EMV Key Agreement

C. Brzuska[1], N.P. Smart[2], B. Warinschi[2], and G.J. Watson[2]

[1] School of Computer Science, School of Engineering
Tel Aviv University, Israel.
[2] Dept. Computer Science,
University of Bristol, UK.

1 Introduction

The EMV chip-and-pin system is used to secure the majority of the world's credit card and ATM transactions; as well as securing electronic banking in many countries. The current system is based on RSA public-key cryptography, combined with DES based symmetric-key cryptography. In the EMV system bank or credit card customers are issued with a plastic card containing an embedded chip holding various cryptographic keys and which can perform various cryptographic operations. The card is used to communicate with a terminal (typically a point-of-sale terminal in a shop, but other terminals are possible). In addition the card can produce cryptograms for sending on to the banking system for processing. However, the cryptographic functionality provided by the card in its first generation incarnation is relatively limited.

As part of a major reworking of the chip-and-pin system the EMV consortium has decided to replace RSA with ECC based systems; and to also enable the card to provide a number of additional cryptographic functionalities. One of these functionalities is the ability to create a secure channel between the card and the terminal. The draft specification for the establishment of the secure channel was published in Nov 2012 [11] as a request for comments. It is to analyse this proposal that the current paper is focused. Since the current EMV system is so widespread (with between one billion and two billion EMV enabled cards in circulation), and protects a significant proportion of consumer banking transactions it is clearly important that any new protocol is given a thorough cryptographic analysis.

The problem of establishing and implementing secure channels is central to practical uses of cryptography and a superficial look at existing literature would let one believe that this is indeed a solved problem. What can be simpler than first running a secure key-exchange protocol and then use the resulting keys to somehow encrypt and authenticate the messages to be sent? Indeed, there are a plethora of works looking at key establishment [2, 3, 6] and a similar number of works looking at how to build secure channels on top of shared keys [1, 4, 13]. However, the traditional key agreement models such as those following the schema set out by Bellare and Rogaway [2] have been shown to be not so usefully applicable to deployed protocols. In particular the indistinguishability of keys notion introduced by Bellare and Rogaway is often broken by the usage of the agreed key to establish the channel or provide key confirmation in practice. Thus recent focus has shifted to analysing real world protocols, such as TLS, and building models which capture the combined property of key establishment and secure channel as a combined primitive, see [12, 8, 5].

The proposed EMV key establishment protocol also cannot be analysed using traditional techniques, and in addition the entire protocol is inherently one-sided in that only the card is authenticated, the terminal is never authenticated. Whilst such one-sided authentication is also inherent in most applications of TLS, only recently has there been a proper treatment of one-sided key establishment in the literature which is described in full detail [7]. The one-sided authentication case is often tagged onto a discussion of the two sided case, and not fully developed. Thus in this paper we also present a fully described model for one-sided authentication.

There are currently two approaches to studying the combined properties of a key establishment and secure channel protocol. The first approach is to prove security of the secure channel protocol assuming agreed keys, and then to prove that the key establishment protocol provides the correct properties one requires for the keys to be used in the secure channel protocol. This *modular approach* is explored in full in [5], where a game-based composition theorem is provided for combining key agreement protocols with other protocols using the previously agreed keys. The approach is shown to work for real world protocols such as TLS. We do not however follow the approach in [5] since it requires an explicit construction of a secure channel protocol; whereas the specification in [11] assumes the secure channel is given and is thus a "black box". The second approach is to treat the combined key establishment and secure channel protocol as a monolithic whole. This approach is typified by the work in Jager et al. [8] on the TLS protocol. We however note some problems with the model of Jager et al. (we ignore the concern on length hiding properties in the work of [8] as they are not relevant to our application); the most important of these issues is that the model allows the creation of paired partners which do not have matching conversations. In this paper we adopt the approach of [8], but we modify their model so as to rectify the problems which we identify.

In particular the main modification we make is to the definition of matching conversation. In the traditional definition of matching conversations, originally presented in [2], two parties are said to have had a matching conversation if their transcripts of exchanged messages (which could be ciphertexts) are in agreement. We relax this definition and insist that the conversations match on the messages received and transmitted before (or after) any encryption/decryption operation. Thus matching conversations is defined on the underlying plaintexts. In the case where no encryption occurs in a message flow, the two definitions coincide.

We present a new definitional framework to capture one-sided key agreement followed by composition with a secure channel. Our new framework is conceptually simpler than previous models. We then use this framework to that the key agreement and secure channel protocol proposed for EMV meets the desired security mode

In the Appendix we present a modification of the Jager et al. definition to deal with the issues we raise, and we show that our conceptually simpler framework is equivalent to the modified Jager et al definition.

The final contribution we make is to analyse the unlinkability properties of the proposed protocol from EMV. One of the design criteria of the protocol is a mild form of unlinkability; in particular an adversary on seeing a message flow between a terminal and a card should not be able to link this cards current transaction with a previous transaction from the same card. The protocol aims to ensure this by not transmitting the certificate in the clear, however the protocol also uses a performance optimization in that the card uses a small emphemeral private key. We quantify exactly what security one obtains in terms of unlinkability given the range from which the emphemeral public key is selected.

The rest of the paper is organized as follows: In Section 2 we present the proposed EMV key exchange protocol, and the necessary underlying hard problems and primitives on which security will be based. Then in Section 3 we discuss various different security models for key exchange followed by usage of the exchanged keys in establishing a secure channel. This is followed in Section 4 by a proof that the proposed EMV key agreement protocol satisfies our security definition. In Section 5 we turn to the unlinkability requirement of the EMV designers; here, we show that the amount of unlinkability is related to the size of the randomness used in masking the public key in the first message flow.

We end this introduction by pointing out a number of recommendations related to the EMV protocol which have been passed to the designers as a result of our analysis:

1. The resulting Diffie–Hellman key should be hashed down to obtain the used symmetric keys. The proposal in [11] says to use a hash function or the x-coordinate of the elliptic curve point as the key deriva-

tion function. We do not consider a choice not using a hash function to be secure; indeed our security analysis crucially relies on the hash being taken.

2. The resulting keys should be used in a uni-directional manner; thus two keys need to be obtained from the hashing process. This avoids a large number of potential replay attacks on the application layer. Else, the application layer would need to be implemented extremely carefully to thwart these attacks. Having two keys, one for each direction, makes the design of a secure application layer less vulnerable. We have implicitly assumed, as this is not stated in [11], that the resulting secure channel should be secure against adversaries both deleting messages and playing messages out of order; since this is the usual definition of a secure channel.
3. The amount of unlinkability is controlled by the size of the blinding factor applied to the first message flow. In the document [11] this value a is selected from the set $\{0,1\}^{32}$, for which an adversary can break the unlinkability with effort of roughly 2^{16} group operations. This range therefore does not provide sufficient protection where unlinkability is a concern.

2 Scheme

Our presentation follows that in [11], augmented with information obtained from public discussions with the authors of the protocol at various meetings. The basic underlying idea of the protocol is to use a Diffie–Hellman key exchange in which one side (the card) has a static public key, however to achieve unlinkability the certificate of this public key is not passed in the clear, and in addition the resulting Diffie–Hellman key is additionally randomized by a small emphemeral secret. This randomization by a small ephemeral secret is reminiscent to the use of small exponents in the MQV key agreement protocol [10]. The resulting Diffie–Hellman key is then hashed using a cryptographic hash function; which we will model as a random oracle.

The Diffie–Hellman group used by the protocol is defined over an elliptic curve $G = E(\mathbb{F}_p)$ having group order a prime q. The prime q is a function of an implicit security parameter k, but in practice the group is fixed and so all our results are given in the concrete security setting. Along with the group G a base point $P \in G$ is given. It is assumed that the following Gap Diffie–Hellman problem is hard to solve for the choice of group G:

Definition 1 (Gap Diffie–Hellman). *Let $\mathcal{O}_{\mathsf{DDH}}$ be an oracle that solves the DDH problem in G, i.e. the oracle takes as input $rP, sP, tP \in G$, and outputs one if $tP = rsP$ and zero otherwise.*
The Gap Diffie–Hellman problem then asks that given $aP, bP \in G$ where $a, b \xleftarrow{r} \mathbb{F}_q$, and access to $\mathcal{O}_{\mathsf{DDH}}$, compute abP (i.e. solve CDH). The advantage of an adversary $\mathcal{A}$ against the Gap Diffie–Hellman problem is defined by

$$\mathbf{Adv}_G^{\mathsf{Gap-DH}}(\mathcal{A}) = \Pr[a, b \xleftarrow{r} \mathbb{F}_q : \mathcal{A}^{\mathcal{O}_{\mathsf{DDH}}}(aP, bP) = abP].$$

To prove our unlinkability property we will also require that the following problem (parametrized by an integer l) be hard:

Definition 2 (Small Decisional Discrete Log (SDDL)). *Given $P, X_0, X_1, rX_i \in G$, where $0 \leq r \leq 2^l < q$ determine i. We define the associated advantage of an adversary $\mathcal{A}$ by the following statement*

$$\mathbf{Adv}_{G,\mathcal{A}}^{\mathsf{SDDL}}(k) = \Pr[i \xleftarrow{r} \{0,1\}, r \xleftarrow{r} \{0,1\}^l, X_0, X_1 \xleftarrow{r} G : \mathcal{A}(X_0, X_1, rX_i) = i] - \frac{1}{2}$$

We pause to discuss the hardness of this problem. If $2^l \approx q$, then the distributions $(X_0, X_1, , rX_0)$ and (X_0, X_1, rX_1) are statistically close, i.e., the advantage is essentially zero even if the adversary is computationally unbounded. So the real question of interest is how small can l be before the above problem becomes easy for computationally bounded adversaries. It is also clear that the best attack against the problem for $2^l \ll q$ will be Pollard Lambda method [14], which runs in time $O(2^{l/2})$. This implies that a 32-bit randomizer r only gives 16-bits of security and an 80-bit randomizer only gives 40 bits of security.

In the following protocol we will see that one can trade efficiency against the difficulty of the SDDL problem. Thus if unlinkability is not a concern one can select a small l and if unlinkability is a concern one can select a larger l. In particular if one required the unlinkability to hold for an adversary willing to devote $O(2^b)$ effort then one should select $l = 2 \cdot b$. If however unlinkability is not a concern then one can select $l = 0$ and $r = 1$.

After the protocol determines secret keys these are then used in a stateful authenticated-encryption scheme $\mathsf{AE} = \{\mathsf{enc}_\kappa(), \mathsf{dec}_\kappa()\}$. The key-agreement scheme should generate a randomly distributed key in order to satisfy the following definitions. Here $\mathsf{enc}_\kappa()$ takes as input a message m, a header h, some state information st_e and returns a ciphertext c and updated state st_e. The decryption algorithm takes as input a ciphertext c, a header h, some state information st_d and returns a plaintext m and updated state st_d. The states st_e and st_d here model the fact that in practice sequence numbers are used to ensure that messages are delivered in order, thus the operations encrypt and decrypt become stateful. This authenticated encryption scheme we assume satisfies the standard properties of indistinguishability under chosen message attack and integrity of ciphertexts for such stateful schemes. See Appendix A for precise definitions of these security notions.

We also assume that there is a public key signature algorithm used to define certificates. In particular each card C has a long term public/private key pair (Q_C, d), where $d \in \mathbb{F}_q$ and $Q_C = dP \in G$. A certificate is a signature/message pair $\mathrm{cert}_C = (\mathsf{sig}_{\mathsf{sk}}(Q_C), Q_C)$ provided by an issuing authority with a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ for some (unspecified) public key signature algorithm $(\mathsf{sig}, \mathsf{ver})$. All that we require of the signature algorithm is that it be existentially unforgeable under a chosen message attack. Again Appendix A gives the precise security definition we will use.

We are now in a position to define the EMV key establishment and secure channel protocol in Figure 1. As well as the components above the protocol makes use of a hash function H which takes elements in the group G and maps them onto a pair of keys for the authenticated encryption scheme. The keys are used to secure the communication in both directions; we propose the use of two keys so that replay attacks are prevented at the level of the protocol as opposed to needing to be dealt with at the application layer.

3 Security Models

As mentioned in the introduction Jager et al. [8] combine the notions of authenticated key exchange [2, 3] and LHAE security [13] to give a combined notion of secure channel establishment. In Appendix B we first present their definition in detail, in this section we identify some definitional issues their approach and give our new definition. In Appendix C we present a minor modification of Jager et al's definition which we prove is equivalent to our new notion. We feel our new notion is conceptually simpler both to understand and to use.

Our analyis is not concerned with the length hiding properties used by Jager et al. [8] and Paterson et al. [13] so we omit this aspect and consider only stateful authenticated encryption (sAE).

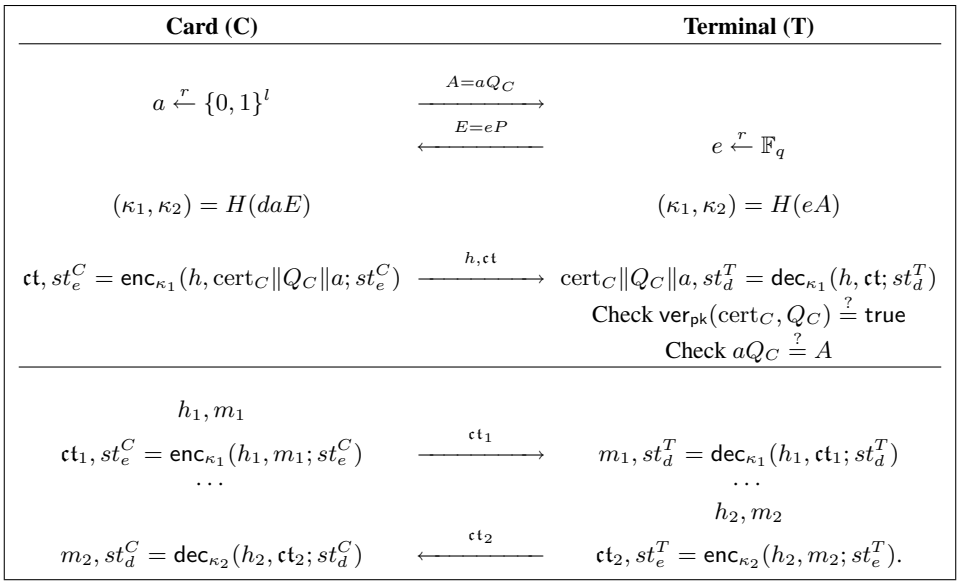

Fig. 1. Combined Authenticed Key Agreement Scheme and Secure Channel Protocol

3.1 Problems with the ACCE Definition

Our first issue with the model in [8] comes from the definition of the states used in the Encrypt and Decrypt queries. The decryption query requires a check whether the ciphertext was previously output by the encryption oracle but these states have not been defined correctly for multiple parties. Consider the query $\mathsf{Decrypt}(\Pi^s_{i,j}, c, h)$. To complete a decryption the definition checks whether $v > u$ or $c \neq c_v$. If either is true then the decrypted message will be returned, otherwise nothing is returned (this originally comes from stateful decryption model of Bellare et al. [1]). Here u and v were both defined with relation to $\Pi^s_{i,j}$ but what we actually need to compare is the u of $\Pi^t_{j,i}$ and the v of $\Pi^s_{i,j}$. Plainly, we need to check that the ciphertext c that i decrypts was not output by an encryption performed by j. In such a case the adversary should not be allowed to see the decryption otherwise he can trivially win.

More issues with these checks may occur if we were to consider the case where the same key is used in both directions of the channel. We would therefore need to check that a ciphertext had not been previously output by both $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$. Such a problem should be avoided by appropriate checks on send/receiver ids and the states of the messages.

Perhaps the most notable issue is the following: In both, the new EMV scheme that we consider and TLS as considered by Jager et al., the final step involves sending an encrypted message which is used to perform the final authentication of the sender. In [8] this corresponds to the message m_{13}. Immediately after sending this message the server will be in an accepted state. We are then permitted to issue a reveal query for the key of the server. Now the adversary may decrypt m_{13} and re-encrypt it using the revealed key but with new randomness. By then sending this new ciphertext instead of the original m_{13} the client will still accept but will no longer have had a matching conversation with the server (despite the plaintext conversation being the same). This therefore invalidates the security requirements of their model.

3.2 Preliminaries

Before presenting our new definition we present some prelininary definitions. Let I be the set of participants. Each participant has a distinct ID i, long-term public key pk_i and corresponding secret key sk_i. The protocol

description is defined by two efficiently computable functions $P = \{\Pi, \mathcal{G}\}$. The function Π defines how honest parties behave and $\mathcal{G}$ is a public/private key pair generation algorithm. We let $\Pi^s_{i,j}$ denote the oracle modelling participant $i \in I$ engaged in a protocol with participant $j \in I$ in session s. Each $\Pi^s_{i,j}$ maintains the following state information:

- $st \in \{0,1\}^*$ is some state information including the conversation so far in both plaintext form and the form it was sent on the channel,
- $\delta \in \{\texttt{accept}, \texttt{reject}, \perp\}$ is the decision (initialised to $\perp$).
- $\rho \in \{\texttt{initiator}, \texttt{responder}\}$ is the role of the participant.
- $\kappa = (k^\rho_{\mathsf{enc}}, k^\rho_{\mathsf{dec}}) \in (\{0,1\}^* \cup \{\perp\})^2$. This is the agreed pair of keys. The order of these keys depends on the role $\rho = \{\texttt{initiator}, \texttt{responder}\}$ and $\kappa = (\perp, \perp)$ unless $\delta = \texttt{accept}$.

An adversary $\mathcal{A}$, which is assumed to control all communication between participating parties, $\mathcal{A}$ can make the following queries:

- $\mathsf{NewSession}(i, \rho)$: Create a new session for user i with role ρ either initiator (card) or responder (terminal).
- $\mathsf{Send}(\Pi^s_{i,j}, m)$: Sends message m from user i to j in session s.
- $\mathsf{Reveal}(\Pi^s_{i,j})$: reveals the current session key κ.
- $\mathsf{Corrupt}(i)$: reveals the long-term private key of i and replaces it with K.

Our security model makes crucial use of a variant of the standard notion of matching conversation. In the standard definition a conversation is said to match if the messages sent "on the wire" match; we call this a wire-matching-conversation to avoid confusion with our new notion which we call plaintext-matching conversation. The basic idea is that two conversations match in their plaintexts if they match on the underlying messages; i.e. after any layers of encryption have been stripped off by the parties.

Definition 3 (Plaintext Conversation). *For an adversary $\mathcal{A}$ and oracle $\Pi^s_{i,j}$ it's plaintext conversation is defined to be the sequence of tuples*

$$(\tau_1, \alpha_1, \beta_1), (\tau_2, \alpha_2, \beta_2), ..., (\tau_m, \alpha_m, \beta_m)$$

where $\tau_m > \tau_{m-1} > ... > \tau_1$ are times and for time τ_t the oracle $\Pi^s_{i,j}$ received α_t after decryption (if any) and responds with β_t prior to any encryption operation.

Definition 4 (Plaintext Matching Conversations). *Let Π be an R move protocol with $R = 2\rho - 1$. Let $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ be two oracles with plaintext conversations C and C' obtained after running the protocol in the presence of $\mathcal{A}$.*

1. *We say that C' is a plaintext matching conversation to C if there exist $\tau_0 < \tau_1 < ... < \tau_R$ and $\alpha_1, \beta_1, ..., \alpha_\rho, \beta_\rho$ such that C is prefixed by*

   $$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), ..., (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

   and C' is prefixed by

   $$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), ..., (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1})$$

2. *We say that C is a plaintext matching conversation to C' if there exists $\tau_0 < \tau_1 < ... < \tau_R$ and $\alpha_1, \beta_1, ..., \alpha_\rho, \beta_\rho$ such that C' is prefixed by*

   $$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), ..., (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau 2\rho - 1, \alpha_\rho, *)$$

and C is prefixed by

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), ..., (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

We then say that $\Pi^t_{j,i}$ has a plaintext matching conversation with $\Pi^s_{i,j}$ if the first has plaintext conversation C', the second has plaintext conversation C, and C' matches C.

We also define the probability $\Pr[\mathsf{NoMatchPlntxt}(\mathcal{A})]$ that an adversary $\mathcal{A}$ can cause the event where at least one oracle $\Pi^s_{i,j}$ has no plaintext matching conversation.

As is usual in security of key establishment protocols the security definitions are based on the notion of a fresh oracle; although usually this is given in the context of what we called wire matching conversations:

Definition 5 (Fresh). *An oracle $\Pi^s_{i,j}$ is **fresh** if the following three conditions hold:*

1. *$\Pi^s_{i,j}$ has accepted.*
2. *Oracle $\Pi^s_{i,j}$ has not been revealed and user i is not corrupted.*
3. *No oracle that has had a plaintext matching conversation with $\Pi^s_{i,j}$ has been revealed and no parent of such a oracle has been corrupted.*

3.3 Our New (Three-Part) Definition

In the definition of Jager et al. (see Appendix B) the adversary is permitted to make three different types of query $\mathsf{Send}^{\mathrm{pre}}$, Encrypt and Decrypt associated to each oracle $\Pi^s_{i,j}$. The Encrypt and Decrypt operations are defined separately from $\mathsf{Send}^{\mathrm{pre}}$ with the proviso that once the oracle has accepted decryption queries are handled by Decrypt rather than $\mathsf{Send}^{\mathrm{pre}}$. In practice a secure channel would not have such a distinction in the calls that can be made, since an adversary may not know when an oracle reaches an accept state. It makes more sense then that we have a single Send operation which handles all operations depending on the state of the input oracle $\Pi^s_{i,j}$.

Thus to achieve greater generality and mirror practice more effectively we resort to only using Send queries. When calling Send an adversary will specify a message m and an operation op. Basic operations may include encrypt and decrypt, thus incorporating the queries of previous definitions. But it may also include other capabilities such as sign not previously captured by the previous definition. This allows an adversary to specify whether he is sending a message on the channel (ciphertext) or inputing an application message (plaintext). Furthermore, our new style of definition allows us to define the program that we shall consider as the correct operation of an honest oracle $\Pi^s_{i,j}$. In our analysis we shall consider a simple composition which consists of an initial key-exchange phase keyexch that enables the establishment of a secure channel based on a stateful authenticated encryption scheme $\mathsf{AE} = (\mathsf{enc}, \mathsf{dec})$. We implicitly assume that the implied state of the encryption and decryption functions, which we denote by st_e and st_d, is initially set to a default value on their first use.

The algorithms keyexch, enc and dec shall each maintain states st_k, st_e, st_d, respectively (we allow keyexch to run enc and hence update its state). Once keyexch has been run a number of times a party will accept the key exchange and establish two keys κ_1 for outgoing message and κ_2 for incoming messages. With each message m sent on the channel there exists a corresponding header field h with any associated data. We define our particular program for the honest operation of a participant as in Figure 2.

Next we give our security definition. This definition is split into three parts: entity authentication, message authentication and message privacy. Note that our definition is given with set of send operations $\mathtt{op} \in \{\mathtt{encrypt}, \mathtt{decrypt}\}$. This is easily extended to considering further types of operations by including further restrictions on the Send queries.

$\Pi(m, \mathtt{op})$:

- **if** $\delta \neq \mathtt{accept}$ **then**
  - $(m', st_k) \leftarrow \mathsf{keyexch}(h, m, st_k)$.
  - **return** m'. //m' may be the empty string
- **else**
  - **if** $\mathtt{op} = \mathtt{encrypt}$ **then**
    - $(m', st_e) \leftarrow \mathsf{enc}_{\kappa_1}(h, m, st_e)$,
    - **return** m'.
  - **elseif** $\mathtt{op} = \mathtt{decrypt}$
    - $(m', st_d) \leftarrow \mathsf{dec}_{\kappa_2}(h, m, st_d)$.
    - **return** m'
  - **else return** $\perp$

Fig. 2.

We shall first define the two separate security experiments for message authentication and privacy. In these experiments we maintain lists for each $\Pi^s_{i,j}$ as follows:

- Application messages sent (encryption performed before sending) $L^{app|sen}_{i,j,s}$.
- Application messages received (after decryption) $L^{app|rec}_{i,j,s}$.
- Channel messages sent $L^{ch|sen}_{i,j,s}$.
- Channel messages received $L^{ch|rec}_{i,j,s}$.

By maintaining lists in this fashion instead of using states (for example u and v in the ACCE definition) our definition can be easily adapted to consider the case of fragmentation. This is a common feature of many secure/authenticated channels in practice and has been formally studied by Boldyreva et al. [4].

Message Authentication First consider the authentication experiment auth that initializes the encryption and decryption states to their default values, and then allows the adversary to make the queries $\mathsf{NewSession}(i, \rho)$, $\mathsf{Reveal}(\Pi^s_{i,j})$, $\mathsf{Corrupt}(i)$ as well as $\mathsf{Send}(\Pi^s_{i,j}, m, \mathtt{op})$ with operations $\mathtt{op} \in \{\mathtt{encrypt}, \mathtt{decrypt}\}$. Note that as specified in Figure 2, a session ignores op, unless it is in an accept state. On the $\mathsf{Send}(\Pi^s_{i,j}, m, \mathtt{op})$ query, the game behaves as follows:

- **if** $\delta \neq \mathtt{accept}$ **then**
  - Run $m' \leftarrow \Pi^s_{i,j}(m, \mathtt{op})$.
  - **return** m'.
- **elseif** $\mathtt{op} = \mathtt{encrypt}$, **then**
  - Run $m' \leftarrow \Pi^s_{i,j}(m, \mathtt{encrypt})$.
  - Set $L^{app|sen}_{i,j,s} \leftarrow L^{app|sen}_{i,j,s} \| m$.
  - Set $L^{ch|sen}_{i,j,s} \leftarrow L^{ch|sen}_{i,j,s} \| m'$ and **return** m'.
- **elseif** $\mathtt{op} = \mathtt{decrypt}$, **then**
  - Run $m' \leftarrow \Pi^s_{i,j}(m, \mathtt{decrypt})$.
  - **if** $m' \neq \perp$, **then** $L^{app|rec}_{i,j,s} \leftarrow L^{app|rec}_{i,j,s} \| m'$.
  - **return** m'.

For the session matching, we consider plaintext session matching as specified in Definition 4. However, we only consider the messages sent and received in a session while $\delta \neq \texttt{accept}$. The notion of freshness that we use in the following definition is according to Definition 5.

We define the following game $\mathbf{Exec}_{\Pi}^{\mathsf{auth}}(\mathcal{A})$ between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

1. The challenger $\mathcal{C}$ generates public/private key pairs for each user $i \in I$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.[3]
2. Adversary $\mathcal{A}$ is allowed to make as many $\mathsf{NewSession}$, Reveal, $\mathsf{Corrupt}$, Send queries as it likes.
3. The adversary stops with no output.

We say that an adversary $\mathcal{A}$ wins the game if there exists a fresh oracle $\Pi_{i,j}^{s}$ with partner $\Pi_{j,i}^{t}$ such that the list $L_{i,j,s}^{app|rec}$ is not a prefix of $L_{j,i,t}^{app|sen}$. Let $\mathsf{Prefix}(X, Y)$ be the function which outputs 1 if $L_{i,j,s}^{app|rec}$ is a prefix of $L_{j,i,t}^{app|sen}$ (provided not empty) and 0 otherwise.

We define the adversary's advantage as:

$$\mathbf{Adv}_{\Pi}^{\mathsf{auth}}(\mathcal{A}) = \Pr[\mathsf{Prefix}(L_{i,j,s}^{app|rec}, L_{j,i,t}^{app|sen}) = 0 : \text{for some fresh } \Pi_{i,j}^{s}].$$

Message Privacy The message privacy experiment initializes the states as the authentication experiment auth, except that each session now also holds a random secret bit $b_{i,j}^{s}$. As before, the adversary can make the queries $\mathsf{NewSession}(i, \rho)$, $\mathsf{Reveal}(\Pi_{i,j}^{s})$, $\mathsf{Corrupt}(i)$ as well as $\mathsf{Send}(\Pi_{i,j}^{s}, m, \mathtt{op})$ with operation $\mathtt{op} = \mathtt{decrypt}$ as well as an augmented Send query $\mathsf{SendLR}(\Pi_{i,j}^{s}, m_0, m_1, \mathtt{op})$ that takes as input two messages (m_0, m_1) and $\mathtt{op} = \mathtt{encrypt}$ to model message indistinguishability. Note that as specified in Figure 2, a session ignores $\mathtt{op}$, unless it is in an $\mathtt{accept}$ state. As before, two sessions are considered partners, if they have matching plaintext conversations on the messages sent and received while both sessions were not in $\mathtt{accept}$ state yet. On the $\mathsf{SendLR}(\Pi_{i,j}^{s}, m_0, m_1, \mathtt{op})$ query, the game behaves as follows:

- **if** $\delta \neq \mathtt{accept}$, **then**
  - Run $m' \leftarrow \Pi_{i,j}^{s}(m, \mathtt{op})$.
  - **return** m'.
- **elseif** $\mathtt{op} = \mathtt{encrypt}$, **then**
  - Run $m' \leftarrow \Pi_{i,j}^{s}(m_{b_{i,j}^{s}}, \mathtt{encrypt})$.
  - Set $L_{i,j,s}^{app|sen} \leftarrow L_{i,j,s}^{app|sen} \| m_{b_{i,j}^{s}}$.
  - Set $L_{i,j,s}^{ch|sen} \leftarrow L_{i,j,s}^{ch|sen} \| m'$.
  - **return** m'.

On the $\mathsf{Send}(\Pi_{i,j}^{s}, m, \mathtt{op})$ query, the game behaves as follows:

- **if** $\delta \neq \mathtt{accept}$ **then**
  - Run $m' \leftarrow \Pi_{i,j}^{s}(m, \mathtt{op})$.
  - **return** m'.
- **elseif** $\mathtt{op} = \mathtt{decrypt}$ **then** (where $\Pi_{j,i}^{t}$ is communicating partner of $\Pi_{i,j}^{s}$)
  - **if** $m' \neq \perp$, **then** $L_{i,j,s}^{app|rec} \leftarrow L_{i,j,s}^{app|rec} \| m'$ and
    - **if** $m \notin L_{j,i,t}^{ch|sen}$, **return** m'.

[3] Note that in the scheme considered in this paper, public keys of cards are not actually made public to $\mathcal{A}$ but are sent in encrypted form during the confirmation step.

$*$ **if** $m \notin L_{j,i,t}^{ch|sen}$, **return** $\perp$.
- **if** $m' = \perp$, then return $\perp$.

To consider privacy we give an extended definition for Send, namely SendLR. Here the input consists of two messages for the encryption operation and the output would be the encryption of the message $m_{b_{i,j}^s}$, where $b_{i,j}^s$ is the random bit of $\Pi_{i,j}^s$. Now we describe the privacy experiment priv.

We define the following game $\mathbf{Exec}_{\Pi}^{\mathsf{priv}}(\mathcal{A})$ between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

1. The challenger $\mathcal{C}$, generates public/private key pairs for each user $i \in I$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.[4]
2. Adversary $\mathcal{A}$ is allowed to make as many NewSession, Reveal, Corrupt, SendLR queries as it likes.
3. Finally $\mathcal{A}$ outputs a tuple (i, j, s, b').

We say the adversary $\mathcal{A}$ wins if its output $b' = b_{i,j}^s$ and $\Pi_{i,j}^s$ is fresh. In this case the output of $\mathbf{Exec}_{\Pi}^{\mathsf{priv}}(\mathcal{A})$ is set to 1. Otherwise the output is 0. Formally we define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\Pi}^{\mathsf{priv}}(\mathcal{A}) = |\Pr[\mathbf{Exec}_{\Pi}^{\mathsf{priv}}(\mathcal{A}) = 1] - 1/2| = |\Pr[b' = b_{i,j}^s] - 1/2|.$$

We can now present our three part security definition for a combined key establishment and authenticated channel protocol:

Definition 6 (EAMAP). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ)-**secure EAMAP protocol** if for all adversaries $\mathcal{A}_{\mathsf{auth}}$ and $\mathcal{A}_{\mathsf{priv}}$ running each in time at most t the following conditions hold (where $\epsilon = \epsilon_{EA} + \epsilon_{MA} + \epsilon_{MP}$):*

1. *(Entity Authentication (EA)): In each of the experiments $\mathbf{Exec}_{\Pi}^{\mathsf{auth}}(\mathcal{A}_{\mathsf{auth}})$ and $\mathbf{Exec}_{\Pi}^{\mathsf{priv}}(\mathcal{A}_{\mathsf{priv}})$, there exists with probability at most ϵ_{EA} an oracle $\Pi_{i,j}^s$ such that:*
   - *$\Pi_{i,j}^s$ accepts at time τ and j is uncorrupted prior to this time, and*
   - *there is no unique oracle $\Pi_{j,i}^t$ such that $\Pi_{i,j}^s$ has a plaintext matching conversation with $\Pi_{j,i}^t$.*
2. *(Message Authentication (MA)): In the experiment $\mathbf{Exec}_{\Pi}^{\mathsf{auth}}(\mathcal{A}_{\mathsf{auth}})$, the advantage of $\mathcal{A}_{\mathsf{auth}}$ is bounded by $\mathbf{Adv}_{\Pi}^{\mathsf{auth}}(\mathcal{A}_{\mathsf{auth}}) \leq \epsilon_{MA}$.*
3. *(Message Privacy (P)): In the experiment $\mathbf{Exec}_{\Pi}^{\mathsf{priv}}(\mathcal{A}_{\mathsf{priv}})$, the advantage of $\mathcal{A}_{\mathsf{priv}}$ is bounded by $\mathbf{Adv}_{\Pi}^{\mathsf{priv}}(\mathcal{A}_{\mathsf{priv}}) \leq \epsilon_{MP}$.*

3.4 One-Sided Authentication

The above security definitions enforce mutual authentication, yet in many scenarios of practical concern only one party needs to be authenticated. For example, the protocol we consider requires authentication of the credit card but does not authenticate the communicating terminal. To model this situation we split our set of participants I in two. Let C be the set of registered participants (the cards) and let T by the set of unauthenticated participants (the terminals). Unauthenticated participants do not hold a long-term private/public key pair. The total set of all participants is therefore $I = C \cup T$. This is defined in the same way as registered and unregistered users in [12]. In our scenario $i \in C$ shall model a card and $j \in T$ a terminal. A terminal $j \in T$ wishes to authenticate a card $i \in C$ and establish a key (additionally a secure channel) with this card. Since all $j \in T$ have no long-term secret then the Corrupt query does not make

[4] Note that in the scheme considered in this paper, public keys of cards are not actually made public to $\mathcal{A}$ but are sent in encrypted form during the confirmation step.

sense for any such $j \in T$. Similarly since the adversary does not need to query the oracle to find the output for user $j \in T$ then he may always be able to compute the session key for $\Pi^s_{i,j}$ without using a Reveal query. We wish to guarantee that an adversary cannot determine the session key or authenticate itself as a participant $i \in C$. An adversary impersonating some unauthenticated participant $j \in T$ would always succeed in establishing a malicious session (key) with $i \in C$

First let us define a one-sided version of fresh.

Definition 7 (One-Sided Fresh). *An oracle $\Pi^s_{i,j}$ where $i \in I = C \cup T$ and $j \in I = C \cup T$, is **OS-fresh** if the following six conditions hold:*

1. *Both i and $j \notin T$, i.e. at least one is a registered participant.*
2. *$\Pi^s_{i,j}$ has accepted.*
3. *Oracle $\Pi^s_{i,j}$ has not been revealed.*
4. *If $i \in C$ then it is uncorrupted.*
5. *If $i \in T$ then $\Pi^s_{i,j}$ has had a plaintext matching conversation with an oracle $\Pi^t_{j,i}$. (This is to ensure that the adversary is not impersonating an unregistered user j).*
6. *No oracle that has had a plaintext matching conversation with $\Pi^s_{i,j}$ has been revealed and no parent of such a oracle has been corrupted if then are a registered participant.*

We can therefore alter our definitions of auth and priv whether the winning conditions now require the oracle to be OS-fresh. We denote our new definitions OS-auth and OS-priv respectively.

Definition 8 (OS-EAMAP). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ)-**secure OS-EAMAP protocol** if for all adversaries $\mathcal{A}$ running in time t the following conditions hold (where $\epsilon = \epsilon_{EA} + \epsilon_{MA} + \epsilon_{MP}$):*

1. *(One-Sided Entity Authentication): There exists with probability at most ϵ_{EA} an oracle $\Pi^s_{i,j}$ where $i \in I = C \cup T$ and $j \in C$ such that:*
   - *$\Pi^s_{i,j}$ accepts at time τ and j is uncorrupted prior to this time, and*
   - *there is no unique oracle $\Pi^t_{j,i}$ such that $\Pi^s_{i,j}$ has a plaintext matching conversation with $\Pi^t_{j,i}$.*
2. *(One-Sided Message Authentication): When $\mathcal{A}$ terminates the advantage $\mathbf{Adv}^{\mathsf{OS\text{-}auth}}_{\Pi}(\mathcal{A}) \leq \epsilon_{MA}$.*
3. *(One-Sided Message Privacy): When $\mathcal{A}$ terminates and outputs (i, j, s, b') the advantage $\mathbf{Adv}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A}) \leq \epsilon_{MP}$.*

Matching Conversations We must also define the notion of a *one-sided plaintext matching conversation* by extending the original definition in a similar fashion. Here we basically wish to establish that a card cannot establish a malicious session with a trusted terminal, i.e. an unauthenticated participant should never get to an accept state without having a matching conversation with a registered participant. The definition of a plaintext matching conversation shall remain the same but we must define the probability $\Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{A})]$ that an adversary $\mathcal{A}$ can cause the event where at least one terminal oracle $\Pi^s_{i,j}$ (for an unauthenticated participant $i \in T$ and registered participant $j \in C$) has no plaintext matching conversation.

4 Proof of Security

Theorem 1. *The EMV protocol Π in Figure 1 is an (t, ϵ)-OS-EAMAP secure protocol. In particular*

– *If there is an adversary $\mathcal{A}$ running in time at most t against the no plaintext matching property of the protocol then there is an adversary $\mathcal{B}$ such that*

$$\Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{A})] \leq \mathbf{Adv}^{\mathsf{eufcma}}_{\mathrm{cert}}(\mathcal{B}) = \epsilon_{EA}.$$

– *If there is an adversary $\mathcal{A}$ running in time at most t against the message authentication property of OS-EAMAP security then there are adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathbf{Adv}^{\mathsf{OS\text{-}auth}}_{\Pi}(\mathcal{A}) \leq n_S{\cdot}(n_C{+}n_T){\cdot}\mathbf{Adv}^{\mathsf{intsfctxt}}_{\mathsf{AE}}(\mathcal{D}){+}n_C{\cdot}(1{-}1/H){\cdot}\mathbf{Adv}^{\mathsf{Gap-DH}}_{E(\mathbb{F}_p)}(\mathcal{C}){+}\mathbf{Adv}^{\mathsf{eufcma}}_{\mathrm{cert}}(\mathcal{B}) = \epsilon_{MA}.$$

where n_C is the number of cards in the system, n_T the number of terminals and n_S the number of sessions.

– *If there is an adversary $\mathcal{A}$ running in time at most t against the message privacy property of OS-EAMAP security then there are adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathbf{Adv}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A}) \leq n_S{\cdot}(n_C{+}n_T){\cdot}\mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{D}){+}n_C{\cdot}(1{-}1/H){\cdot}\mathbf{Adv}^{\mathsf{Gap-DH}}_{E(\mathbb{F}_p),}(\mathcal{C}){+}\mathbf{Adv}^{\mathsf{eufcma}}_{\mathrm{cert}}(\mathcal{B}) = \epsilon_{MP}.$$

The proof of this theorem is given in Appendix D.

5 Unlinkability

A further property that this protocol aims to achieve is unlinkability. This means that it should be hard for an adversary to determine when two particular sessions involve the same card. We define this security property in terms of the game $\mathbf{Exec}^{\mathsf{unlink}}_{\Pi}(\mathcal{A})$ between adversary $\mathcal{A}$ and challenger $\mathcal{C}$, which is defined as follows:

1. The challenger $\mathcal{C}$, generates public/private key pairs for each user $i \in C$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.[5]
2. Adversary $\mathcal{A}$ is allowed to make as many NewSession, Reveal, Corrupt, Send queries as it likes.
3. At some point $\mathcal{A}$ outputs two identities $i_0 \in C$ and $i_1 \in C$.
4. The challenger then chooses a bit $b \xleftarrow{r} \{0,1\}$ and create a new oracle $\mathcal{O} = \Pi^s_{i_b,j}$ for some fresh $j \in T$, by calling $\mathsf{NewSession}(i_b)$.
5. Adversary $\mathcal{A}$ then continues making queries NewSession, Reveal, Corrupt, Send; in these queries he can now use $\mathcal{O}$.
6. Eventually $\mathcal{A}$ stops and outputs a bit b'.

We say the adversary $\mathcal{A}$ wins if its output $b' = b$. In this case the output of $\mathbf{Exec}^{\mathsf{unlink}}_{\Pi}(\mathcal{A})$ is set to one, otherwise the output is zero. Formally we define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{unlink}}_{\Pi}(\mathcal{A}) = |\Pr[\mathbf{Exec}^{\mathsf{unlink}}_{\Pi}(\mathcal{A}) = 1] - 1/2| = |\Pr[b' = b] - 1/2|.$$

Definition 9. *Unlinkability A protocol $(\Pi, \mathsf{Game}) \Rightarrow 1$ is $(t, \epsilon_{\mathsf{unlink}})$-unlinkable, if for all adversaries $\mathcal{A}$ running in time t, $\mathbf{Adv}^{\mathsf{unlink}}_{\Pi}(\mathcal{A}) \leq \epsilon_{\mathsf{unlink}}$.*

Theorem 2. *If π is a key-secure key-agreement protocol, $\mathsf{AE} = (\mathsf{enc}, \mathsf{dec})$ is an ind-cca secure authenticated-encryption scheme and the SDDL problem is hard, then Π is secure in the sense of* unlink.

$$\mathbf{Adv}^{\mathsf{unlink}}_{\Pi}(\mathcal{A}) \leq n_C^2 \cdot \left(\mathbf{Adv}^{\mathsf{SDDL}}_{E(\mathbb{F}_p)}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{C}) + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{B})\right)$$

The proof of this theorem is given in Appendix E.

[5] Note that in the scheme considered in this paper, public keys of cards are not actually made public to $\mathcal{A}$ but are sent in encrypted form during the confirmation step.

6 Acknowledgements

This work was support in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. The second author was also partially supported by a Royal Society Wolfson Merit Award. Research supported in part by the Israel Ministry of Science and Technology (grant 3-9094) and by the Israel Science Foundation (grant 1155/11 and grant 1076/11).

References

1. Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Trans. Inf. Syst. Secur.*, 7(2):206–241, 2004.
2. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.
3. Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1997.
4. Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. Security of symmetric encryption in the presence of ciphertext fragmentation. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 682–699. Springer, 2012.
5. Christina Brzuska, Marc Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. Less is more: Relaxed yet composable security notions for key exchange. *IACR Cryptology ePrint Archive*, 2012:242, 2012.
6. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.
7. Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 2012. Online first; print version to appear.
8. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of tls-dhe in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293. Springer, 2012.
9. Caroline Kudla and Kenneth G. Paterson. Modular security proofs for key agreement protocols. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 549–565. Springer, 2005.
10. Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.
11. EMVCo LLC. EMV ECC key establishment protocols. *http://www.emvco.com/specifications.aspx?id=243*, 2012.
12. Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. The tls handshake protocol: A modular analysis. *J. Cryptology*, 23(2):187–223, 2010.
13. 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, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2011.
14. John Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

A Basic Security Definitions

The authenticated encryption scheme we assume satisfies the following two properties which are variants of the stateful security models of Bellare et al. [1] and Paterson et al. [13]:

Definition 10 (IND-sfCCA). *Consider the authenticated-encryption scheme* $\mathsf{AE} = \{\mathsf{enc}_\kappa(), \mathsf{dec}_\kappa()\}$. *Let* $\mathcal{A}$ *be an adversary with access to a left-or-right encryption oracle* $\mathsf{enc}_\kappa(h, \mathsf{LR}_b(m_0, m_1); st_e)$ *and a decryption oracle* $\mathsf{dec}_\kappa(h, c; st_d)$. *It is mandated that any two messages queried to* $\mathsf{enc}_\kappa(h, \mathsf{LR}_b(m_0, m_1); st_e)$ *have equal length. We define an experiment as follows:*

$\mathbf{Exec}^{\mathsf{indsfcca}-b}_{\mathsf{AE}}(\mathcal{A})$
$\kappa \xleftarrow{r} \{0,1\}^k$, $st_e \leftarrow \emptyset$ *and* $st_d \leftarrow \emptyset$
$u \leftarrow 0; v \leftarrow 0; \mathsf{phase} \leftarrow 0$
Run $\mathcal{A}^{\mathsf{enc}_\kappa, \mathsf{dec}_\kappa}$
  Reply to $\mathsf{enc}_\kappa(h, \mathsf{LR}_b(m_0, m_1); st_e)$ *as follows:*
    $u \leftarrow u + 1; (c_u, st_e) \xleftarrow{r} \mathsf{enc}_\kappa(h, m_b; st_e)$
    $\mathcal{A} \Leftarrow c_u$
  Reply to $\mathsf{dec}_\kappa(h, c; st_d)$ *as follows:*
    $v \leftarrow v + 1; (m, st_d) \xleftarrow{r} \mathsf{dec}_\kappa(h, c; st_d)$
    if $v > u$ *or* $c \neq c_v$ ***then*** $\mathsf{phase} \leftarrow 1$
    if $\mathsf{phase} = 1$ ***then*** $\mathcal{A} \Leftarrow m$
Until $\mathcal{A}$ *returns a bit* b'
return b'

The attacker wins when $b' = b$*, and his advantage is defined as*

$$\mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{A}) = \Pr[\mathbf{Exec}^{\mathsf{indsfcca}-1}_{\mathsf{AE}}(\mathcal{A}) = 1] - \Pr[\mathbf{Exec}^{\mathsf{indsfcca}-0}_{\mathsf{AE}}(\mathcal{A}) = 1].$$

Definition 11 (INT-sfCTXT). *Consider the authenticated-encryption scheme* $\mathsf{AE} = \{\mathsf{enc}_\kappa(), \mathsf{dec}_\kappa()\}$*. Let* $\mathcal{A}$ *be an adversary that has access to the oracles* $\mathsf{enc}_\kappa(h, m; st_e)$ *and* $\mathsf{dec}_\kappa(h, c; st_d)$*. We define an experiment as follows:*

$\mathbf{Exec}^{\mathsf{intsfctxt}}_{\mathsf{AE}}(\mathcal{A})$
$\kappa \xleftarrow{r} \{0,1\}^k$, $st_e \leftarrow \emptyset$ *and* $st_d \leftarrow \emptyset$
$u \leftarrow 0; v \leftarrow 0; \mathsf{phase} \leftarrow 0$
Run $\mathcal{A}^{\mathsf{enc}_\kappa, \mathsf{dec}_\kappa}$
  Reply to $\mathsf{enc}_\kappa(h, m; st_e)$ *as follows:*
    $u \leftarrow u + 1; (c_u, st_e) \xleftarrow{r} \mathsf{enc}_\kappa(h, m; st_e)$
    $\mathcal{A} \Leftarrow c_u$
  Reply to $\mathsf{dec}_\kappa(h, c; st_d)$ *as follows:*
    $v \leftarrow v + 1; (m, st_d) \xleftarrow{r} \mathsf{dec}_\kappa(h, c; st_d)$
    if $v > u$ *or* $c \neq c_v$ ***then*** $\mathsf{phase} \leftarrow 1$
    if $m \neq \perp$ *and* $\mathsf{phase} = 1$ ***then return*** *1*
    if $m \neq \perp$ ***then*** $\mathcal{A} \Leftarrow 1$ ***else*** $\mathcal{A} \Leftarrow 0$
Until $\mathcal{A}$ *halts*
return 0

The advantage $\mathbf{Adv}^{\mathsf{intsfctxt}}_{\mathsf{AE}}(\mathcal{A})$ *of an adversary is defined as the probability of* $\mathcal{A}$ *winning the above game.*

Definition 12 (EUF-CMA). *Consider the signature scheme* $\{\mathsf{keysig}, \mathsf{sig}, \mathsf{ver}\}$*, where* keysig *be the key generation method for this scheme. Let* $\mathcal{A}$ *be an adversary that has access to the oracle* $\mathsf{sig}_{sk}(\cdot)$*. We define the experiment as follows:*

$\mathbf{Exec}^{\mathsf{eufcma}}_{(\mathsf{sig},\mathsf{ver})}(\mathcal{A})$
$(pk, sk) \xleftarrow{r} \mathsf{keysig}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{sig}_{sk}(\cdot)}$
if $\mathsf{ver}_{pk}(m, \sigma) = 1$*; and* m *has not been queried to* $\mathsf{sig}_{sk}(\cdot)$
then return *1* ***else return*** *0*

The attacker's advantage is defined as

$$\mathbf{Adv}^{\mathsf{eufcma}}_{(\mathsf{sig},\mathsf{ver})}(\mathcal{A}) = \Pr[\mathbf{Exec}^{\mathsf{eufcma}}_{(\mathsf{sig},\mathsf{ver})}(\mathcal{A}) = 1].$$

B Jager et al's Definition of ACCE

We now present the ACCE definition of Jager et al. [8]. In this definition each oracle $\Pi^s_{i,j}$ maintains an additional state variable $b^s_{i,j} \xleftarrow{r} \{0,1\}$ chosen at random at the start of the game, two states u and v used to ensure $\mathcal{A}$ cannot submit a ciphertext previously output by the oracle queries Encrypt to Decrypt defined below, and two states st_e and st_d for the encryption and decryption operations of the stateful symmetric encryption scheme (each oracle $\Pi^s_{i,j}$ shall maintain a different set of states). As before we let enc and dec be the encryption and decryption algorithms of our symmetric encryption scheme. The adversary $\mathcal{A}$ will be permitted to make the following queries:

- $\mathsf{Send}^{\mathrm{pre}}(\Pi^s_{i,j}, m)$: This is identical to the Send query in the preliminaries section above, except that it replies with $\perp$ if oracle $\Pi^s_{i,j}$ has state $\delta = \texttt{accept}$ (this shall be handled by the decrypt query).
- $\mathsf{Reveal}(\Pi^s_{i,j})$ and $\mathsf{Corrupt}(i, K)$ are identical to those in the section above.
- $\mathsf{Encrypt}(\Pi^s_{i,j}, m_0, m_1, h)$: takes as input two equal length messages m_0 and m_1 and a header h. If $\Pi^s_{i,j}$ has $\delta \neq \texttt{accept}$ then $\Pi^s_{i,j}$ returns $\perp$. Otherwise it proceeds with encryption as in Figure 3 dependent on the internal state $b^s_{i,j}$.
- $\mathsf{Decrypt}(\Pi^s_{i,j}, c, h)$: takes as input a ciphertext c and a header h. If $\Pi^s_{i,j}$ has $\delta \neq \texttt{accept}$ then $\Pi^s_{i,j}$ returns $\perp$. Otherwise it proceeds with decryption as in Figure 3.

$\underline{\mathsf{Encrypt}(\Pi^s_{i,j}, m_0, m_1, h)}$	$\underline{\mathsf{Decrypt}(\Pi^s_{i,j}, c, h)}$
$u := u + 1$	$v := v + 1$
$(c^{(0)}, st_e^{(0)}) \leftarrow \mathsf{enc}(k^\rho_{\mathsf{enc}}, h, m_0)$	**if** $b^s_{i,j} = 0$ **then return** $\perp$
$(c^{(1)}, st_e^{(1)}) \leftarrow \mathsf{enc}(k^\rho_{\mathsf{enc}}, h, m_1)$	$(m, st_d) \leftarrow \mathsf{dec}(k^\rho_{\mathsf{dec}}, h, c, st_d)$
$(c_u, st_e) := (c^{(b)}, st_e^{(b)})$	**if** $v > u$ or $c \neq c_v$, **then** $\mathsf{phase} := 1$
return c_u	**if** $\mathsf{phase} = 1$ **then return** m
	else return $\perp$

Fig. 3.

We define the following game $\mathbf{Exec}^{\mathsf{ACCE}}_{\Pi}(\mathcal{A})$ between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

1. The challenger $\mathcal{C}$, generates public/secret key pairs for each user $i \in I$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ is allowed to make as many $\mathsf{Send}^{\mathrm{pre}}$, Reveal, Corrupt, Encrypt, Decrypt queries as it likes.
3. Finally $\mathcal{A}$ outputs a triple (i, j, s, b').

We say the adversary $\mathcal{A}$ wins if it outputs $b' = b^s_{i,j}$. In this case the output of $\mathbf{Exec}^{\mathsf{ACCE}}_{\Pi}(\mathcal{A})$ is set to 1. Otherwise the experiment returns 0. Formally we define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{ACCE}}_{\Pi}(\mathcal{A}) = |\Pr[\mathbf{Exec}^{\mathsf{ACCE}}_{\Pi}(\mathcal{A}) = 1] - 1/2| = |\Pr[b' = b^s_{i,j}] - 1/2|.$$

Definition 13 (ACCE). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ)-**secure ACCE protocol** if for all adversaries $\mathcal{A}$ running in time t the following conditions hold (where $\epsilon = \epsilon_{EA} + \epsilon_{sAE}$):*

1. *(Entity Authentication/EA): There exists with probability at most ϵ_{EA} an oracle $\Pi^s_{i,j}$ such that:*
   - *$\Pi^s_{i,j}$ accepts when $\mathcal{A}$ issues its τ_0-th query with partner j, and*
   - *P_j is uncorrupted with $\tau_0 < \tau_j$ (i.e. at time of accept), and*

– *there is no unique oracle $\Pi^t_{j,i}$ such that $\Pi^s_{i,j}$ has a (wire) matching conversation with $\Pi^t_{j,i}$.*

2. *(Secure Channel/sAE): When $\mathcal{A}$ terminates and outputs (i, j, s, b') such that*
   – *$\Pi^s_{i,j}$ accepts when $\mathcal{A}$ issues its τ_0-th query with intended partner j, and*
   – *P_j is uncorrupted with $\tau_0 < \tau_j$ (i.e. at time of accept), and*
   – *$\mathcal{A}$ did not issue a* Reveal*-query to $\Pi^s_{i,j}$ nor $\Pi^t_{j,i}$ (such that they had a wire matching conversation).*

   the advantage is bounded by $\mathbf{Adv}^{\mathsf{ACCE}}_{\Pi}(\mathcal{A}) = |\Pr[b' = b^s_{i,j}] - 1/2| \leq \epsilon_{sAE}$.

B.1 Note on unique conversations:

We point out that in the above definition of entity authentication requires that there exists a unique oracle $\Pi^t_{j,i}$ such that $\Pi^s_{i,j}$ has a wire matching conversation with $\Pi^t_{j,i}$. That is, there exists one and only one such oracle. Bellare and Rogaway prove [2, Appendix C] that if the probability of no wire matching conversations is "small" then the probability of $\Pi^s_{i,j}$ having a wire matching conversation with two different oracles $\Pi^t_{j,i}$ and $\Pi^{t'}_{j',i}$ is also "small". (The proof exploits one of the "multi-matching" oracles becoming out-of-sync but then still accepting.)

C Alternate ACCE

To fix the two problems with ACCE we could make the following changes to the original definition. First consider the two states u and v previously used during the Encrypt and Decrypt queries. We instead will define the two states $u^s_{i,j}$ and $v^s_{i,j}$ which are then used to ensure that $\mathcal{A}$ cannot submit a ciphertext previously output by Encrypt to Decrypt. Once an oracle is in an accept state then Encrypt and Decrypt shall now proceed as defined in Figure 4.

$\underline{\mathsf{Encrypt}(\Pi^s_{i,j}, m_0, m_1, h)}$	$\underline{\mathsf{Decrypt}(\Pi^t_{j,i}, \{c, h\})}$
$u^s_{i,j} := u^s_{i,j} + 1$	$v^t_{j,i} := v^t_{j,i} + 1$
$(c^{(0)}, st_e^{(0)}) \leftarrow \mathsf{enc}(k^{\rho}_{\mathsf{enc}}, h, m_0)$	**if** $b^t_{j,i} = 0$ **then return** $\perp$
$(c^{(1)}, st_e^{(1)}) \leftarrow \mathsf{enc}(k^{\rho}_{\mathsf{enc}}, h, m_1)$	$(m, st_d) \leftarrow \mathsf{dec}(k^{\rho}_{\mathsf{dec}}, h, c, st_d)$
$(c_u, st_e) := (c^{(b)}, st_e^{(b)})$	**if** $v^t_{j,i} > u^s_{i,j}$ or $c \neq c_v$, **then** $\mathsf{phase}^t_{j,i} := 1$
return c_u	**if** $\mathsf{phase}^t_{j,i} = 1$ **then return** m
	else return $\perp$

Fig. 4.

The rest of the security experiment remains the same and we denote this new experiment by $\mathbf{Exec}^{\mathsf{altACCE}}_{\Pi}(\mathcal{A})$, where the advantage of $\mathcal{A}$ is again defined as:

$$\mathbf{Adv}^{\mathsf{altACCE}}_{\Pi}(\mathcal{A}) = |\Pr[\mathbf{Exec}^{\mathsf{altACCE}}_{\Pi}(\mathcal{A}) = 1] - 1/2| = |\Pr[b' = b^s_{i,j}] - 1/2|.$$

The second change is then to fix the issue with the reveal query this occurs in the first part of the following *altered* definition.

Definition 14 (alternate-ACCE).
*A protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ)-**secure ACCE protocol** if for all adversaries $\mathcal{A}$ running in time t the following conditions hold (where $\epsilon = \epsilon_{EA} + \epsilon_{sAE}$):*

1. *(Entity Authentication/EA): There exists with probability at most ϵ_{EA} an oracle $\Pi^s_{i,j}$ such that:*
   - *$\Pi^s_{i,j}$ accepts when $\mathcal{A}$ issues its τ_0-th query with partner j, and*
   - *P_j is uncorrupted with $\tau_0 < \tau_j$ (i.e. at time of accept), and*
   - *there is no unique oracle $\Pi^t_{j,i}$ such that $\Pi^s_{i,j}$ has a plaintext matching conversation with $\Pi^t_{j,i}$.*
2. *(Secure Channel/sAE): When $\mathcal{A}$ terminates and outputs (i, j, s, b') such that*
   - *$\Pi^s_{i,j}$ accepts when $\mathcal{A}$ issues its τ_0-th query with intended partner j, and*
   - *P_j is uncorrupted with $\tau_0 < \tau_j$ (i.e. at time of accept), and*
   - *$\mathcal{A}$ did not issue a* Reveal*-query to $\Pi^s_{i,j}$ nor $\Pi^t_{j,i}$ (such that they had a wire matching conversation).*

   the advantage is bounded by $\mathbf{Adv}^{\mathsf{altACCE}}_{\Pi}(\mathcal{A}) = |\Pr[b' = b^s_{i,j}] - 1/2| \leq \epsilon_{sAE}$.

Please note that the authentication requirement uses the notion of plaintext matching conversations, while the secure channel definition makes use of the standard definition of (wire) matching conversations.

C.1 Relations between notions

By defining our new notion in three parts we create a notion with greater flexibility allowing it to be used in a "mix-and-match" style. In some situations it may be necessary to (along with entity authentication) ensure only one of message privacy or authenticity, where as in other situations both may be required. In the original ACCE paper by Jager et al. [8] it is necessary to achieve both message authenticity and privacy, namely sLHAE security. In the situation where we require all three conditions from a scheme (built as a key exchange followed by an authenticated encryption scheme) it is easy to prove that our notion of EAMAP and that of alternate-ACCE are equivalent. The analysis is similar to that when we compare AE security with IND-CCA security. A scheme which is AE secure is also IND-CCA secure but a scheme which is IND-CCA secure is not necessarily AE secure. Given that a scheme is both INT-CTXT and IND-CPA (or even IND-CCA security as in our case) secure it will achieve AE security. It may seem strange that our notion's constituent parts are INT-CTXT and IND-CCA rather than INT-CTXT and IND-CPA but this is due to our mix-and-match philosophy where we may only require the established channel to be solely IND-CCA secure rather than AE secure.

Theorem 3. *Consider an adversary $\mathcal{A}$ against $(\mathcal{G}, \Pi)$ where Π follows the construction from earlier with* $\mathsf{op} = \mathsf{Encrypt}$ *or* Decrypt*, i.e. key exchange followed by secure channel, then the security of alternate-ACCE and EAMAP are equivalent.*

The proof is similar to proving that a scheme which is AE secure is both IND-CCA secure and INT-CTXT secure, and a scheme which is both IND-CPA and INT-CTXT secure is also AE secure. The ACCE paper uses the results from [13] proving the sLHAE security of the MEE construction used in TLS. Since we have omitted the details for length-hiding from our notions we omit these from the proof also but is extension to the length-hiding setting is straight-forward.

Proof. (sketch) First let us prove that a scheme which is alternate-ACCE secure is also EAMAP secure. Since the requirements for entity authentication are the same it is trivial to show that one implies the other. Assume we have an adversary $\mathcal{A}$ which breaks the alternate-ACCE security. We shall use this adversary to construct two new adversaries $\mathcal{B}_{\mathsf{priv}}$ and $\mathcal{B}_{\mathsf{auth}}$ against EAMAP message privacy and authenticity. $\mathcal{B}_{\mathsf{priv}}$ and $\mathcal{B}_{\mathsf{auth}}$ will use their own oracles to provide simulations of $\mathcal{A}$'s oracles. The Corrupt and Reveal queries are straightforward and consists of simply forwarding messages to the corresponding oracle. When $\mathcal{A}$ makes a $\mathsf{Send}^{\mathrm{pre}}$ query then this is forwarded to the Send oracle with $\mathsf{op} = \emptyset$. When $\mathcal{A}$ makes a Encrypt query

then this is forwarded to the Send oracle with $\mathtt{op} = \mathtt{encrypt}$. When $\mathcal{A}$ makes a Decrypt query then this is forwarded to the Send oracle with $\mathtt{op} = \mathtt{decrypt}$.

Let E be the event that $\mathcal{A}$ creates a ciphertext forgery which by the end of the experiment has not been output by the encryption oracle. It is easy to verify that we then have:

$$\begin{aligned}\Pr[\mathcal{A} \text{ ``wins''}] &= \Pr[\mathcal{A} \text{ ``wins''} \wedge E] + \Pr[\mathcal{A} \text{ ``wins''} \wedge \neg E] \\ &\leq \Pr[\mathcal{B}_{\mathsf{auth}} \text{ ``wins''}] + \Pr[\mathcal{B}_{\mathsf{priv}} \text{ ``wins''}]\end{aligned}$$

The result then follows.

To prove that a scheme which is EAMAP secure is also alternate-ACCE secure Again, since the requirements for entity authentication are the same it is trivial to show that one implies the other. The rest follows from the proof in [13]. (Given a scheme which is INT-CTXT and IND-CPA secure we obtain an AE secure scheme.)

D Proof of Theorem 1

The proof of this theorem will be accomplished in the following subsections. Before proceeding with the main proof we first examine a related concept of Key Secrecy for a simpler protocol.

D.1 Key Secrecy

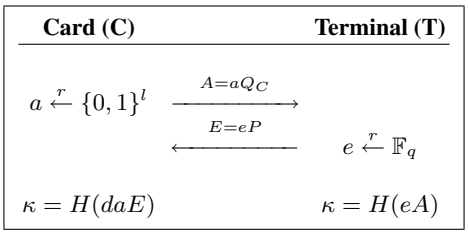

Fig. 5. Unauthenticated Key-Agreement Scheme

We begin our analysis by studying the simpler protocol, π, described in Figure 5. To analyse this protocol we are only interested in whether the secret key remains secret, and so we introduce a new security game to model this fact. Define the following game $\mathbf{Exec}_{\Pi}^{\mathsf{KSec}}(\mathcal{A})$ between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

1. The challenger $\mathcal{C}$, generates public/secret key pairs for each user $i \in I$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ is allowed to make as many NewSession, Send, Reveal, Corrupt queries as it likes.
3. Finally $\mathcal{A}$ outputs a pair Π^* and κ^*.

We say the adversary $\mathcal{A}$ wins if Π^* is fresh and κ^* is the key agreed by κ^*. In this case the output of $\mathbf{Exec}_{\Pi}^{\mathsf{Test}}(\mathcal{A})$ is set to 1. Otherwise the output is 0. Formally we define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\Pi}^{\mathsf{KSec}}(\mathcal{A}) = |\Pr[\mathbf{Exec}_{\Pi}^{\mathsf{KSec}}(\mathcal{A}) = 1]|.$$

Definition 15 (Key Secrecy). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a $(t, \epsilon_{\mathsf{KSec}})$-**key secret AK protocol** if for all adversaries $\mathcal{A}$ running in time t the following holds:*

1. *In the presence of a benign adversary on $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ both oracles accept holding the same session key κ, and this key is distributed uniformly at random on $\{0,1\}^k$.*
2. *If uncorrupted oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have had a plaintext matching conversation then both accept and hold the same session key κ.*
3. *$\mathcal{A}$'s advantage is bounded by $\mathbf{Adv}^{\mathsf{KSec}}_{\Pi}(\mathcal{A}) \leq \epsilon_{\mathsf{KSec}}$.*

We can also define a weaker version of this model for one-sided authentication by running the experiment in the same way as before but changing the winning condition slightly. We say the adversary $\mathcal{A}$ wins if Π^* is *OS-fresh* and κ^* is the key agreed by κ^*.

Definition 16 (Weak Key Secrecy). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a $(t, \epsilon_{\mathsf{wKSec}})$-**weak Key-secure AK protocol** if for all adversaries $\mathcal{A}$ running in time t the following holds:*

1. *In the presence of a benign adversary on $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ both oracles accept holding the same session key κ, and this key is distributed uniformly at random on $\{0,1\}^k$.*
2. *If uncorrupted oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have had a plaintext matching conversation then both accept and hold the same session key κ.*
3. *$\mathcal{A}$'s advantage is bounded by $\mathbf{Adv}^{\mathsf{wKSec}}_{\Pi}(\mathcal{A}) \leq \epsilon_{\mathsf{wKSec}}$.*

Given this definition we can now analyse the protocol in Figure 5.

Lemma 1. *The key secrecy of the reduced protocol π is reducible to the Gap Diffie–Hellman assumption, i.e. we have for all adversaries $\mathcal{A}$ there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{A}) \leq n_C \cdot (1 - 1/H) \cdot \mathbf{Adv}^{\mathsf{Gap-DH}}_{E(\mathbb{F}_p)}(\mathcal{B}),$$

where n_C is the number of cards in the system.

Proof. The proof of this lemma uses the technique first presented in [9] for analysing a hashed Diffie–Helman based key agreement protocol. Assume we have an adversary $\mathcal{A}$ against the key secrecy of π we shall use this to construct an adversary $\mathcal{B}$ against Gap Diffie–Hellman, where $\mathcal{B}$ is given the challenge aP, bP.

The algorithm $\mathcal{B}$ begins by setting up n_C registered participants by choosing a secret key $d_i \xleftarrow{r} \mathbb{F}_q$ for each registered participant $i \in C$ and sets the public key $Q_i = d_i P$ except for one participant $i^* \in I$ where we set the public key to aP. $\mathcal{B}$ also sets up n_T unregistered participants.

Algorithm $\mathcal{B}$ will then use its DDH oracle $\mathcal{O}_{DDH}$ to provide simulations of $\mathcal{A}$'s oracles as follows:

- $\mathsf{NewSession}(i, \rho)$ – $\mathcal{B}$ starts a new session for i. All participants may have a total of n_s sessions.
- $\mathsf{Send}(\pi^s_{i,j}, m)$ –
  - For $i \in C$ (and $\rho = \texttt{initiator}$), select at random $\alpha^s_{i,j} \xleftarrow{r} \mathbb{F}_q$ to create message $A = \alpha^s_{i,j} Q_i$.
  - For $i \in T$ (or $i \in I$ and $\rho = \texttt{responder}$), select at random $\beta^s_{i,j} \xleftarrow{r} \mathbb{F}_q$ to create message $E = \beta^s_{i,j} bP$.

  This will result in a shared key $\kappa = H(\alpha^s_{i^*,j}\beta^s_{i^*,j} abP)$ for participant i^* in some session s with partner $j \in I \cup T$.
- $\mathsf{Corrupt}(i, d')$ –
  - For $i \in C$, then return d_i and replace it with d' unless $i = i^*$ in which case abort
  - For $i \in T$, return $\perp$.
- $\mathsf{Reveal}(\pi^s_{i,j})$ – To answer Reveal queries, $\mathcal{B}$ will maintain a Guess session key list (G-List). Each element on the G-List is a tuple of the form (τ, i, j, κ_R). Queries are answered as follows:

- First $\mathcal{B}$ checks the G-list and if there is an entry for i, j then $\mathcal{B}$ outputs the corresponding κ_R.
- If not then $\mathcal{B}$ checks whether the H-list (see below) contains an (M, h, st_h) with $\mathcal{O}_{DDH}(\alpha^s_{i,j}Q_i, \beta^s_{i,j}bP, M) = 1$. If it does then $\mathcal{B}$ sets $st_h = \{i, j\}$ and adds to G-list (τ, i, j, h).
- Otherwise $\mathcal{B}$ returns a randomly chosen key.

– $\mathsf{hash}(M)$ – To answer hash queries, $\mathcal{B}$ maintains an H-List containing tuples of the form (M, h, st_h). Queries are answered as followed:
- $\mathcal{B}$ first checks whether M is on the H-list. If it is, then $\mathcal{B}$ outputs h.
- If not then $\mathcal{B}$ must check whether $\mathsf{hash}(M)$ is already an valid entry on the G-list for some pair of participants (i, j) by calling its $\mathcal{O}_{DDH}$.
- If it is a valid entry for some pair of participants (i, j) then $\mathcal{B}$ returns the corresponding κ_R from the G-list and adds $(M, \kappa_R, \{i, j\})$ to the H-list.
- Otherwise $\mathcal{B}$ chooses a random hash h and adds (M, h, st_h) to list.

Eventually, $\mathcal{A}$ will output its guess $\pi^* = \pi^s_{i,j}$ and κ^*, The probability that $\mathcal{A}$ chooses $i = i^*$ is $1/n_C$ Note that in this case i^* will not have been corrupted so the simulation has been perfect. At this point $\mathcal{B}$ searches the H-list for the entry $(M^*, \kappa^*, st^*_\kappa)$ corresponding to κ^*, using $\mathcal{O}_{DDH}$ to verify that the entry corresponds to i^*, j. If this entry does not exist then $\mathcal{A}$ must have output a random guess for the key, in which case his probability of success is at best $1/H$, where H is the size of the output to the function hash. Since we assume $\mathcal{A}$ to be a winning adversary with probability $(1 - 1/H)$ $\mathcal{A}$ queries hash such that his guess is on the H-list. If it is on the list then $\mathcal{B}$ calculates the solution to the gap-DH problem as $(1/\alpha^s_{i^*,j}\beta^s_{i^*,j})M^*$.

D.2 One-sided Entity Authentication

We now turn to proving the various properties in our main theorem. We start with one-sided entity authentication:

Lemma 2. *If there is an adversary $\mathcal{A}$ against the no plaintext matching property of the protocol then there is an adversary $\mathcal{B}$ such that*

$$\Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{A})] \leq \mathbf{Adv}^{\mathsf{eufcma}}_{\mathrm{cert}}(\mathcal{B}).$$

Proof. Assume we have an adversary $\mathcal{A}$ against the $\mathsf{OS\text{-}NoMatchPlntxt}$ of Π we shall use this to construct an adversary $\mathcal{B}$ against the EUF-CMA property of the signature scheme that the card issuer used to sign the certificate.

Algorithm $\mathcal{B}$ begins by setting up n_C registered participants and n_T unregistered participants by calling its sign oracle to generate the certificates for the registered participants. All other keys necessary $\mathcal{B}$ shall generate itself and model $\mathcal{A}$'s queries appropriately.

For $\mathcal{B}$ to win there must exist one oracle $\Pi^s_{i,j}$ that accepts but has had no plaintext matching application conversation with $\Pi^t_{j,i}$. For $\Pi^s_{i,j}$ to accept the final confirmation message received after decryption, namely (a', Q', cert') must verify correctly, (i.e. $A = a'Q'$ and the signature on cert' verifies correctly) and must not have been output by $\Pi^t_{j,i}$.

If $\mathcal{A}$ succeeds in getting $\Pi^s_{i,j}$ to accept without a plaintext matching conversation then either:

– He must have forged a new cert for a pair (a, Q) output by $\Pi^t_{j,i}$.
– Or forged cert on a completely new Q.

In either case $\mathcal{B}$ simply forwards the forged cert to his verify oracle and wins the EUF-CMA game.

D.3 Message Authentication

We now turn to the message authentication property:

Lemma 3. *If π is a key-secure key-agreement protocol, $\mathsf{AE} = (\mathsf{enc}, \mathsf{dec})$ is an int-ctxt secure authenticated encryption scheme and plaintext matching conversations holds, then Π is secure in the sense of* OS-auth. *In particular if there is an adversary $\mathcal{A}$ against the* OS-auth *property then there are adversaries $\mathcal{B}, \mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathbf{Adv}^{\mathsf{OS\text{-}auth}}_{\Pi}(\mathcal{A}) \leq n_S \cdot (n_C + n_T) \cdot \mathbf{Adv}^{\mathsf{intsfctxt}}_{\mathsf{AE}}(\mathcal{D}) + n_C \cdot (1 - 1/H) \cdot \mathbf{Adv}^{\mathsf{Gap-DH}}_{E(\mathbb{F}_p)}(\mathcal{C}) + \mathbf{Adv}^{\mathsf{eufcma}}_{\mathrm{cert}}(\mathcal{B}).$$

where n_C is the number of cards in the system, n_T the number of terminals and n_S the number of sessions.

Proof. We shall prove this result via a sequence of games. Let $\mathcal{A}$ be adversary attacking Π in the sense of auth.
Game 0: This game is identical to $\mathbf{Exec}^{\mathsf{OS\text{-}auth}}_{\Pi}(\mathcal{A})$.

$$\Pr[\mathsf{Game0} \Rightarrow 1] = \mathbf{Adv}^{\mathsf{OS\text{-}auth}}_{\Pi}(\mathcal{A})$$

Game 1: This proceeds identically to the previous game but aborts if no plaintext matching occurs for some terminal oracle $\Pi^s_{i,j}$. It is easy to see that

$$\Pr[\mathsf{Game0} \Rightarrow 1] \leq \Pr[\mathsf{Game1} \Rightarrow 1] + \Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{B}')]$$

Game 2: This proceeds identically to the previous game but aborts if $\mathcal{A}$ makes a query to H which reveals the key for an oracle $\Pi^s_{i,j}$. Again it is easy to see that

$$\Pr[\mathsf{Game1} \Rightarrow 1] \leq \Pr[\mathsf{Game2} \Rightarrow 1] + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}')$$

Game 3: The challenger now selects at random an oracle $\Pi^{s^*}_{i^*,j^*}$. The game aborts and returns random b' if $\mathsf{Prefix}(L^{app|rec}_{i,j,s}, L^{app|sen}_{j,i,t}) = 0$ for $(i, j, s) \neq (i^*, j^*, s^*)$.

$$\Pr[\mathsf{Game2} \Rightarrow 1] \leq n_S \cdot (n_C + n_T) \cdot \Pr[\mathsf{Game3} \Rightarrow 1]$$

It remains to study the probability that $\mathcal{A}$ wins ($\mathsf{Game3} \Rightarrow 1$). We shall use $\mathcal{A}$ in $\mathsf{Game3} \Rightarrow 1$ to construct a new adversary $\mathcal{D}$ against the INT-CTXT security of AE. What we effectively do is set the output of the random oracle H for the key corresponding to $\Pi^{s^*}_{i^*,j^*}$ to be the key chosen at random for the INT-CTXT experiment. When $\mathcal{A}$ makes a Send query with $\mathsf{op} = \mathsf{Encrypt}$ or $\mathsf{Decrypt}$ these are forward to $\mathcal{D}$'s enc and dec oracles respectively. All other queries $\mathsf{NewSession}, \mathsf{Reveal}, \mathsf{Corrupt}$ and Send when $\mathsf{op} = \emptyset$ are simulated internally by $\mathcal{D}$ selecting appropriate randomness. Since $\mathcal{A}$ does not make any reveal queries or hash queries corresponding to the key of $\Pi^{s^*}_{i^*,j^*}$ the simulation shall remain perfect. If $\mathcal{A}$ wins the auth game then $\mathsf{Prefix}(L^{app|rec}_{i^*,j^*,s^*}, L^{app|sen}_{j^*,i^*,t^*}) = 0$ and therefore $\mathcal{A}$ has output a ciphertext forgery which allows $\mathcal{D}$ to win the INT-CTXT game. We therefore have,

$$\Pr[\mathsf{Game3} \Rightarrow 1] \leq \mathbf{Adv}^{\mathsf{intsfctxt}}_{\mathsf{AE}}(\mathcal{D})$$

Combining all of the above we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A}) &= \Pr[\mathsf{Game0} \Rightarrow 1] \\
&\leq \Pr[\mathsf{Game1} \Rightarrow 1] + \Pr[\mathsf{OS\text{-}NoMatchPIntxt}(\mathcal{B}')] \\
&\leq \Pr[\mathsf{Game2} \Rightarrow 1] + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}') + \Pr[\mathsf{OS\text{-}NoMatchPIntxt}(\mathcal{B}')] \\
&\leq n_S \cdot (n_C + n_T) \cdot \Pr[\mathsf{Game3} \Rightarrow 1] + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}') \\
&\quad + \Pr[\mathsf{OS\text{-}NoMatchPIntxt}(\mathcal{B}')] \\
&\leq n_S \cdot (n_C + n_T) \cdot \mathbf{Adv}^{\mathsf{intsfctxt}}_{\mathsf{AE}}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}') + \Pr[\mathsf{OS\text{-}NoMatchPIntxt}(\mathcal{B}')]
\end{aligned}$$

With the final result following from applying Lemmas 1 and 2.

D.4 Message Privacy

We now turn to the message privacy property:

Lemma 4. *If π is a key-secure key-agreement protocol, $\mathsf{AE} = (\mathsf{enc}, \mathsf{dec})$ is an ind-cca secure authenticated-encryption scheme and plaintext matching conversations holds. Then Π is secure in the sense of* OS-priv*, i.e. any adversary $\mathcal{A}$ against the* OS-priv *property can be turned into adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathbf{Adv}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A}) \leq n_S \cdot (n_C + n_T) \cdot \mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{D}) + n_C \cdot (1 - 1/H) \cdot \mathbf{Adv}^{\mathsf{Gap-DH}}_{E(\mathbb{F}_p)}(\mathcal{C}) + \mathbf{Adv}^{\mathsf{eufcma}}_{\mathrm{cert}}(\mathcal{B}).$$

Proof. We shall prove this result via a sequence of games. Let $\mathcal{A}$ be adversary attacking Π in the sense of priv.

Game 0: This game is identical to $\mathbf{Exec}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A})$.

$$\Pr[\mathsf{Game0} \Rightarrow 1] - \frac{1}{2} = \mathbf{Adv}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A})$$

Game 1: This proceeds identically to the previous game but aborts if no plaintext matching occurs for some terminal oracle $\Pi^s_{i,j}$. It is easy to see that we can define an algorithm $\mathcal{B}'$ such that

$$\Pr[\mathsf{Game0} \Rightarrow 1] \leq \Pr[\mathsf{Game1} \Rightarrow 1] + \Pr[\mathsf{OS\text{-}NoMatchPIntxt}(\mathcal{B}')]$$

Game 2: This proceeds identically to the previous game but aborts if $\mathcal{A}$ makes a query to H which reveals the key for an oracle $\Pi^s_{i,j}$. Again it is easy to see that

$$\Pr[\mathsf{Game1} \Rightarrow 1] \leq \Pr[\mathsf{Game2} \Rightarrow 1] + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}')$$

Game 3: The challenger now selects at random an oracle $\Pi^{s^*}_{i^*,j^*}$. The game aborts if the attacker outputs (i, j, s, b') such that $(i, j, s) \neq (i^*, j^*, s^*)$, the game will instead return a random bit.

$$\Pr[\mathsf{Game2} \Rightarrow 1] - \frac{1}{2} \leq n_S \cdot (n_C + n_T) \cdot \left(\Pr[\mathsf{Game3} \Rightarrow 1] - \frac{1}{2} \right)$$

It remains to study the probability that $\mathcal{A}$ wins ($\mathsf{Game3} \Rightarrow 1$). We shall use $\mathcal{A}$ in $\mathsf{Game3} \Rightarrow 1$ to construct a new adversary $\mathcal{D}$ against the IND-CCA security of AE. What we effectively do is set the output of the random oracle H for the key corresponding to $\Pi^{s^*}_{i^*,j^*}$ to be the key chosen at random for the IND-CCA experiment. When $\mathcal{A}$ makes a Send query with $\mathsf{op} = \mathsf{Encrypt}$ or Decrypt these are forward to $\mathcal{D}$'s enc

and dec oracles respectively. All other queries NewSession, Reveal, Corrupt and Send when $\mathtt{op} = \emptyset$ are simulated internally by $\mathcal{D}$ selecting appropriate randomness. Since $\mathcal{A}$ does not make any reveal queries or hash queries corresponding to the key of $\Pi^{s^*}_{i^*,j^*}$ the simulation shall remain perfect. When $\mathcal{A}$ outputs its guess (i^*, j^*, s^*, b'), $\mathcal{D}$ shall forward b' as its guess. We therefore have,

$$\Pr[\mathsf{Game3} \Rightarrow 1] - \frac{1}{2} \leq \mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{D})$$

Combining all of the above, we yield:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{OS\text{-}priv}}_{\Pi}(\mathcal{A}) &= \Pr[\mathsf{Game0} \Rightarrow 1] - \frac{1}{2} \\
&\leq \Pr[\mathsf{Game1} \Rightarrow 1] - \frac{1}{2} + \Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{B}')] \\
&\leq \Pr[\mathsf{Game2} \Rightarrow 1] - \frac{1}{2} + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}') + \Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{B}')] \\
&\leq n_S(n_C + n_T)\left(\Pr[\mathsf{Game3} \Rightarrow 1] - \frac{1}{2}\right) + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}') \\
&\quad + \Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{B}')] \\
&\leq n_S(n_C + n_T)\mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{C}') + \Pr[\mathsf{OS\text{-}NoMatchPlntxt}(\mathcal{B}')]
\end{aligned}$$

Again the final result follows from applying Lemmas 1 and 2.

E Proof of Theorem 2

Proof. We shall prove this result via a sequence of games. Let $\mathcal{A}$ be adversary attacking Π in the sense of unlink.

Game 0: This game is identical to $\mathbf{Exec}^{\mathsf{unlink}}_{\Pi}(\mathcal{A})$.

$$\Pr[\mathsf{Game0} \Rightarrow 1] - \frac{1}{2} = \mathbf{Adv}^{\mathsf{unlink}}_{\Pi}(\mathcal{A})$$

Game 1: The challenger now selects at random i^*_0 and i^*_1. The game aborts and returns random b' if $\mathcal{A}$ does not output $i_0 = i^*_0$ and $i_1 = i^*_1$. We obtain

$$\Pr[\mathsf{Game0} \Rightarrow 1] - \frac{1}{2} \leq n_C^2 \cdot \left(\Pr[\mathsf{Game1} \Rightarrow 1] - \frac{1}{2}\right)$$

Game 2: This proceeds identically to the previous game but aborts if $\mathcal{A}$ makes a query to H which reveals the key for the oracle $\mathcal{O}$. We obtain

$$\Pr[\mathsf{Game1} \Rightarrow 1] \leq \Pr[\mathsf{Game2} \Rightarrow 1] + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{B})$$

Game 3: This proceeds identically to the previous game except that whenever Send is called with $\mathcal{O}$ and $\mathtt{op} = \mathtt{encrypt}$ then the challenger replaces m with a random message which it then encrypts. Again it is easy to see that we obtain

$$\Pr[\mathsf{Game2} \Rightarrow 1] \leq \Pr[\mathsf{Game3} \Rightarrow 1] + \mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{C})$$

It remains to study the probability that $\mathcal{A}$ wins ($\mathsf{Game3} \Rightarrow 1$). We shall use $\mathcal{A}$ to construct a new adversary $\mathcal{D}$ against the SDDL problem. The algorithm $\mathcal{D}$ shall simulate the queries for $\mathcal{A}$ using its challenge P, X_0, X_1, rX_{i^*}, where $i^* = 0$ or 1. Upon starting, $\mathcal{D}$ initialises n_C cards by choosing secret keys d_i at random and calculating the public keys $Q_i = d_i P$; except for cards with indexes i_0^* and i_1^*, where it sets the public keys to $Q_{i_0^*} = X_0$ and $Q_{i_1^*} = X_1$ respectively. When $\mathcal{A}$ makes Send queries these are modelled appropriately by $\mathcal{D}$:

- When $\mathtt{op} = \emptyset$ then $\mathcal{D}$ chooses a or e at random and returns aQ_i or eP as appropriate. If the card has already received eP then $\mathcal{D}$ returns the $\mathsf{enc}_\kappa(cert_i, a, Q_i)$ where $\kappa = H(ad_i eP)$.
- When $\mathtt{op} = \mathtt{encrypt}$ or $\mathtt{decrypt}$ then $\mathcal{D}$ performs the necessary encryption or decryption using the appropriate key ($\kappa = H(ad_i eP)$).

When $\mathcal{A}$ performs a Send query with input oracle $\mathcal{O}$ then the first key exchange message sent shall be the challenge message, i.e. $A = rX_{i^*}$. Following this $\mathcal{D}$ will calculate all other responses appropriately. The key established by $\mathcal{O}$ shall be $H(erX_{i^*})$, for some randomly chosen e.

If $\mathcal{A}$ makes a Reveal or $\mathsf{Corrupt}$ then $\mathcal{D}$ returns the appropriate key material to $\mathcal{A}$ ($\mathcal{A}$ is not permitted to make such queries for $\mathcal{O}$).

Eventually $\mathcal{A}$ outputs its guess b'. If $\mathcal{A}$ wins then he has successfully determined that $\mathcal{O}$ is associated to $i_{b'}^*$. Since the encryption $\mathsf{enc}_\kappa(cert_i, a, Q_i)$ was replaced by a random ciphertext in Game 3 this means that $\mathcal{A}$ must have determined b' from the first key exchange message of $\mathcal{O}$, $A = rQ_{i_{b'}^*} = rX_{b'}$. $\mathcal{A}$ has therefore solved $\mathcal{D}$'s challenge and $\mathcal{D}$ can output b' as his (winning) guess. We therefore have:

$$\Pr[\mathsf{Game3} \Rightarrow 1] - \frac{1}{2} \leq \mathbf{Adv}^{\mathsf{SDDL}}_{E(\mathbb{F}_p)}(\mathcal{D})$$

Combining all of the above:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{unlink}}_{\Pi}(\mathcal{A}) &= \Pr[\mathsf{Game0} \Rightarrow 1] - \frac{1}{2} \\
&\leq n_C^2 \cdot \left(\Pr[\mathsf{Game1} \Rightarrow 1] - \frac{1}{2} \right) \\
&\leq n_C^2 \cdot \left(\Pr[\mathsf{Game2} \Rightarrow 1] - \frac{1}{2} + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{B}) \right) \\
&\leq n_C^2 \cdot \left(\Pr[\mathsf{Game3} \Rightarrow 1] - \frac{1}{2} + \mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{C}) + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{B}) \right) \\
&\leq n_C^2 \cdot \left(\mathbf{Adv}^{\mathsf{SDDL}}_{E(\mathbb{F}_p)}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{indsfcca}}_{\mathsf{AE}}(\mathcal{C}) + \mathbf{Adv}^{\mathsf{wKSec}}_{\pi}(\mathcal{B}) \right)
\end{aligned}$$