On End-to-End Encryption

Britta Hale and Chelsea Komlo

Abstract

End-to-end encryption (E2EE) is both vitally important to security and privacy online, yet currently under-defined. In this note, we map intuitive notions of end-to-end encryption to existing notions of encryption. In particular, we introduce the notion of *endness* as an notion which end-to-end systems must achieve in addition to traditional security notions associated with encryption, and provide formalizations to capture practical requirements. We demonstrate how the notion of encryption plus endness relates to a variety of case studies that either meet normative security understanding of E2EE or are considered normative failures. Finally, we extend these observations to authentication, and real-world authenticated channel use variants, including authenticated encryption with associated data and message franking.

1 Introduction

End-to-end encryption (E2EE) is a common mechanism employed to ensure the security of user-to-user communication protocols such as in secure messaging. Standard definitions exist for the types of underlying security properties often expected of E2EE architectures, such as confidentiality, integrity, authenticity, and forward secrecy. Yet, while there is a general intuition of what E2EE does not describe, there is a lack of clarity in the broader notion and what constitutes E2EE is currently not well-defined.

In this note, we examine how existing notions of encryption map onto established expectations of end-to-end encrypted protocols. We choose authenticated encryption with associated data (AEAD) as the primitive under consideration due to widespread usage; however, it is an exemplar only and a more basic form of just encryption can be considered as well. The reasoning for juxtaposing authenticity and confidentiality guarantees will become clearer later.

We demonstrate that while the notion of an AEAD scheme captures the requirements of encryption and data authenticity, existing AEAD definitions *do not* sufficiently capture the notions associated with what constitutes a valid endpoint. Consequently, we take a step towards formalizing additional requirements for securely encrypting with respect to the "ends" in end-to-end encryption. We apply this framing to case studies of both proposed and existing secure communication protocols.

2 Preliminaries

Let λ represent the security parameter in unary representation. $x \leftarrow y$ denotes the assignment of the value of y to x, and $x \xleftarrow{\$} S$ denotes sampling an element x from the non-empty set S uniformly at random, and we write $x \xleftarrow{\$} A()$ to denote the variable x as the random output of $A()$ (likewise for random sampling).

We employ code-based games to define notions of security. $\mathsf{Exp}^{\mathsf{sec}}_{\Sigma,\mathcal{A}}$ denotes the experiment Exp associated with the security guarantee sec. The experiment is considered over inputs of some scheme Σ and an adversary $\mathcal{A}$ that is playing against Σ.

3 Encryption with Undefined Ends

The broader notion of symmetric encryption and its associated security guarantees is well understood in cryptography [3]. Beyond the basic core notion of confidentiality, a wide range of sub-variations and security notions exist. For example, encryption can be defined with respect to Indistinguishability under Chosen Plaintext (IND-CPA), Indistinguishability under Chosen Ciphertext (IND-CCA), and Authenticated with Associated Data (AEAD). Associated data can be included to ensure analysis is closer to realistic protocols operation, as protocols often expose metadata that leaks information about the message contents, such as time, data, sender, and recipient. This variance among encryption guarantees already points to a first hurdle in providing a precise definition for E2EE.

In what follows, we build upon the notion of AEAD, but any other encryption notion could similarly be employed. We now present the definition of AEAD and its security notion for reference later, building upon prior definitions in the literature [22, 20, 12, 4].

Definition 3.1. *An* **AEAD** *scheme is a tuple of algorithms* $\Sigma = (KeyGen, Encrypt, Decrypt)$*, defined as follows:*

- $KeyGen(\lambda; r) \rightarrow k$*: A deterministic algorithm that accepts a security parameter* λ *and optional random coins* $r \in R$ *from a sample space* R*. Outputs a secret key* k *from the keyspace* K*.*
- $Encrypt(k, ad, m) \xrightarrow{\$} c$*: A potentially probabilistic algorithm that accepts as input a key* $k \in K$*, associated data* $ad \in AD$*, and a message* $m \in M$*. It outputs a ciphertext* $c \in C$*.*
- $Decrypt(k, ad, c) \rightarrow \perp / m$*: A deterministic algorithm that accepts as input a key* $k \in K$*, associated data* $ad \in AD$*, and a ciphertext* $c \in C$*, and outputs either the plaintext message* $m \in M$ *or a failure message* $\perp$*.*

An AEAD scheme Σ is correct if for every $k \in K$, $ad \in AD$ and $m \in M$,

$$\Pr[\ Decrypt(k, ad, Encrypt(k, ad, m)) = m\ ] = 1\ .$$

Security. While stronger notions exist [22, 4, 20], we say that an AEAD scheme is secure following a simplified experiment that does not account for nonces, statefulness, or a hierarchy of authentication guarantees. The security experiment can be seen in Figure 1

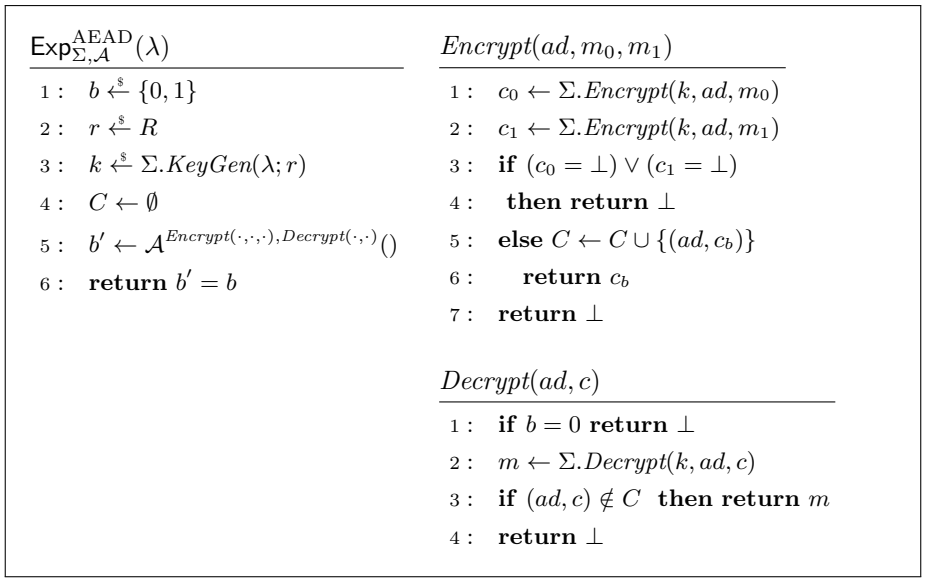

Figure 1: Authenticated Encryption with Associated Data experiment that defines the advantage of an adversary $\mathcal{A}$ against an AEAD scheme $\Sigma = (KeyGen, Encrypt, Decrypt)$ with respect to a security parameter λ. The goal to the adversary is to distinguish a challenge ciphertext that is the encryption of one out of two messages chosen by the adversary, with the additional adversarial capability to decrypt, where contextual but unencrypted data *ad* is also provided.

An AEAD scheme Σ is *secure* if the following advantage function is negligible

$$\mathsf{Adv}^{\mathrm{aead}}_{\Sigma,\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathrm{AEAD}}_{\Sigma,\mathcal{A},b=1}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathrm{AEAD}}_{\Sigma,\mathcal{A},b=0}(\lambda) = 1]\right| .$$

Encryption with Undefined Ends. Per basic terminology, an AEAD scheme should be more than sufficient to completely define what constitutes end-to-end encryption. Namely, an AEAD guarantees confidentiality such that a third party cannot read the data, and provides further benefits of data authenticity. Furthermore it provides authenticity for associated data, i.e, data that might be sent in the clear but which also require strong security guarantees, such as message headers. A system that is deemed to be end-to-end secure nominally appears to indicate that encryption alone is sufficient, and so an argument under

current definitions could even be made the AEAD alone yields E2EE. However, subtleties arise with this framing.

Among the more implicit assumptions of E2EE is the authenticity of data. This guarantee applies not only to the manipulation protection of data from its source, but the addition and replacement of data as well. Suppose that an attacker, by system design, was allowed to replace any user's message with one of its own choosing. If the receiver processes that message under the assumption that their intended communication partner sent it then we have an intuitive break in the notion of "end-to-end". The first step in patching such a gap is to add data authenticity to the confidentiality implied by E2EE – hence AEAD. However, the second step is more subtle: how can that the receiver be assured that the message really came from the "end" in question?

Ends. While an AEAD provides sufficient framing to define the composition of the confidentiality, integrity, and authenticity security properties often required for the primitive mechanism in E2EE, AEAD lacks one critical notion – that of how to define *endpoints*. In absence of well-defined "ends", authentication could be called into question, let alone confidentiality. For example, if an adversary can, by system design, modify or read information between the point of user data entry and the point at which a secure channel starts, then this would intuitively break E2EE in the commonly accepted sense.

In short, the issue in applying AEAD as a definitional primitive for E2EE is that it simply assumes that the higher-level application has already negotiated the correct endpoint, which in possession of the encryption/decryption key. This can be seen in Figure 1, where the security experiment assumes that keys have already been established between correct "ends" of the channel, and that the adversary is restricted to winning against the channel environment. In other words, once a suitable endpoint mechanism has been defined and entity authentication is enforced, there the E2EE security is reduced to the AEAD security of the channel. Thus, there is the natural component-wise characterization of E2EE as exemplified in the name, i.e., "Encryption + Ends".

Given this consideration, formalization for endpoints is needed to coherently model end-to-end encryption. We take a closer look now at heuristic cases that lack E2EE per well-accepted norms, and illustrate how they can be characterized by a poor definition of ends.

3.1 Heuristic Failures

We now take a closer look at several examples and examine how these relate to intuitive notions of end-to-end encryption. Notably, these examples do not meet intuitive notions of E2EE, despite not violating any security requirements of encryption schemes (e.g., AEAD from Figure 1).

Example 3.1. ***GCHQ Ghost User Proposal.*** *The Government Communications Headquarters (GCHQ) Ghost User proposal [15] is as follows: upon*

request by law enforcement, service providers will silently add a law enforcement participant to the requested conversation, without *notifying the existing participants that an additional user has been added. The added "ghost user" will be undetectable by the existing members of the conversation, but will silently have read access to the content of the messages, similarly to a real user*[1]*. As such, the protocol does not violate the underlying cryptographic encryption primitive, but instead simply requires suppression of any notification at the application layer that an additional user has been added to the conversation. While the communicating parties are aware of each other as "ends" to the channel, they are unaware of the ghost user "end".*

The GCHQ Ghost User proposal does not match widely held intuitive notions of end-to-end encryption [7, 23, 13]. We will later formalize this intuition, but simply note here that at the core of the violation is a matter of system transparency and user awareness. The GCHQ ghost user proposal requires some components of the system to remain secret, namely the logic of when a ghost user is added to a conversation, violating Kerckhoffs's principal. Further, users do not explicitly agree to reveal their communication history to the ghost user, which represents a potentially unbounded set of unknown identities.

Example 3.2. ***TLS and Zoom.*** *Before Zoom introduced its current protocol for end-to-end encryption [2], Zoom server's maintained cryptographic keys used to encrypt communication between users [16]. While the company claimed at that time to offer E2EE, Zoom later clarified that there was in fact a discrepancy in their use of the term "end-to-end" [8]. In short, Zoom's use of end-to-end encryption simply implied that encryption was performed between two endpoints of any type, but did not clarify* which *endpoints. Because client communication was encrypted to a "Zoom connector" (a Zoom client operating in the Zoom cloud), encryption was performed between users and servers in Zoom's cloud using either TLS for TCP connections or AES (with a key negotiated over TLS) for UDP connections [14]. In short, encryption was* not *between Zoom users directly. Hence, the identity that the sender encrypted their messages to (the public key of Zoom's servers), did not match the identity of the* intended *receiver of those messages (another Zoom user).*

When considering an AEAD in isolation, Zoom's prior claims of end-to-end encryption appear to hold. In fact, without clarification of endpoints, the concept of encrypting between any two parties aligns to the security experiment for AEAD, or indeed that of IND-CPA and IND-CCA – all experiments that model encryption and decryption between two generic, unspecified ends. However, intuitively, Zoom's claims of E2EE in this context were false – even though a user's communication was encrypted, a third-party entity (Zoom's servers) other than the end-user *conversation participants themselves could access user-level communication. In other words, there exists a discrepancy in the* equivalency *of which entities were allowed access. An end-user would naturally assume that none other than the other end-user(s) at the application-layer would be able to*

[1] Note nothing prevents the ghost user from posting messages, but such an action would of course alert all other users in the group to their presence.

access the data, under E2EE of such an application. There is thus an equivalency set for approved (or assumed) application-layer users, that Zoom servers would not naturally be within.

Example 3.3. ***Auto-Forwarding Plaintext Without User Consent.*** *As one example of a system that performs auto-forwarding of plaintext without user consent, we examine a proposed mechanism by Apple to detect Child Sexual Abuse Material (CSAM) in photos uploaded to iCloud in a privacy-preserving manner. The specific system designed by Apple employs blinded hashes of images and Private Set Intersection (PSI) to determine if an image matched against a centralized database [1].*

When viewed with the lens of end-to-end encryption as a closed loop (in other words, when the sender and receiver are the same entity), Apple's CSAM detection mechanism raised similar concerns as the GCHQ ghost user proposal. Although iCloud was not claimed to be an E2EE system, the auto-forwarding feature prompted many to quickly point out that this change – if used in the context of an E2EE system – breaks a normative notion of E2EE [6, 18]. This argument is due to the fact that an entity other than the sender or receiver learns the contents of a user's encrypted data if it matches content within a centralized database, without the knowledge or explicit approval of the sender or receiver. As such, this intuitive violation is important to consider when defining the expected characteristics of E2EE.

At the core of these examples and the controversies surrounding them is a clarity of definition for E2EE. Each use case could describe a validly functioning system in its own right. However, they incited debate because the variants broke some normative expectation of what E2EE entails, despite E2EE claims or simply expectation thereof. For some of these, there is a compounding factor in the concern for normalizing such non-E2EE system functionality even where reasonable user expectations would assume it. To this end, we take a closer look at modeling E2EE and requisite aspects of clarity for a system making such claims.

4 Modeling Endness

We now take a step towards formalizing the notion of appropriate ends in end-to-end encrypted protocols and systems, which is beyond the standard notion of a plain encryption scheme. Unlike standard notions of encryption, we require that protocols and systems claiming "endness' must be *explicit* about what endpoints are within any particular application, and how users can verify that they are in fact engaging with the appropriate endpoint.

Towards this end, we now define a set of three requirements that must be met for a system to be classified as end-to-end encryption, where *scheme security* may entail encryption.

1. **Encryption.** The standard notion of encryption and a specified corresponding security guarantee is preserved, e.g., as in Section 3. Encryption

goals for a system may extend to AEAD security, IND-CCA, IND-CPA, etc. As the strength of the E2EE security of the system varies over such a selection, an explicit specification is required.

2. **Equivalence Relation.** Given the space of all communicating identities $\mathcal{E}$, the system has an explicitly defined "equivalence relation", in terms of network topology. Namely, the equivalency relation and $\mathcal{E}$ form a setoid. The equivalence relation is defined relative to a system, versus as a mathematical relation. Generally, an equivalence relation will apply to "identities" within the same network layer, e.g. application-layer user identities, or link-layer identities.

3. **Endness.** A set of end identities $E \subseteq \mathcal{E}$ must be explicitly committed to by each participant and can be verified with respect to some pre-defined verification process (e.g., root of trust or out-of-band authentication). We formalize this requirement in Figure 2.

We now give more intuition for the second and third requirements.

Equivalence Relations The use of equivalence relation is to avoid ambiguity; while some definitions of E2EE focus explicitly on application-layer communication, others argue that transport layer protocols can also offer E2EE [19, 11]. We do not explicitly enforce the topology layer in use, but rather require that it is *specified*. Furthermore, since endpoint identities are implementation-specific, it is a requirement for systems claiming end-to-endness to explicitly define any irregular assumptions.

As an example, an equivalence relation might be related to user profiles within a messaging application. Another example would be identities associated lower in the stack. If an identity is at the link layer, e.g., a proxy at a router, then all other identities must be likewise (e.g. a "user" is represented by all proxies under their control having similar data access). In that case, E2EE is only assumed router-to-router, and any higher layer components may have access to data that the sender did not intend. Thus, the relative quality of E2EE is not only predicated on the encryption type employed but also the distance between that end and the user, in terms of network layers.

Endness We consider a break in end-to-endness if there is an identity among the communication partners of a user the that is not *explicitly* committed to. Namely, a sender i commits to a set of identities E and, if a receiver is not part of that set then there is a break in endness. For example, if Alice sends a message to Bob's mobile phone and the system automatically forwards that to another user Carol, without either Alice explicitly committing to Carol as a trusted identity. This constitutes a break in end-to-endness from Alice's view. By further restricting the equivalence relation defined above to parties able to read messages sent by an identity i, we can observe an equivalence class $[i]$ and say that endness is broken if $E \neq [i]$.

Equivalence Relations, Encryption, and Endness in Practice We now demonstrate how our notion of end-to-end encryption captures the heuristic failures reviewed in Section 3.1.

Example 4.1. ***GCHQ Ghost User.*** *The GCHQ Ghost User case is a break of endness, in that messages are forwarded to an identity in an broader identity set E' which is distinct from the identity set E committed to by the user. The user commits to a set of total users E that have access to the conversation, whereas the actual access set is $E' = E \cup \{GhostUser\}$.*

Example 4.2. ***Zoom and TLS.*** *The earlier case of Zoom demonstrates a break in both the committed identity set as well as a mismatch in the equivalence relation. 'Higher layer' Zoom identities specifically imply Zoom clients from the user view, given the video chat room functionality of Zoom, and thus a claim of E2EE was interpreted as client-to-client. Thus, when the Zoom server acted as an additional entity in the conversation there was a break in endness ($E' = E \cup \{ZoomServer\}$) from the view of users. If, on the other hand, Zoom had claimed merely transport layer security and specified the equivalence relation between the set of ends including the a specific server identity (or identities), then there would arguably be less contention on misinformation regarding that design's security.*

Separately, the issue of specific server identities also links to the question of access in terms of server location. For example, Zoom has provided some control over the specific data center regions used for data in transit [10]. Thus, while the servers were not specified as "ends" in terms of data access, it does hint at the transparency possible in identifying, e.g., a ZoomServer.

Example 4.3. ***Auto-Forwarding Plaintext Without User Consent.*** *In the auto-forwarding case we again see a break in endness, that the system automatically provides the content to a identity outside of the committed user set, E. If the user, A, is assured by their service provider that they have sole access to their data (i.e., the user commits to $E = \{A\}$ at time of encryption) and the data is auto-forwarded, either in plaintext or with keys provided to the third party, then there is an immediate break in the set of ends. In this case we see a clear distinction between an end user action of copying data and sending to another party versus a system auto-forwarding by design; namely that it is the set of committed "ends" that have choice and final determination over whether forwarding may take place or not. If an end identity is trusted, then there is a trust transitivity to their decision to relay information, whereas the broader system is not trusted with the information by default (hence why a system claims to provide E2EE in the first place).*

One could ask whether this auto-forwarding case could be considered in the form of a service provider advertising E2EE with end set $E = \{User, ServiceProvider\}$, and therefore satisfying our definition of end-to-endness. The answer is no. *A service provider is not a single entity in the sense of a user identity, but rather a conceptual representation of what could potentially be an unbounded set. Information access under the identity ServiceProvider is provided to an unspecified set of other system administrators, i.e., "ends", whose*

identities the original user did not commit to. Thus, it is not specified who exactly has access to the information, as ServiceProvider in this case is actually representative of a set of identities unknown to the user. The case of Zoom-Server data center regions for data in transit, noted at the end of Example 4.1, illustrates some degree of clarification on a ServiceProvider identity (i.e., a potential minimum transparency on the ServiceProvider "end" location).

Example 4.4. ***Naive End Sets.*** *Consider a naive end set example, where an application claiming E2EE based solely on IPSec simply adds every router identity on the wire to E. This does not provide a loophole in our definition, since the final ends of the chain of routers have not committed to the full set. Indeed, commitment is pairwise between routers, so each channel only commits to E such that $|E| = 2$. The system is designed to auto-forward the data from one router to the next, outside of the commitment set.*

4.1 Encryption Plus Endness

By now the reader may recognize that identity "ends" are normally specified during a protocol run, for example that two sides present certificates attesting to the owners thereof, and that this mutual authentication then feeds into the trust of the channel encryption. Thus, while part of E2EE is in the wider system specification and not within the bounds of cryptographic modeling, we can capture a portion of it.

Thus, we now present a modification to the notion of AEAD as follows, where ends (and the commitment to them) are modeled as additions to the core scheme. To capture the concept of the protocol link where ends integrate with the channel security, we simply model commitment to the end set and a commit verify functionality. This end-to-end adaption can also be applied to standard encryption, or other schemes.

We utilize the notion of a set of ends as $E = \{(i, pk_i)\}$, where $\{(i, pk_i)\}$ is a (claimed) set of identity, public key pairs, as well as the notion of a *commitment set* $\mathbf{c}$ to E. The commitment can then be verified at an appropriate time.

For example, E might be public keys presented during a TLS handshake, and $\mathbf{c}$ would correspond to certificates issued by trusted certificate authorities; for TLS, the respective receivers verify the commitment (certificate verify) during the handshake to confirm the validity of E before proceeding. Then only the identities in E should have access to the encrypted data. Another example includes the Signal protocol, where E represents the identity of users in the system and claimed public keys. In Signal, users apply the trust-on-first-use paradigm, so there is no immediate confirmation of the identity set. Instead, at a time of the user's choosing, the user will generate e.g., a numeric sequence or QR code $\mathbf{c}$ over the identities and public keys for comparison with the other user. Thus the verification of the commitment (e.g., comparison of identities and QR codes) occurs later than the handshake phase.

Definition 4.1. *An* **end-to-end AEAD scheme** Σ *is a tuple of algorithms* $\Sigma = (Commit, CommitVfy, KeyGen, Encrypt, Decrypt)$ *and end-set* $E = \{(i, pk_i)\}$

defined as follows:

- $Commit(E) \rightarrow \mathbf{c}$*: A possibly probabilistic algorithm. Accepts as input a set of identities* $E \subseteq \mathcal{E}$ *from the space of all identities* $\mathcal{E}$ *and outputs a commitment* $\mathbf{c}$ *that is an authentication commitment value for* E*.*
- $CommitVfy(E, \mathbf{c}) \rightarrow b$*: A deterministic algorithm. Accepts as input an identity set* $E \subseteq \mathcal{E}$ *and a commitment to an identity set* $\mathbf{c}$ *and outputs a bit* $b \in \{0, 1\}$*.*
- $KeyGen(\lambda, E; r) \rightarrow k$*: A deterministic algorithm that accepts a security parameter* λ*, an identity set* $E \in \mathcal{E}$*, and optional random coins* $r \in R$ *from a sample space* R
- $Encrypt(k, ad, m) \xrightarrow{\$} c$*: A potentially probabilistic algorithm that accepts as input a key* $k \in K$*, associated data* $ad \in AD$*, and a message* $m \in M$*. It outputs a ciphertext* $c \in C$*. and outputs a ciphertext* $c \in C$*.*
- $Decrypt(k, ad, c) \rightarrow \perp / m$*: A deterministic algorithm that accepts as input a key* $k \in K$*, associated data* $ad \in AD$*, and a ciphertext* $c \in C$*, and outputs either the plaintext message* $m \in M$ *or a failure message* $\perp$*.*

For correctness, in addition to the notion of correctness for an AEAD as in Definition 3.1, we require that

$$\Sigma.CommitVfy(E, \mathbf{c}) = 1 \text{ iff}$$
$$\mathbf{c} \leftarrow \Sigma.Commit(E) \ .$$

Remark 4.1. *Note that the correctness requirement implies not only that verification is correct for a committed set, but also that the set* E *input to* $CommitVfy$ *matches the same identity set input originally into* $Commit$*. The latter appears to be a natural and obvious implication, but it actually points to a system choice, not a cryptographic one. Namely, the local set* E *of assumed communication partners cannot be naively assigned per e.g., certificates shared in a protocol. The identities in* E *should be clearly specified, i.e., the identities that the system claims have access to the plaintext that is input into* $Encrypt$ *and output from* $Decrypt$*. This may imply application-layer clients, but is not restricted to them.*

As demonstrated in Figure 1, AEAD assumes that the key k that is input into $\Sigma.Encrypt$ and $\Sigma.Decrypt$ is authentic and known only by the correct parties. For security for an end-to-end AEAD scheme Σ, we remove this assumption, requiring system accountability on key access. We describe this security notion more formally in Figure 2 and say that a scheme is secure with respect to "end-to-endness" if this security experiment holds. For completeness, we present the modified AEAD notion in Appendix B, that runs on the modified key provided by this experiment.

Remark 4.2. *The Endness experiment shown in Figure 2 is intentionally composable with other security experiments of type* experiment*, allowing for adaptation to e.g., IND-CPA security.*

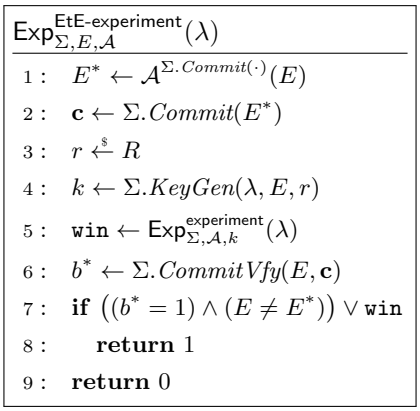

Figure 2: Endness Experiment

Definition 4.2. *We define the adversarial advantage of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{EtE\text{-}experiment}}_{\Sigma,E,\mathcal{A}}(\lambda)$ shown in Figure 2 as $\mathsf{Adv}^{\mathsf{EtE\text{-}experiment}}_{\Sigma,E,\mathcal{A}}(\lambda)$ and say that a scheme Σ satisfies* **end-to-end** *security under Σ if the following advantage function is negligible in λ:*

$$\mathsf{Adv}^{\mathsf{EtE\text{-}experiment}}_{\Sigma,E,\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{EtE\text{-}experiment}}_{\Sigma,E,\mathcal{A}}(\lambda) = 1]\right| \quad .$$

We now review several additional examples, to provide intuition about how this definition of end-to-endness is employed in practice.

Example 4.5. ***Signal Linked Devices with Per-User Identity Keys.*** *Signal's Sesame protocol [17] represents a user as a single* ***UserRecord****, which are indexed by a* ***UserID****. Each UserRecord contains a set of* ***DeviceRecords****, which are indexed by* ***DeviceID****. When Alice sends a message to Bob, Alice encrypts the message to* each *of Bob's devices represented in his UserRecord. Bob adds a new device by updating his UserRecord with an additional DeviceRecord.*

Informally, Signal's Sesame protocol is *end-to-end encrypted in our definition because the Signal application representing each user commits to a set of UserRecords containing the identities and devices for other known and trusted users. When framed in the context of Definition 4.1, the identity set E is composed of both Alice and Bob's UserRecords. $\Sigma.Commit$ generates a QR code with respect to this identity set. $\Sigma.CommitVfy$ is performed out of band by both Alice and Bob, who manually confirm if their QR codes match (with the expectation of the other's identity in E). $\Sigma.CommitVfy$ outputs 1 if Alice and Bob confirm the match, otherwise, it outputs 0.*

Example 4.6. ***Mutual TLS (mTLS).*** *While TLS is generally used in a setting where only the client is required to authenticate the server [21], Mutual TLS (mTLS) refers to a variant of the TLS protocol where both parties mutually authenticate to one another, by presenting and verifying each others' TLS certificates. Example 3.2 noted that an encrypted channel by itself does not constitute an E2EE system. However, a TLS connection between endpoints that*

are within the same equivalence class, as well as a clear commitment to and verification of the set of ends, could in fact constitute an E2EE system in our definition.

Employing the framing of Definition 4.1, E contains the set of public keys for all users that are allowed access. This might be two parties exchanging messages, or multiple parties. Performing $\Sigma.Commit$ requires all parties to interact with a certificate authority; the input is E (a set of public keys representing identities) and the output $\mathbf{c}$ is a set of certificates issued by that certificate authority with respect to those public keys. $\Sigma.CommitVfy$ accepts the set of public keys as E, in this case presented during the protocol handshake, and the set of certificates as $\mathbf{c}$, and verifies that E is valid with respect to $\mathbf{c}$. The ensuing TLS Record Layer protocol that would be assessed on e.g., AEAD security is then run off of the session key derived over E and other protocol randomness (i.e., capturing the TLS Handshake). Thus we see a natural framing of endness in terms of the participants acting during a protocol run for establishment of a secure channel.

Remark 4.3. *Note that Definition 4.1 explicitly relies on asymmetric cryptography and primarily precludes claimed E2EE protocols that rely on symmetric cryptography for authentication. If an identity is not paired with a public key then it is not possible for an end device to commit to the unique entity. One consequence of this requirement is that we capture protocols that employ message forwarding to identities not in E, as outside of the bounds we consider to be E2EE.*

Thus, a protocol using a key distribution center (KDC), such as in Kerberos [24], would not fall under our definition of E2EE. In the case of manual key distribution between users, e.g., if Alice and Bob manually transfer a symmetric key out-of-band, all three of the first functions – Commit, CommitVfy, KeyGen – can occur during that manual key transfer.

This may raise the question regarding distinction between these cases; however note that in the symmetric key distribution case Commit and CommitVfy cannot occur since it is not possible for identities to verify that the KDC has only provided the keying material to the intended partners.

5 E2EE vs. E2EA

In terminology, end-to-end encryption applies to the guarantee of endness for confidentiality *only*. However, many modern messaging applications extend to AEAD in order to ensure data authenticity as well. AEAD merges confidentiality and authenticity guarantees; yet our note on endness thus far has focused on endness for confidentiality only. This raises a question: how does end-to-end authentication (E2EA) relate to E2EE in practice?

The most natural correspondence between E2EE to E2EA is through use of AEAD as a dual paradigm in the sense of specific encryption modes such as AES-GCM or ChaCha20-Poly1305. Wickr, for example, applies the former [9]. However, authentication can also be achieved through message authenti-

cation codes (MACs), digital signatures, and authenticated encryption variants through a layering of primitives (e.g., encrypt-then-MAC or signcryption [27]).

In a comparison of endness for AEAD, MACs, and digital signatures, we make the following observations:

- In the definition of security of AEAD presented in Definition 1, key generation for both encryption and authenticity occurs simultaneously and therefore share the same 'ends'. This implies that endness is equally defined across the combination of confidentiality and authenticity. If E2EE is well-defined according to our notion, then E2EA is also.
- MACs share the symmetric property of AEAD, implying that auto-forwarding to other "ends" outside of some set is infeasible while maintaining the authenticity property. However, unlike AEAD, the end set for authenticity ($E_{\mathcal{A}}$) and the end set for confidentiality ($E_{\mathcal{C}}$) may or may not be identical, leading to potential irregular cases, e.g., $E_{\mathcal{C}} \subset E_{\mathcal{A}}$. In such a case, those ends that are able to verify authenticity of a message – and potentially prove attribution – could constitute a larger set than those able to read the message. In a case of auto-forwarding, this could imply that a third party can attribute a message to Alice after it is decrypted. Naturally, such attribution depends on algorithm layer ordering. For example, encrypt-and-MAC and MAC-then-encrypt would be more susceptible to this than encrypt-then-MAC for the case described.
- Digital signatures present an interesting case in secure messaging as they lack "ends" in verification. Any party in possession of the public-key can verify the signature, so it is not even possible to pre-specify the end set $E_{\mathcal{A}}$. For example, suppose that an application uses a sign-then-encrypt paradigm; even if the ends for encryption $E_{\mathcal{C}}$ are well-defined, there is no guarantee for the sender than only those ends in $E_{\mathcal{C}}$ can attribute the message. In fact, the signed message can be forwarded or shared, with corresponding attribution to the original sender. Thus $E_{\mathcal{A}}$, at time of commitment by the sender is such that $|E_{\mathcal{A}}| = \infty$. (Note that this applies to standard digital signatures – deniable authentication variants [25, 26] may have a different committing set.)

  Even though an application can temporarily restrict access to the public key infrastructure, such a temporary solution is by no means a long-term guarantee for endness. For example, the iMessage supports digital signatures on messages for authentication; while the public keys of any given user are not publicly listed, Alice has no guarantee about the saticity of the authentication receiver end set, or of new users gaining access to that attribution.

In some messaging applications, the above differentiation has been used for creative security goals and new primitives as we will look at next.

Example 5.1. ***Message Franking.*** *"Message Franking" [5] is a technique to provide cryptographic assurance to a voluntary user reporting of other users'*

private messages. Specifically, if Alice sends Bob a message, her message is confidential but also authenticated. If Bob wishes to report Alice's message to the service provider as offensive, he can do so in a way that reveals the message contents to the service provider in a way that demonstrates that the message was indeed sent by Alice. However, without Bob taking this step, the contents of Alice's conversation with Bob is otherwise unattainable by the service provider. Technically, a user reports another user's message to the service provider by by sending the plaintext message, along with a franking tag that is generated by the sender before the message is sent. Note that while message franking allows Bob to prove to the service provider that Alice sent a specific message to Bob, it preserves third-party deniability. The service provider can verify that the message was sent by Alice, but a third party cannot.

Message franking intuitively meets our definition of end-to-end encryption, because users voluntarily reveal specific messages sent to them to the service provider. Thus messages are not auto-forwarded but rather the action is explicitly chosen by an "end". Outside of such a choice, a service provider cannot otherwise learn the contents of an encrypted conversation or other messages that the user chooses not to reveal.

Employing the framing of Definition 4.1 in the context of Facebook message franking, performing $\Sigma.Commit$ commits a user to their identity public key associated with a particular device, which is then registered with the service provider. Performing $\Sigma.CommitVfy$ entails inspecting another user's identity public key and verifying (out of band) with that user that it is indeed authentic.

Message franking, however, utilizes a larger authentication end set $E_{\mathcal{A}}$, than confidentiality end set $E_{\mathcal{C}}$, namely $E_{\mathcal{C}} \subset E_{\mathcal{A}}$, since the ends that can verify authenticity of a message include more entities (namely the ServiceProvider) than those able to decrypt. Furthermore, as in Example 4.3 for auto-forwarding to an unspecified ServiceProvider, the exact end identities associated with $E_{\mathcal{A}}$ could be better specified. Hence we see message franking as a case of E2EE without well-defined E2EA.

6 Conclusion

As with all messaging applications, the features and trade-offs therein differ by use-case goals. Consequently, we neither commend nor condemn any one combination of goals. Rather we define an application achieving any variant of end-to-endness as one that is A) clear and B) honest in its 1) description of the ends-set to users and 2) potential changes in the end sets over time. We lay out three core components in specifying E2EE, to avoid ambiguity, assumptions, and potential user mistrust. In essence, such transparency can take the form of e.g., E2E–{IND-CPA}–'Application Clients'–$\{Alice, Bob\}$.

In a broader case, an application using message franking should precisely and clearly define $E_{\mathcal{C}}$ and $E_{\mathcal{A}}$ during normal operation as well as the precise changes to $E_{\mathcal{A}}$ if a message is 'franked'. Likewise, in cases where a plaintext message is digitally signed in a system purporting end-to-endness, then it should be stated

that the authentication end set $E_{\mathcal{A}}$ is unbounded, and not that verification and attribution is limited to the system.

Acknowledgments

We thank Mallory Knodel for motivating this note and her helpful discussion and review, and Chris Wood for his review.

References

[1] Apple. CSAM Detection - Technical Summary. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf. Accessed 2021-12-09.

[2] Josh Blum, Simon Booth, Brian Chen, Oded Gal1, Maxwell Krohn, Julia Len, Karan Lyons, Antonio Marcedone, Mike Maxim, Merry Ember Mou, Jack O'Connor, Surya Rien, Miles Steele, Matthew Green, Lea Kissner, and Alex Stamos. E2E Encryption for Zoom Meetings. https://github.com/zoom/zoom-e2e-whitepaper/blob/master/zoom_e2e.pdf. Accessed 2021-12-08.

[3] Dan Boneh and Victor Shoup. *A graduate course in applied cryptography (version 0.5)*. 2020. cryptobook.us.

[4] Colin Boyd, Britta Hale, Stig Frode Mjølsnes, and Douglas Stebila. From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS. In Kazue Sako, editor, *CT-RSA, San Francisco, CA, USA, February 29 - March 4, 2016*, volume 9610 of *LNCS*, pages 55–71. Springer, 2016.

[5] Facebook. Messanger Secret Conversations Technical Whitepaper. https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf. Accessed 2022-1-01.

[6] Sharon Bradford Franklin and Greg Nojeim. International Coalition Calls on Apple to Abandon Plan to Build Surveillance Capabilities into iPhones, iPads, and other Products. https://cdt.org/insights/international-coalition-calls-on-apple-to-abandon-plan-to-build-surveillance-capabilities-into-iphones-ipads-and-other-products/. Accessed 2022-04-05.

[7] Sharon Bradford Franklin and Andi Wilson Thompson. Open Letter to GCHQ on the Threats Posed by the Ghost Proposal. https://www.lawfareblog.com/open-letter-gchq-threats-posed-ghost-proposal. Accessed 2022-04-05.

[8] Oded Gal. The Facts Around Zoom and Encryption for Meetings/Webinars. https://blog.zoom.us/facts-around-zoom-encryption-for-meetings-webinars/. Accessed 2021-12-10.

[9] Chris Howell, Tom Leavy, and Joël Alwen. Wickr's Messaging Protocol. https://wickr.com/wickrs-messaging-protocol/, May 2018. Accessed 2021-04-07.

[10] Brendan Ittelson. Coming April 18: Control Your Zoom Data Routing. https://blog.zoom.us/coming-april-18-control-your-zoom-data-routing/, 2020. Zoom Blog. Accessed 2022-4-08.

[11] Mallory Knodel, Fred Baker, Olaf Kolkman, Sofia Celi, and Gurshabad Grover. Definition of End-to-end Encryption. https://datatracker.ietf.org/doc/draft-knodel-e2ee-definition/. Accessed 2022-04-05.

[12] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Santa Barbara, CA, USA, August 18-22, 2013*, volume 8042 of *LNCS*, pages 429–448. Springer, 2013.

[13] Susan Landau. Exceptional Access: The Devil is in the Details. https://www.lawfareblog.com/exceptional-access-devil-details3. Accessed 2022-04-05.

[14] Micah Lee and Yael Grauer. ZOOM Meetings Aren't End-to-End Encrypted, Despite Misleading Marketing. https://theintercept.com/2020/03/31/zoom-meeting-encryption/. Accessed 2022-4-07.

[15] Ian Levy and Crispin Robinson. Principles for a More Informed Exceptional Access Debate. https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate. Accessed 2021-12-01.

[16] Bill Marczak and John Scott-Railton. Move Fast and Roll Your Own Crypto: A Quick Look at the Confidentiality of Zoom Meetings. https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/. Accessed 2021-12-10.

[17] Moxie Marlinspike and Trevor Perrin. The Sesame Algorithm: Session Management for Asynchronous Message Encryption. https://signal.org/docs/specifications/sesame/sesame.pdf. Accessed 2021-12-05.

[18] India McKinney and Erica Portnoy. Apple's Plan to "Think Different" About Encryption Opens a Backdoor to Your Private Life. https://www.eff.org/deeplinks/2021/08/apples-plan-think-different-about-encryption-opens-backdoor-your-private-life. Accessed 2022-04-05.

[19] A. Muffett. A Duck Test for End-to-End Secure Messaging. https://datatracker.ietf.org/doc/html/draft-muffett-end-to-end-secure-messaging-02.txt. Accessed 2022-04-05.

[20] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011, Seoul, South Korea, December 4-8, 2011.*, volume 7073 of *LNCS*, pages 372–389. Springer, 2011.

[21] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. https://datatracker.ietf.org/doc/html/rfc8446. Accessed 2022-03-11.

[22] 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.

[23] Internet Society. Ghost Proposal Fact Sheet. https://www.internetsociety.org/wp-content/uploads/2020/03/Ghost-Protocol-Fact-Sheet.pdf. Accessed 2022-04-05.

[24] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter Conference. Dallas, Texas, USA, January 1988*, pages 191–202. USENIX Association, 1988.

[25] Nik Unger and Ian Goldberg. Deniable key exchanges for secure messaging. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1211–1223. ACM, 2015.

[26] Nik Unger and Ian Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *Proceedings of Privacy Enhancing Technology*, 2018(1):21–66, 2018.

[27] Yuliang Zheng. Digital signcryption or how to achieve cost(signature + encryption). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, page 165–179, Berlin, Heidelberg, 1997. Springer-Verlag.

A Other Encryption Experiments

Here we provide a more general encryption definition (not AEAD) along with typical IND-CPA and IND-CCA experiments for reference.

Definition A.1. *An* **encryption** *scheme is a tuple of algorithms* $\Sigma = (\mathit{KeyGen}, \mathit{Encrypt}, \mathit{Decrypt})$*, defined as follows:*

- $\mathit{KeyGen}(\lambda; r) \to k$*: A deterministic algorithm that accepts a security parameter* λ *and optional random coins* $r \in R$ *from a sample space* R*. Outputs a secret key* $k \xleftarrow{\$} K$ *from the keyspace* K*.*
- $\mathit{Encrypt}(k, m) \xrightarrow{\$} c$*: A probabilistic algorithm that accepts as input a key* $k \in K$ *and a message* $m \in M$*. It outputs a ciphertext* $c \in C$*.*
- $\mathit{Decrypt}(k, c) \to \perp / m$*: A deterministic algorithm that accepts as input a key* k *and a ciphertext* c*, and outputs either the plaintext message* $m \in M$ *or a failure message* $\perp$*.*

An encryption scheme Σ is correct if for every $k \in K$ and $m \in M$,

$$\Pr[\ \mathit{Decrypt}(k, \mathit{Encrypt}(k, m)) = m\ ] = 1\ .$$

$\mathsf{Exp}^{\mathrm{indcpa}}_{\Sigma,\mathcal{A}}(\lambda)$

1 : $b \xleftarrow{\$} \{0,1\}$
2 : $r \xleftarrow{\$} R$
3 : $k \xleftarrow{\$} \Sigma.\mathit{KeyGen}(\lambda; r)$
4 : $b' \leftarrow \mathcal{A}^{\mathit{Encrypt}(\cdot,\cdot)}()$
5 : **return** $b' = b$

$\mathit{Encrypt}(m_0, m_1)$

1 : $c_0 \leftarrow \Sigma.\mathit{Encrypt}(k, m_0)$
2 : $c_1 \leftarrow \Sigma.\mathit{Encrypt}(k, m_1)$
3 : **if** $(c_0 = \perp) \vee (c_1 = \perp)$
4 : **return** $\perp$
5 : **else return** c_b
6 : **return** $\perp$

Figure 3: Indistinguishability under Chosen-Plaintext Attack (IND-CPA) experiment that defines the advantage of an adversary $\mathcal{A}$ against an encryption scheme $\Sigma = (\mathit{KeyGen}, \mathit{Encrypt}, \mathit{Decrypt})$ with respect to a security parameter λ. The goal to the adversary is to distinguish a challenge ciphertext that is the encryption of one out of two messages chosen by the adversary.

An encryption scheme Σ is IND-CPA-secure if the following advantage function is negligible

$$\mathsf{Adv}^{\mathrm{indcpa}}_{\Sigma,\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathrm{indcpa}}_{\Sigma,\mathcal{A}}(\lambda) = 1] - 1/2\right|\ .$$

An encryption scheme Σ is IND-CCA-secure if the following advantage function is negligible

$$\mathsf{Adv}^{\mathrm{indcca}}_{\Sigma,\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathrm{indcca}}_{\Sigma,\mathcal{A}}(\lambda) = 1] - 1/2\right|\ .$$

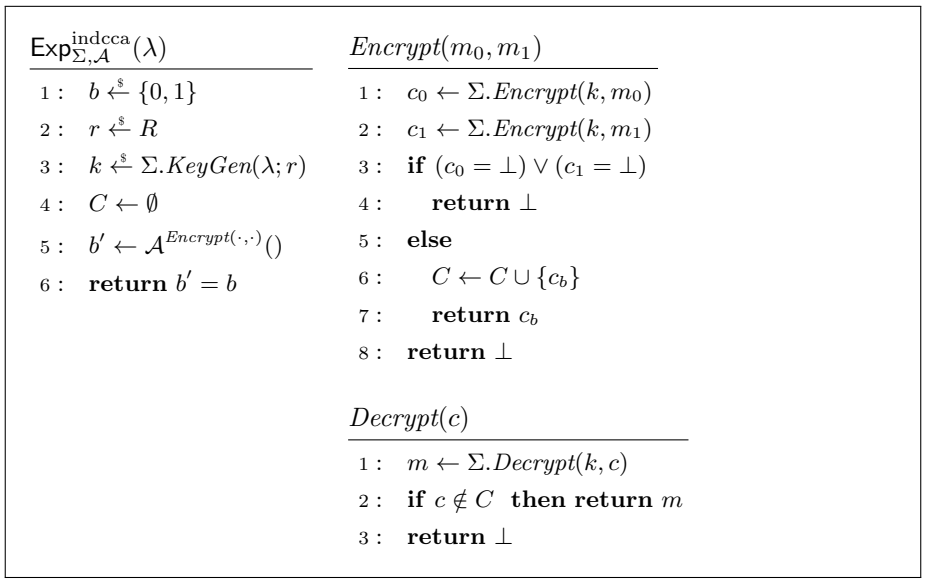

Figure 4: Indistinguishability under Chosen-Ciphertext Attack (IND-CCA) experiment that defines the advantage of an adversary $\mathcal{A}$ against an encryption scheme $\Sigma = (KeyGen, Encrypt, Decrypt)$ with respect to a security parameter λ. The goal to the adversary is to distinguish a challenge ciphertext that is the encryption of one out of two messages chosen by the adversary, with the additional adversarial capability to decrypt.

B End-to-End AEAD Experiment

For an end-to-end AEAD, we now show the modified AEAD experiment in Figure 5 with respect to the Endness security experiment in Figure 2, for completeness.

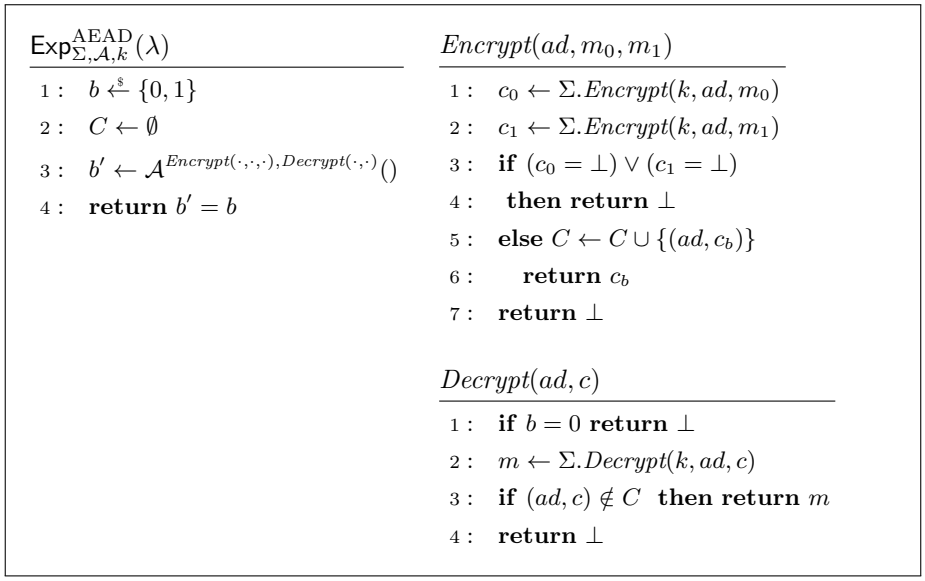

Figure 5: Modification to Authenticated Encryption with Associated Data experiment that defines the advantage of an adversary $\mathcal{A}$ against an AEAD scheme Σ with respect to a security parameter λ, while run together with the end-to-endness experiment in Figure 2. This modification is introduced by the endness experiment in Figure 2, which requires that an identity set E additionally be employed as a parameter into *KeyGen*. Hence, the key k is given as a parameter into the AEAD experiment.