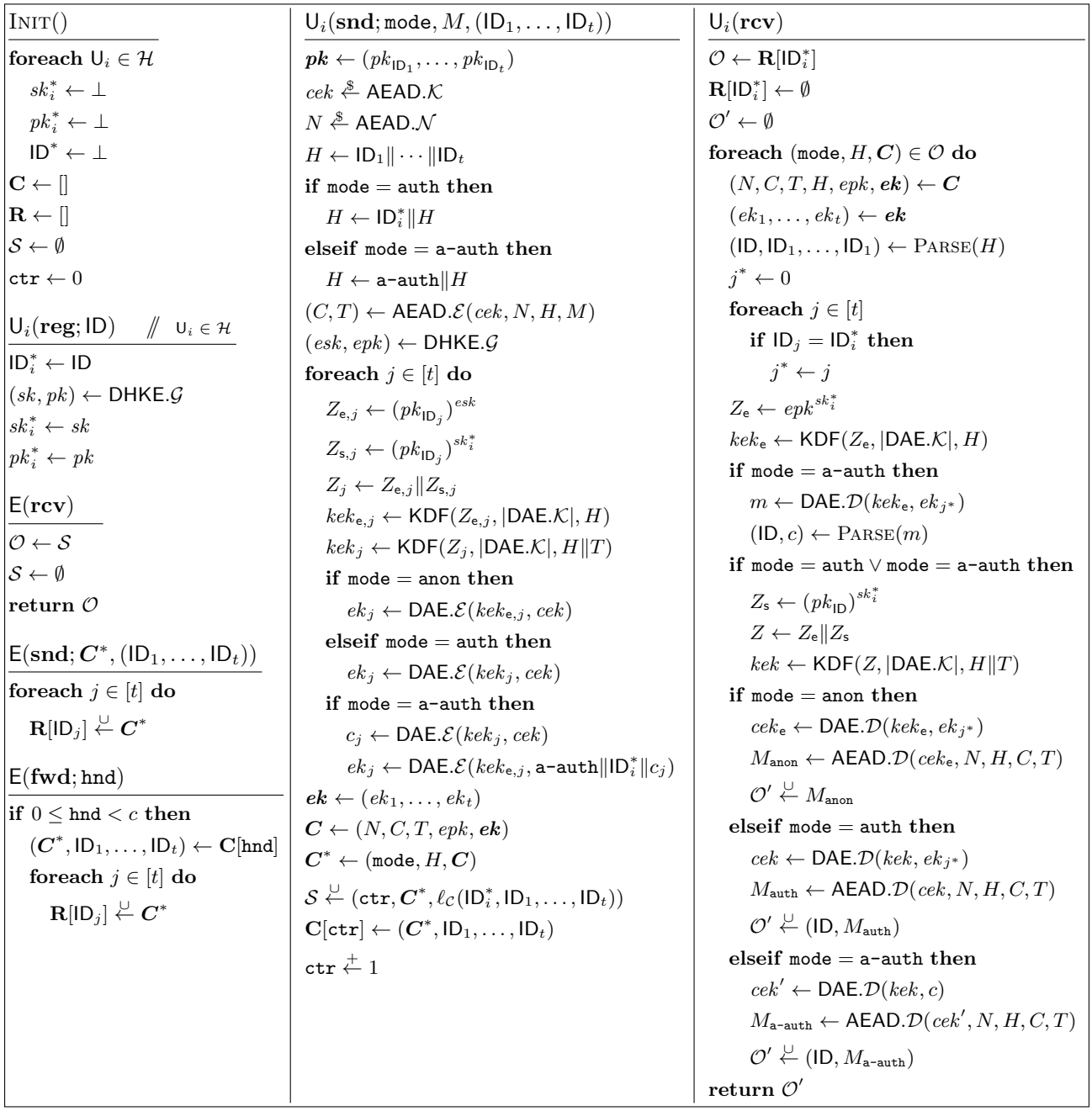

Figure 14: Hybrid H_0 for the proof of Theorem 1.

INIT()

foreach $\mathsf{U}_i \in \mathcal{H}$
  $sk_i^* \leftarrow \bot$
  $pk_i^* \leftarrow \bot$
  $\mathsf{ID}^* \leftarrow \bot$
$\mathbf{C} \leftarrow []$
$\mathbf{R} \leftarrow []$
$\mathbf{I} \leftarrow []$
$\mathbf{K}_\mathsf{e} \leftarrow []$
$\mathbf{K} \leftarrow []$
$\mathcal{S} \leftarrow \emptyset$
$\mathtt{ctr} \leftarrow 0$

$\mathsf{U}_i(\mathbf{reg}; \mathsf{ID})$ // $\mathsf{U}_i \in \mathcal{H}$

$\mathsf{ID}_i^* \leftarrow \mathsf{ID}$
$\mathbf{I}[\mathsf{ID}_i^*] \leftarrow i$
$(sk, pk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
$sk_i^* \leftarrow sk$
$pk_i^* \leftarrow pk$

$\mathsf{E}(\mathbf{rcv})$

$\mathcal{O} \leftarrow \mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$
return $\mathcal{O}$

$\mathsf{E}(\mathbf{snd}; \boldsymbol{C}^*, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

// Assuming $\boldsymbol{C}^*.H$ matches $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$
if $\exists \mathtt{hnd}, \boldsymbol{C}^{*\prime}, \mathsf{ID}'_1, \ldots, \mathsf{ID}'_t$:
    $\mathbf{C}[\mathtt{hnd}] = (\boldsymbol{C}^{*\prime}, \mathsf{ID}'_1, \ldots, \mathsf{ID}'_t)$
    $\wedge\, \boldsymbol{C}^{*\prime}.\boldsymbol{C} = \boldsymbol{C}^*.\boldsymbol{C}$ **then**
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{hnd}, \boldsymbol{C}^*)$
else
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (-1, \boldsymbol{C}^*)$

$\mathsf{E}(\mathbf{fwd}; \mathtt{hnd})$

if $0 \le \mathtt{hnd} < c$ **then**
  $(\boldsymbol{C}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t) \leftarrow \mathbf{C}[\mathtt{hnd}]$
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{hnd}, \boldsymbol{C}^*)$

$\mathsf{U}_i(\mathbf{snd}; \mathtt{mode}, M, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

$\boldsymbol{pk} \leftarrow (pk_{\mathsf{ID}_1}, \ldots, pk_{\mathsf{ID}_t})$
$cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$
$N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
$H \leftarrow \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$
if $\mathtt{mode} = \mathtt{auth}$ **then**
  $H \leftarrow \mathsf{ID}_i^* \| H$
elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
  $H \leftarrow \mathtt{a\text{-}auth} \| H$
$(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$
$(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
foreach $j \in [t]$ **do**
  if $\mathsf{U}_i \in \mathcal{H} \wedge \forall \iota \in [t] : \mathsf{U}_{\mathbf{I}[\mathsf{ID}_\iota]} \in \mathcal{H}$ **then**
    $kek_{\mathsf{e},j} \xleftarrow{\$} \{0,1\}^{|\mathsf{DAE}.\mathcal{K}|}$
    $\mathbf{K}_\mathsf{e}[\mathtt{ctr}, j] \leftarrow kek_{\mathsf{e},j}$
    $kek_j \xleftarrow{\$} \{0,1\}^{|\mathsf{DAE}.\mathcal{K}|}$
    $\mathbf{K}[\mathtt{ctr}, i, j] \leftarrow kek_j$
  else
    $Z_{\mathsf{e},j} \leftarrow (pk_{\mathsf{ID}_j})^{esk}$
    $Z_{\mathsf{s},j} \leftarrow (pk_{\mathsf{ID}_j})^{sk_i^*}$
    $Z_j \leftarrow Z_{\mathsf{e},j} \| Z_{\mathsf{s},j}$
    $kek_{\mathsf{e},j} \leftarrow \mathsf{KDF}(Z_{\mathsf{e},j}, |\mathsf{DAE}.\mathcal{K}|, H)$
    $kek_j \leftarrow \mathsf{KDF}(Z_j, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
  if $\mathtt{mode} = \mathtt{anon}$ **then**
    $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathsf{e},j}, cek)$
  elseif $\mathtt{mode} = \mathtt{auth}$ **then**
    $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
  if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
    $c_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
    $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathsf{e},j}, \mathtt{a\text{-}auth} \| \mathsf{ID}_i^* \| c_j)$
$\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
$\boldsymbol{C} \leftarrow (N, C, T, epk, \boldsymbol{ek})$
$\boldsymbol{C}^* \leftarrow (\mathtt{mode}, H, \boldsymbol{C})$
$\mathcal{S} \xleftarrow{\cup} (\mathtt{ctr}, \boldsymbol{C}^*, \ell_\mathcal{C}(\mathsf{ID}_i^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t))$
$\mathbf{C}[\mathtt{ctr}] \leftarrow (\boldsymbol{C}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$
$\mathtt{ctr} \xleftarrow{+} 1$

$\mathsf{U}_i(\mathbf{rcv})$

$\mathcal{O} \leftarrow \mathbf{R}[\mathsf{ID}_i^*]$
$\mathbf{R}[\mathsf{ID}_i^*] \leftarrow \emptyset$
$\mathcal{O}' \leftarrow \emptyset$
foreach $(\mathtt{hnd}, (\mathtt{mode}, H, \boldsymbol{C})) \in \mathcal{O}$ **do**
  $(N, C, T, H, epk, \boldsymbol{ek}) \leftarrow \boldsymbol{C}$
  $(ek_1, \ldots, ek_t) \leftarrow \boldsymbol{ek}$
  $(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_1) \leftarrow \textsc{Parse}(H)$
  $j^* \leftarrow 0$
  foreach $j \in [t]$
    if $\mathsf{ID}_j = \mathsf{ID}_i^*$ **then**
      $j^* \leftarrow j$
  if $\mathsf{U}_i \in \mathcal{H}$ **then**
    if $\mathtt{hnd} \neq -1$ **then** $kek_\mathsf{e} \leftarrow \mathbf{K}_\mathsf{e}[\mathtt{hnd}, i]$
    else $kek_\mathsf{e} \xleftarrow{\$} \{0,1\}^{|\mathsf{DAE}.\mathcal{K}|}$
  else
    $Z_\mathsf{e} \leftarrow epk^{sk_i^*}$
    $kek_\mathsf{e} \leftarrow \mathsf{KDF}(Z_\mathsf{e}, |\mathsf{DAE}.\mathcal{K}|, H)$
  if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
    $m \leftarrow \mathsf{DAE}.\mathcal{D}(kek_\mathsf{e}, ek_{j^*})$
    $(\mathsf{ID}, c) \leftarrow \textsc{Parse}(m)$
  if $\mathtt{mode} = \mathtt{auth} \vee \mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
    if $\mathsf{U}_i \in \mathcal{H}$ **then**
      $i' \leftarrow \mathbf{I}[\mathsf{ID}]$
      if $\mathtt{hnd} \neq -1$ **then** $kek \leftarrow \mathbf{K}[\mathtt{hnd}, i', i]$
      else $kek \xleftarrow{\$} \{0,1\}^{|\mathsf{DAE}.\mathcal{K}|}$
    else
      $Z_\mathsf{s} \leftarrow (pk_{\mathsf{ID}})^{sk_i^*}$
      $Z \leftarrow Z_\mathsf{e} \| Z_\mathsf{s}$
      $kek \leftarrow \mathsf{KDF}(Z, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
  if $\mathtt{mode} = \mathtt{anon}$ **then**
    $cek_\mathsf{e} \leftarrow \mathsf{DAE}.\mathcal{D}(kek_\mathsf{e}, ek_{j^*})$
    $M_\mathtt{anon} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek_\mathsf{e}, N, H, C, T)$
    $\mathcal{O}' \xleftarrow{\cup} M_\mathtt{anon}$
  elseif $\mathtt{mode} = \mathtt{auth}$ **then**
    $cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek, ek_{j^*})$
    $M_\mathtt{auth} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
    $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M_\mathtt{auth})$
  elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
    $cek' \leftarrow \mathsf{DAE}.\mathcal{D}(kek, c)$
    $M_\mathtt{a\text{-}auth} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek', N, H, C, T)$
    $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M_\mathtt{a\text{-}auth})$
return $\mathcal{O}'$

Figure 15: Hybrid H_1 for the proof of Theorem 1 with the changes from H_0 highlighted in gray.

$\textsc{Init}()$

foreach $\mathsf{U}_i \in \mathcal{H}$
  $sk_i^* \leftarrow \bot$
  $pk_i^* \leftarrow \bot$
  $\mathsf{ID}^* \leftarrow \bot$
$\mathbf{C} \leftarrow []$
$\mathbf{R} \leftarrow []$
$\mathbf{I} \leftarrow []$
$\mathbf{K} \leftarrow []$
$\mathbf{J} \leftarrow []$
$\mathcal{S} \leftarrow \emptyset$
$\mathtt{ctr} \leftarrow 0$

$\mathsf{U}_i(\mathbf{reg}; \mathsf{ID})$ // $\mathsf{U}_i \in \mathcal{H}$

$\mathsf{ID}_i^* \leftarrow \mathsf{ID}$
$\mathbf{I}[\mathsf{ID}_i^*] \leftarrow i$
$(sk, pk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
$sk_i^* \leftarrow sk$
$pk_i^* \leftarrow pk$

$\mathsf{E}(\mathbf{rcv})$

$\mathcal{O} \leftarrow \mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$
return $\mathcal{O}$

$\mathsf{E}(\mathbf{snd}; \boldsymbol{C}^*, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

// Assuming $\boldsymbol{C}^*.H$ matches $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$
if $\exists \mathtt{hnd}, \boldsymbol{C}^{*\prime}, \mathsf{ID}'_1, \ldots, \mathsf{ID}'_t$:
    $\mathbf{C}[\mathtt{hnd}] = (\boldsymbol{C}^{*\prime}, \mathsf{ID}'_1, \ldots, \mathsf{ID}'_t)$
    $\wedge\ \boldsymbol{C}^{*\prime}.\boldsymbol{C} = \boldsymbol{C}^*.\boldsymbol{C}$ **then**
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{hnd}, \boldsymbol{C}^*)$
else
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (-1, \boldsymbol{C}^*)$

$\mathsf{E}(\mathbf{fwd}; \mathtt{hnd})$

if $0 \le \mathtt{hnd} < c$ **then**
  $(\boldsymbol{C}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t) \leftarrow \mathbf{C}[\mathtt{hnd}]$
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{hnd}, \boldsymbol{C}^*)$

$\mathsf{U}_i(\mathbf{snd}; \mathtt{mode}, M, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

$\boldsymbol{pk} \leftarrow (pk_{\mathsf{ID}_1}, \ldots, pk_{\mathsf{ID}_t})$
$cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$
$\mathbf{K}[\mathtt{ctr}] \leftarrow cek$
$N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
$H \leftarrow \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$
if $\mathtt{mode} = \mathtt{auth}$ **then**
  $H \leftarrow \mathsf{ID}_i^* \| H$
elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
  $H \leftarrow \mathtt{a\text{-}auth} \| H$
$(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$
$(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
foreach $j \in [t]$ **do**
  if $\mathsf{U}_i \in \mathcal{H} \wedge \forall \iota \in [t] : \mathsf{U}_{\mathbf{I}[\mathsf{ID}_\iota]} \in \mathcal{H}$ **then**
    $ek_j \xleftarrow{\$} \mathsf{DAE}.\mathcal{C}$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $\mathbf{J}[\mathtt{ctr}, j] \leftarrow \mathsf{ID}_i^*$
  else
    $Z_{\mathtt{e},j} \leftarrow (pk_{\mathsf{ID}_j})^{esk}$
    $Z_{\mathtt{s},j} \leftarrow (pk_{\mathsf{ID}_j})^{sk_i^*}$
    $Z_j \leftarrow Z_{\mathtt{e},j} \| Z_{\mathtt{s},j}$
    $kek_{\mathtt{e},j} \leftarrow \mathsf{KDF}(Z_{\mathtt{e},j}, |\mathsf{DAE}.\mathcal{K}|, H)$
    $kek_j \leftarrow \mathsf{KDF}(Z_j, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
  if $\mathtt{mode} = \mathtt{anon}$ **then**
    $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathtt{e},j}, cek)$
  elseif $\mathtt{mode} = \mathtt{auth}$ **then**
    $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
  if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
    $c_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
    $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathtt{e},j}, \mathtt{a\text{-}auth} \| \mathsf{ID}_i^* \| c_j)$
$\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
$\boldsymbol{C} \leftarrow (N, C, T, epk, \boldsymbol{ek})$
$\boldsymbol{C}^* \leftarrow (\mathtt{mode}, H, \boldsymbol{C})$
$\mathcal{S} \xleftarrow{\cup} (\mathtt{ctr}, \boldsymbol{C}^*, \ell_{\mathcal{C}}(\mathsf{ID}_i^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t))$
$\mathbf{C}[\mathtt{ctr}] \leftarrow (\boldsymbol{C}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$
$\mathtt{ctr} \xleftarrow{+} 1$

$\mathsf{U}_i(\mathbf{rcv})$

$\mathcal{O} \leftarrow \mathbf{R}[\mathsf{ID}_i^*]$
$\mathbf{R}[\mathsf{ID}_i^*] \leftarrow \emptyset$
$\mathcal{O}' \leftarrow \emptyset$
foreach $(\mathtt{hnd}, (\mathtt{mode}, H, \boldsymbol{C})) \in \mathcal{O}$ **do**
  $(N, C, T, H, epk, \boldsymbol{ek}) \leftarrow \boldsymbol{C}$
  $(ek_1, \ldots, ek_t) \leftarrow \boldsymbol{ek}$
  $(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_1) \leftarrow \textsc{Parse}(H)$
  $j^* \leftarrow 0$
  foreach $j \in [t]$
    if $\mathsf{ID}_j = \mathsf{ID}_i^*$ **then**
      $j^* \leftarrow j$
  if $\mathsf{U}_i \in \mathcal{H}$ **then**
    if $\mathtt{hnd} \neq -1$ **then** $cek \leftarrow \mathbf{K}[\mathtt{hnd}]$
    else $cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$
    $M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $\mathsf{ID} \leftarrow \mathbf{J}[\mathtt{hnd}, j^*]$
    if $\mathtt{mode} = \mathtt{anon}$ **then**
      $\mathcal{O}' \xleftarrow{\cup} M$
    elseif $\mathtt{mode} = \mathtt{auth}$ **then**
      $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M)$
    elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M)$
  else
    $Z_{\mathtt{e}} \leftarrow epk^{sk_i^*}$
    $kek_{\mathtt{e}} \leftarrow \mathsf{KDF}(Z_{\mathtt{e}}, |\mathsf{DAE}.\mathcal{K}|, H)$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $m \leftarrow \mathsf{DAE}.\mathcal{D}(kek_{\mathtt{e}}, ek_{j^*})$
      $(\mathsf{ID}, c) \leftarrow \textsc{Parse}(m)$
      $Z_{\mathtt{s}} \leftarrow (pk_{\mathsf{ID}})^{sk_i^*}$
      $Z \leftarrow Z_{\mathtt{e}} \| Z_{\mathtt{s}}$
      $kek \leftarrow \mathsf{KDF}(Z, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
    if $\mathtt{mode} = \mathtt{anon}$ **then**
      $cek_{\mathtt{e}} \leftarrow \mathsf{DAE}.\mathcal{D}(kek_{\mathtt{e}}, ek_{j^*})$
      $M_{\mathtt{anon}} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek_{\mathtt{e}}, N, H, C, T)$
      $\mathcal{O}' \xleftarrow{\cup} M_{\mathtt{anon}}$
    elseif $\mathtt{mode} = \mathtt{auth}$ **then**
      $cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek, ek_{j^*})$
      $M_{\mathtt{auth}} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
      $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M_{\mathtt{auth}})$
    elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $cek' \leftarrow \mathsf{DAE}.\mathcal{D}(kek, c)$
      $M_{\mathtt{a\text{-}auth}} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek', N, H, C, T)$
      $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M_{\mathtt{a\text{-}auth}})$
return $\mathcal{O}'$

Figure 16: Hybrid H_2 for the proof of Theorem 1 with the changes from H_1 highlighted in gray.

$\textsc{Init}()$

foreach $\mathsf{U}_i \in \mathcal{H}$
  $sk_i^* \leftarrow \bot$
  $pk_i^* \leftarrow \bot$
  $\mathsf{ID}^* \leftarrow \bot$
$\mathbf{M} \leftarrow []$
$\mathbf{C} \leftarrow []$
$\mathbf{R} \leftarrow []$
$\mathbf{I} \leftarrow []$
$\mathcal{S} \leftarrow \emptyset$
$\mathtt{ctr} \leftarrow 0$

$\mathsf{U}_i(\mathbf{reg}; \mathsf{ID})$ // $\mathsf{U}_i \in \mathcal{H}$

$\mathsf{ID}_i^* \leftarrow \mathsf{ID}$
$\mathbf{I}[\mathsf{ID}_i^*] \leftarrow i$
$(sk, pk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
$sk_i^* \leftarrow sk$
$pk_i^* \leftarrow pk$

$\mathsf{E}(\mathbf{rcv})$

$\mathcal{O} \leftarrow \mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$
return $\mathcal{O}$

$\mathsf{E}(\mathbf{snd}; \boldsymbol{C}^*, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

// Assuming $\boldsymbol{C}^*.H$ matches $\mathsf{ID}_1, \ldots, \mathsf{ID}_t$
if $\exists \mathtt{hnd}, \boldsymbol{C}^{*\prime}, \mathsf{ID}'_1, \ldots, \mathsf{ID}'_t$:
  $\mathbf{C}[\mathtt{hnd}] = (\boldsymbol{C}^{*\prime}, \mathsf{ID}'_1, \ldots, \mathsf{ID}'_t)$
  $\wedge\ \boldsymbol{C}^{*\prime}.\boldsymbol{C} = \boldsymbol{C}^*.\boldsymbol{C}$ **then**
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{hnd}, \boldsymbol{C}^*)$
else
  foreach $j \in [t]$ **do**
    $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (-1, \boldsymbol{C}^*)$

$\mathsf{E}(\mathbf{fwd}; \mathtt{hnd})$

if $0 \le \mathtt{hnd} < c$ **then**
  if $\leftarrow \mathbf{C}[\mathtt{hnd}] \neq \bot$ **then**
    $(\boldsymbol{C}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t) \leftarrow \mathbf{C}[\mathtt{hnd}]$
    foreach $j \in [t]$ **do**
      $\mathbf{R}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{hnd}, \boldsymbol{C}^*)$
  else
    foreach $i \in [n]$ **do**
      if $\exists M : (\mathtt{hnd}, M) \in \mathbf{M}[\mathsf{ID}_i^*]$ **then**
        $\mathbf{R}[\mathsf{ID}_i^*] \xleftarrow{\cup} M$
      elseif $\exists \mathsf{ID}, M : (\mathtt{hnd}, \mathsf{ID}, M) \cdots$
          $\cdots \in \mathbf{M}[\mathsf{ID}_i^*]$ **then**
        $\mathbf{R}[\mathsf{ID}_i^*] \xleftarrow{\cup} (\mathsf{ID}, M)$

$\mathsf{U}_i(\mathbf{snd}; \mathtt{mode}, M, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

$\boldsymbol{pk} \leftarrow (pk_{\mathsf{ID}_1}, \ldots, pk_{\mathsf{ID}_t})$
$cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$
$N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
$H \leftarrow \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$
if $\mathtt{mode} = \mathtt{auth}$ **then**
  $H \leftarrow \mathsf{ID}_i^* \| H$
elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
  $H \leftarrow \mathtt{a\text{-}auth} \| H$
$(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
if $\mathsf{U}_i \in \mathcal{H} \wedge \forall \iota \in [t] : \mathsf{U}_{\mathbf{I}[\mathsf{ID}_\iota]} \in \mathcal{H}$ **then**
  $(C, T) \xleftarrow{\$} \{0,1\}^{|M|+\tau}$
  foreach $j \in [t]$ **do**
    $ek_j \xleftarrow{\$} \mathsf{DAE}.\mathcal{C}$
    if $\mathbf{I}[\mathsf{ID}_j] \neq \bot$ **then**
      if $\mathtt{mode} = \mathtt{anon}$ **then**
        $\mathbf{M}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{ctr}, M)$
      elseif $\mathtt{mode} = \mathtt{auth}$ **then**
        $\mathbf{M}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{ctr}, \mathsf{ID}_i^*, M)$
      elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
        $\mathbf{M}[\mathsf{ID}_j] \xleftarrow{\cup} (\mathtt{ctr}, \mathsf{ID}_i^*, M)$
else
  $(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$
  foreach $j \in [t]$ **do**
    $Z_{\mathtt{e},j} \leftarrow (pk_{\mathsf{ID}_j})^{esk}$
    $Z_{\mathtt{s},j} \leftarrow (pk_{\mathsf{ID}_j})^{sk_i^*}$
    $Z_j \leftarrow Z_{\mathtt{e},j} \| Z_{\mathtt{s},j}$
    $kek_{\mathtt{e},j} \leftarrow \mathsf{KDF}(Z_{\mathtt{e},j}, |\mathsf{DAE}.\mathcal{K}|, H)$
    $kek_j \leftarrow \mathsf{KDF}(Z_j, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
    if $\mathtt{mode} = \mathtt{anon}$ **then**
      $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathtt{e},j}, cek)$
    elseif $\mathtt{mode} = \mathtt{auth}$ **then**
      $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $c_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
      $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathtt{e},j}, \mathtt{a\text{-}auth} \| \mathsf{ID}_i^* \| c_j)$
$\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
$\boldsymbol{C} \leftarrow (N, C, T, epk, \boldsymbol{ek})$
$\boldsymbol{C}^* \leftarrow (\mathtt{mode}, H, \boldsymbol{C})$
$\mathcal{S} \xleftarrow{\cup} (\mathtt{ctr}, \boldsymbol{C}^*, \ell_{\mathcal{C}}(\mathsf{ID}_i^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t))$
if $\mathsf{U}_i \in \mathcal{H} \wedge \forall \iota \in [t] : \mathsf{U}_{\mathbf{I}[\mathsf{ID}_\iota]} \in \mathcal{H}$ **then**
  $\mathbf{C}[\mathtt{ctr}] \leftarrow \bot$
else
  $\mathbf{C}[\mathtt{ctr}] \leftarrow (\boldsymbol{C}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_t)$
$\mathtt{ctr} \xleftarrow{+} 1$

$\mathsf{U}_i(\mathbf{rcv})$

$\mathcal{O} \leftarrow \mathbf{R}[\mathsf{ID}_i^*]$
$\mathbf{R}[\mathsf{ID}_i^*] \leftarrow \emptyset$
$\mathcal{O}' \leftarrow \emptyset$
foreach $\boldsymbol{O} \in \mathcal{O}$ **do**
  if $\boldsymbol{O} = (\mathtt{hnd}, (\mathtt{mode}, H, \boldsymbol{C}))$ **then**
    $(N, C, T, H, epk, \boldsymbol{ek}) \leftarrow \boldsymbol{C}$
    $(ek_1, \ldots, ek_t) \leftarrow \boldsymbol{ek}$
    $(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_1) \leftarrow \textsc{Parse}(H)$
    $j^* \leftarrow 0$
    foreach $j \in [t]$
      if $\mathsf{ID}_j = \mathsf{ID}_i^*$ **then**
        $j^* \leftarrow j$
    $Z_{\mathtt{e}} \leftarrow epk^{sk_i^*}$
    $kek_{\mathtt{e}} \leftarrow \mathsf{KDF}(Z_{\mathtt{e}}, |\mathsf{DAE}.\mathcal{K}|, H)$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $m \leftarrow \mathsf{DAE}.\mathcal{D}(kek_{\mathtt{e}}, ek_{j^*})$
      $(\mathsf{ID}, c) \leftarrow \textsc{Parse}(m)$
      $Z_{\mathtt{s}} \leftarrow (pk_{\mathsf{ID}})^{sk_i^*}$
      $Z \leftarrow Z_{\mathtt{e}} \| Z_{\mathtt{s}}$
      $kek \leftarrow \mathsf{KDF}(Z, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
    if $\mathtt{mode} = \mathtt{anon}$ **then**
      $cek_{\mathtt{e}} \leftarrow \mathsf{DAE}.\mathcal{D}(kek_{\mathtt{e}}, ek_{j^*})$
      $M_{\mathtt{anon}} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek_{\mathtt{e}}, N, H, C, T)$
      $\mathcal{O}' \xleftarrow{\cup} M_{\mathtt{anon}}$
    elseif $\mathtt{mode} = \mathtt{auth}$ **then**
      $cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek, ek_{j^*})$
      $M_{\mathtt{auth}} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
      $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M_{\mathtt{auth}})$
    elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $cek' \leftarrow \mathsf{DAE}.\mathcal{D}(kek, c)$
      $M_{\mathtt{a\text{-}auth}} \leftarrow \mathsf{AEAD}.\mathcal{D}(cek', N, H, C, T)$
      $\mathcal{O}' \xleftarrow{\cup} (\mathsf{ID}, M_{\mathtt{a\text{-}auth}})$
  elseif $\boldsymbol{O} = M \wedge \boldsymbol{O} = (\mathsf{ID}, M)$ **then**
    $\mathcal{O}' \xleftarrow{\cup} \boldsymbol{O}$
return $\mathcal{O}'$

Figure 17: Hybrid H_3 for the proof of Theorem 1 with the changes from H_2 highlighted in gray.

$\textsc{Init}()$

$\mathbf{I}, \mathbf{J}, \mathbf{H} \leftarrow []$
$\mathcal{S} \leftarrow \emptyset$
$\mathtt{ctr} \leftarrow 0$

$\mathsf{A\text{-}INS}_{n,\ell}.\mathsf{U}_i(\mathbf{reg}; \mathsf{ID})$ // $i \in [n]$, $\mathsf{U}_i \in \mathcal{C}$

$\mathbf{I}[i] \leftarrow \mathsf{ID}$
$\mathbf{J}[\mathsf{ID}] \leftarrow i$
$\mathsf{DIDComm}_{n,\ell,\mathcal{C}}.\mathsf{U}_i(\mathbf{reg}; \mathsf{ID})$

$\mathsf{E}(\mathbf{reg}; i, \mathsf{ID})$ // Activated by $\mathsf{DIDComm}_{n,\ell,\mathcal{C}}$

$\mathsf{ID}_i^* \leftarrow \mathsf{ID}$
$\mathbf{I}[i] \leftarrow \mathsf{ID}$
$\mathbf{J}[\mathsf{ID}] \leftarrow i$
$(sk_i^*, pk_i^*) \leftarrow \mathcal{G}$
return pk_i^* // Sent back to $\mathsf{DIDComm}_{n,\ell,\mathcal{C}}$

$\mathsf{U}_i(\mathbf{snd}; M, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$ // $\mathsf{U}_i \in \mathcal{C}$

$\mathcal{S} \xleftarrow{\cup} (\mathtt{ctr}, M, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$
$\mathtt{ctr} \xleftarrow{+} 1$

$\mathsf{U}_i(\mathbf{rcv})$ // $i \in [n]$, $\mathsf{U}_i \in \mathcal{C}$

$\mathsf{ID} \leftarrow \mathbf{I}[i]$
$\mathcal{O} \leftarrow \mathbf{R}[\mathsf{ID}]$
$\mathbf{R}[\mathsf{ID}] \leftarrow \emptyset$
return $\mathcal{O}$

$\mathsf{E}(\mathbf{fwd}; \mathtt{hnd})$

$\mathsf{DIDComm}_{n,\ell,\mathcal{C}}.\mathsf{E}(\mathbf{fwd}; \mathtt{hnd})$

$\mathsf{E}(\mathbf{rcv})$

$\mathcal{O} \leftarrow \emptyset$
foreach $\boldsymbol{O} = (\mathtt{hnd}, \mathtt{mode}, l_\mathsf{m}, l_\mathsf{a}) \in \mathcal{S}$ **do**
  if $l_\mathsf{m} \in \mathbb{N}$ **then**
    $M \xleftarrow{\$} \{0,1\}^{l_\mathsf{m}}$
  if $l_\mathsf{a} \neq \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$ **then**
    $\mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t \xleftarrow{\$} \mathbf{ID}^t$
  $\boldsymbol{pk} \leftarrow (pk_{\mathsf{ID}_1}, \ldots, pk_{\mathsf{ID}_t})$
  $cek \xleftarrow{\$} \mathsf{AEAD}.\mathcal{K}$
  $N \xleftarrow{\$} \mathsf{AEAD}.\mathcal{N}$
  $H \leftarrow \mathsf{ID}_1 \| \cdots \| \mathsf{ID}_t$
  if $\mathtt{mode} = \mathtt{auth}$ **then**
    $H \leftarrow \mathsf{ID}_i^* \| H$
  elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
    $H \leftarrow \mathtt{a\text{-}auth} \| H$
  $(esk, epk) \leftarrow \mathsf{DHKE}.\mathcal{G}$
  $(C, T) \leftarrow \mathsf{AEAD}.\mathcal{E}(cek, N, H, M)$
  $\mathbf{H}[\mathtt{ctr}] \leftarrow (N, H, C, T)$
  foreach $j \in [t]$ **do**
    $Z_{\mathsf{e},j} \leftarrow (pk_{\mathsf{ID}_j})^{esk}$
    $Z_{\mathsf{s},j} \leftarrow (pk_{\mathsf{ID}_j})^{sk_i^*}$
    $Z_j \leftarrow Z_{\mathsf{e},j} \| Z_{\mathsf{s},j}$
    $kek_{\mathsf{e},j} \leftarrow \mathsf{KDF}(Z_{\mathsf{e},j}, |\mathsf{DAE}.\mathcal{K}|, H)$
    $kek_j \leftarrow \mathsf{KDF}(Z_j, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
    if $\mathtt{mode} = \mathtt{anon}$ **then**
      $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathsf{e},j}, cek)$
    elseif $\mathtt{mode} = \mathtt{auth}$ **then**
      $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $c_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_j, cek)$
      $ek_j \leftarrow \mathsf{DAE}.\mathcal{E}(kek_{\mathsf{e},j}, \mathtt{a\text{-}auth} \| \mathsf{ID}_i^* \| c_j)$
  $\boldsymbol{ek} \leftarrow (ek_1, \ldots, ek_t)$
  $\boldsymbol{C} \leftarrow (N, C, T, epk, \boldsymbol{ek})$
  $\boldsymbol{C}^* \leftarrow (\mathtt{mode}, H, \boldsymbol{C})$
  $\mathcal{O} \xleftarrow{\cup} \boldsymbol{C}^*$
$\mathcal{S} \leftarrow \emptyset$
return $\mathcal{O}$

$\mathsf{E}(\mathbf{snd}; \boldsymbol{C}, (\mathsf{ID}_1, \ldots, \mathsf{ID}_t))$

foreach $j \in [t]$ **do**
  if $\mathsf{U}_{\mathbf{J}[\mathsf{ID}_j]} \in \mathcal{C}$ **then**
    $\mathsf{DIDComm}_{n,\ell,\mathcal{C}}.\mathsf{E}(\mathbf{inj}; \boldsymbol{C})$
$\mathcal{R}, \mathcal{R}' \leftarrow \emptyset$
$(N, C, T, epk, \boldsymbol{ek}) \leftarrow \boldsymbol{C}$
$(\mathtt{mode}, H, \boldsymbol{C}) \leftarrow \boldsymbol{C}^*$
$\mathtt{hnd} \leftarrow \mathbf{H}[(N, H, C, T)]$
if $\mathtt{hnd} \neq \perp$
  $\mathsf{DIDComm}_{n,\ell,\mathcal{C}}.\mathsf{E}(\mathbf{fwd}; \mathtt{hnd})$
else
  $(N, C, T, H, epk, \boldsymbol{ek}) \leftarrow \boldsymbol{C}$
  $(ek_1, \ldots, ek_t) \leftarrow \boldsymbol{ek}$
  $(\mathsf{ID}, \mathsf{ID}_1, \ldots, \mathsf{ID}_1) \leftarrow \textsc{Parse}(H)$
  foreach $j \in [t]$
    $i \leftarrow \mathbf{J}[\mathsf{ID}]$
    $j^* \leftarrow \mathbf{J}[\mathsf{ID}_j]$
    $Z_\mathsf{e} \leftarrow epk^{sk_i^*}$
    $kek_\mathsf{e} \leftarrow \mathsf{KDF}(Z_\mathsf{e}, |\mathsf{DAE}.\mathcal{K}|, H)$
    if $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $m \leftarrow \mathsf{DAE}.\mathcal{D}(kek_\mathsf{e}, ek_{j^*})$
      $(\mathsf{ID}, c) \leftarrow \textsc{Parse}(m)$
      $Z_\mathsf{s} \leftarrow (pk_{\mathsf{ID}})^{sk_i^*}$
      $Z \leftarrow Z_\mathsf{e} \| Z_\mathsf{s}$
      $kek \leftarrow \mathsf{KDF}(Z, |\mathsf{DAE}.\mathcal{K}|, H \| T)$
    if $\mathtt{mode} = \mathtt{anon}$ **then**
      $cek_\mathsf{e} \leftarrow \mathsf{DAE}.\mathcal{D}(kek_\mathsf{e}, ek_{j^*})$
      $M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek_\mathsf{e}, N, H, C, T)$
    elseif $\mathtt{mode} = \mathtt{auth}$ **then**
      $cek \leftarrow \mathsf{DAE}.\mathcal{D}(kek, ek_{j^*})$
      $M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek, N, H, C, T)$
    elseif $\mathtt{mode} = \mathtt{a\text{-}auth}$ **then**
      $cek' \leftarrow \mathsf{DAE}.\mathcal{D}(kek, c)$
      $M \leftarrow \mathsf{AEAD}.\mathcal{D}(cek', N, H, C, T)$
    if $M = \perp$ **then**
      $\mathcal{R}' \xleftarrow{\cup} \mathsf{ID}_j$
    else
      $\mathcal{R} \xleftarrow{\cup} \mathsf{ID}_j$
$\mathsf{DIDComm}_{n,\ell,\mathcal{C}}.\mathsf{E}(\mathbf{snd}; \mathsf{ID}, M, \mathcal{R}, \mathcal{R}')$

Figure 18: Simulator $\boldsymbol{\sigma}$ for the proof of Theorem 1.

Proof 2 (Proof (sketch) of Theorem 3) *Just as we did in the proof of Theorem 1, we can define the first hybrid resource H_0 by replacing the code $(H, \boldsymbol{C}) \leftarrow \ldots$ calling the cryptographic primitives in each of the cases by the actual implementations as defined in Figure 2, and additionally Figures 10 and 12, for Theorem 3. We can then "flatten" the code, as we did in the proof of Theorem 1, that is, reuse variables that are defined in the same way across different code paths. For Theorem 1, this would mean that most of the hybrid H_0 from Theorem 1 can be recycled, and only t more calls to* $\mathsf{AEAD}.\mathcal{E}$, $\mathsf{AEAD}.\mathcal{D}$, *and* KDF *should be added. The next hybrids, and the resulting reductions, would then be straightforward, since the same set of cryptographic primitives is used.*