

Secure Key Exchange and Sessions Without Credentials

Ran Canetti^{*}, Vladimir Kolesnikov^{**}, Charles Rackoff^{***}, and Yevgeniy Vahlis[†]

Abstract. Secure communication is a fundamental cryptographic primitive. Typically, security is achieved by relying on an existing credential infrastructure, such as a PKI or passwords, for identifying the end points to each other. But what can be obtained when no such credential infrastructure is available?

Clearly, when there is no pre-existing credential infrastructure, an adversary can mount successful “man in the middle” attacks by modifying the communication between the legitimate endpoints. Still, we show that not all is lost, as long as the adversary’s control over the communication is not complete: We present relatively efficient key exchange and secure session protocols that provide the full guarantee of secure communication *as long as the adversary fails to intercept even a single message* between the legitimate endpoints.

To obtain this guarantee we strengthen the notion of key exchange to require that the keys exchanged in any two sessions are independent of each other as long as each session has at least one honest endpoint, *even if both sessions has an adversarial endpoint*. We call this notion **credential-free key exchange**. We then strengthen the existing notion of secure session protocols to provide the above guarantee given a CFKE (existing definitions and constructions are insufficient for this purpose). We provide two alternative definitions and constructions of CFKE, a game-based one with a construction in the RO model, and a UC one with a construction in the CRS model.

^{*} Tel-Aviv University, canetti@tau.ac.il

^{**} Bell Labs, kolesnikov@research.bell-labs.com

^{***} University of Toronto, rackoff@cs.toronto.edu

[†] AT&T Labs SRC, evahlis@gmail.com

1 Introduction

Secure communication over adversary-controlled channels is one of the most widely and frequently used achievements of cryptography. The standard approach to secure communication involves two steps. First, the two conversing parties A and B securely establish a *session key*, and second, they use the key to encrypt and authenticate the exchanged messages. The first step, *key exchange* (KE), ensures that only authorized players are able to successfully compute the keys. This guarantee holds even if the adversary is capable of complete control of the channel, including arbitrary message observation, alteration and scheduling.

Traditionally, the ability of communicating parties to authenticate themselves almost always requires possession of secrets, and corresponding authorization architectures and protocols employ some kind of *infrastructure*, which manages these secrets. Common examples include Public Key Infrastructure (PKI), shared long term private keys, and human memorizable passwords.

In this work, we focus on efficiently achieving a strong security guarantee for the problems of key exchange and secure sessions in the credential-free setting.

Practical need for authentication without credentials. While trusted infrastructure aids greatly in certifying the identities, capabilities or permissions of communicating parties, its heavy cost is not justified in many non-security-critical applications. In some cases, there may not be an authority who is qualified or trusted to operate the infrastructure and issue credentials. Some applications may use infrastructure opportunistically and use PKI when available, but may fall back on weak authentication without credentials. In yet other scenarios, such as ad-hoc peer-to-peer information sharing networks, it is not required to associate a network entity with a real-life entity, but rather there is a need to ensure the persistence of the connection.

The practical importance of the problem has led to a rich body of network security research. Several techniques of *weak authentication* have emerged, were standardized, and successfully widely deployed. Our work is a more formal approach that achieves much stronger security.

1.1 The Setting and Our Contributions

In our setting, two parties A and B have decided to have a communication session over an insecure channel. A and B do not share any information other than which channel to use for communication, and possibly a global public reference string. The decision itself may have been made in an insecure way; for example, A may have really been invited by an adversary Adv rather than by B . As there is no infrastructure to support authentication, Adv may falsely claim any identity. The channel A and B will use for their session may also be controlled by Adv , who can read from it and write to it at will. Without credentials, we have no hope of preventing Adv falsely taking another player's identity or playing man-in-the-middle (MIM).

Motivated by today's networking architectures (e.g., the difficulty of the adversary to always be an active MIM), we present a candidate for what we believe is the "next best" achievable security guarantee in this setting. Namely, we require that *the adversary must remain continuously active on the channel throughout the entire session to avoid detection*.

We formalize this by proposing new definitions of KE and secure sessions, which, when combined, guarantee the above property (see Sect. 5.1). Our definitional approach is as follows. In the KE preceding the communication, either the exchanged key is hidden from Adv , or, if the adversary is playing MIM between A and B , then A and B will output keys that are random and independent from each other, and the adversary's view contains no secrets about these keys. We model this by requiring the view of the adversary to be simulatable given the outputs of the parties. We identify and formalize

a security property of secure session protocols, which we call MIM-integrity, which ensures that the MIM-adversary must remain continuously active on the channel throughout the entire session to avoid detection. We show that combining above KE and secure sessions guarantees secure communication in the above sense. In this work we:

1. Present definitions of secure credential-free key exchange (CFKE) in the standalone and universally composable (UC) [Can00] settings. Our definitions guarantee that if two parties participate in a CFKE protocol, they either output the same key which is completely hidden from the adversary, or output two keys that are uniformly random given the transcript of the view of the adversary.
2. Present simple definitions of secure credential-free secure sessions (CFSS), which, in addition to the standard security properties, guarantee that MIM must be continuously active to avoid detection.
3. We show that composing any CFKE and CFSS results in secure communication in the credential-free setting. We also show that, in contrast, session protocols, even those standardized by the Internet bodies, are *insecure* in our setting.
4. Describe a construction of a CFKE protocol in the standalone model with a global common reference string (CRS), and prove its security in the standard model. We then show how to construct a secure session with a new integrity property (MIM-integrity) that guarantees that an adversary must modify every message in transit from A to B or be detected.
5. Analyze two existing KE protocols: we show that the well known hashed Diffie-Hellman protocol satisfies a variant of our standalone definition in the random oracle model. We additionally compare our standalone protocol to the instantiation of our UC definition using the construction of [BCL⁺05], and show that it provides a useful tradeoff in settings with high latency, where low round complexity is crucial.

1.2 Intuition for Our Constructions

CFKE. The key difference between a CFKE protocol and an unauthenticated key exchange (such as the basic Diffie-Hellman protocol) is that neither party should be able to influence the outcome too much. This is necessary to achieve independence of keys in the man-in-the-middle scenario.

A naive CFKE approach may be to use a coin toss protocol as a black box, and simply encrypt its messages. However, there is no key infrastructure that would allow A and B to encrypt! Indeed, the whole purpose of CFKE is to attempt to establish such an infrastructure. Moreover, simply exchanging public keys to be used for encryption won't work since the adversary may be able to replace (or modify) the public keys on the way, stripping the coin toss protocol from any privacy. This leads us to the following intuitive observation: any cryptographic keys that are sent across the channel and are used for the security of KE, must be strongly tied to the randomness used to generate the final key.

Overview of the construction. Our credential-free key exchange protocol is symmetric, and consists of two rounds where A and B simultaneously send messages to each other. It proceeds as follows:

1. A and B generate and announce public keys, and commit to the public keys *together with* random nonces. Here the commitment must have strong non-malleability properties (which we identify and formally define).
2. Using the public keys announced in the first round, A and B send to each other encrypted decommitment strings for the commitments they announced in the first round.
3. A and B use the decommitment strings to obtain each other's nonces, and output their exclusive-OR.

Relying on the non-malleability property of the commitment scheme, we get that no adversary can modify either A 's public key or her nonce, without changing the other (similarly for B). Moreover, since the nonce is hidden by the commitment, the adversary is unable to commit to a related nonce, if she decides to modify the public key. On the other hand, if she leaves the public key untouched, she will not be able to learn B 's decommitment when he sends it in the next round.

Using CFKE for communication. CFKE can be used to construct secure session protocols with security against MIM-adversaries. We show that the following is a secure credential-free secure session (CFSS): a secure session protocol in the traditional sense, where additionally the sender simply attaches to each message, in the clear, a dedicated part of the secret key. That is, if A 's key is $K_A = (K_{A,1}, K_{A,2})$, then $K_{A,1}$ is used for the underlying secure session protocol, and $K_{A,2}$ is attached as an authenticator to each message. Now, when B receives a message from A with an attached authenticator $K_{A,2}$, he checks if $K_{A,2} = K_{B,2}$, and aborts if the check fails. It is easy to see that any adversary that does not change the authenticator on each message will be caught as soon as a single message is transmitted directly.

1.3 Discussion

Password-authenticated key exchange (PAKE) [BM92,KOY01,GL01,GL03,CHK⁺05,KV11,CDSVW12], when executed with fixed publicly known password, achieves guarantees somewhat similar to those of CFKE, and at the first glance may seem sufficient for our task. There are, however, important differences between CFKE and PAKE, which render PAKE inapplicable to our setting.

Firstly, the definitional approach to PAKE fundamentally differs from our approach: a successful Adv in PAKE may be able to fix the output key to an arbitrary value, which would render such KE useless for CFSS (indeed, MIM Adv can set the two players' key to be equal and known to him, and thus break the CFSS protection).

Beyond just definitions, many natural PAKE protocols in fact do allow a player to set the session key (and, to our knowledge, no protocols prove otherwise). Indeed, such a PAKE can be easily constructed from any secure PAKE by adding a round where the successfully authenticated players (now presumed honest by PAKE definitions) are allowed to set the session key to anything of their choice.

More importantly, many existing protocols from the literature, e.g., of Canetti et al. [CDSVW12] have the above feature. We stress that it is a natural property of their approach; provably avoiding it would complicate their construction and would require a new definition and a proof. In [CDSVW12], UC-secure PAKE is built from Oblivious Transfer (OT), roughly as follows: the two players run several OT instances, where the secrets are randomly chosen strings, and the password defines the selection bits. The session key of OT receiver is the XOR of the received secrets, and OT sender's key is the XOR of the secrets corresponding to his password. It is easy to see that in the above PAKE, the OT sender can set the session key to any string of his choice. We note that the above PAKE OT idea is the basis for two constructions of [CDSVW12], and in both of them the property of OT sender being able to set the key is preserved.

Practical Impact of MIM-Integrity. Despite seeming simplicity, MIM-integrity is a subtle concept and can be violated by seemingly secure natural protocols. Consider a session protocol where players periodically refresh the session key. Typically, one player chooses a nonce n , and each player updates his key sk to $\text{PRF}_{sk}(n)$. One prominent example of such a protocol is in EAP-TLS [ALE09]. This refresh is a frequently employed "best security practice", which aims to limit the amount of ciphertext an adversary can collect to attack the underlying encryption. However, CFSS using above security heuristic is completely insecure, as, since AES is invertible, MIM Adv can choose nonces

n_A, n_B which would set the refreshed keys $K_A = K_B$, allowing the adversary to withdraw and only occasionally interfere with communication when needed.

Open directions. The main open direction left by our work is to design new secure session protocols and notions of security against MIM adversaries that rely on other (possibly weaker or incomparable) properties of networks in practice. For example, one may be able to design secure sessions that always detect a MIM adversary assuming network delay, or that messages are sent in pieces and that an adversary is unable to predict future content (this idea was used in the Interlock protocol [RS84]). We believe that our notion (and constructions) of CFKE can be used as a formal basis for such future explorations.

1.4 Related Work

The problem of unauthenticated secure communication over insecure channels traces back to the work of Dolev, Dwork, and Naor in [DDN91]. They introduced the non-malleability guarantee: any adversary controlling the channel between two parties either remains essentially passive, or is forced to run two independent instances of the protocol, one with each honest party. The study of non-malleability has led to a rich body of research that can be roughly classified according to two types of constructions: constructions of specific non-malleable primitives such as encryption, zero-knowledge proof systems, commitments, etc; and non-malleable “meta-protocols” that are used to establish per-session infrastructures. These include non-malleable coin tossing [Bar02], establishing a public key infrastructure [BCL⁺05], and a shared secret key infrastructure [CCGS10]. Such protocols can then provide setup for the protocols that need it.

Barak [Bar02] defines and constructs non-malleable coin tossing protocols, where two parties wish to agree on an almost unbiased *public* random string in the presence of MIM adversary. This coin tossing guarantees that an adversary must either allow the players to output the same random string, or cause them to output two separate and independently generated random strings. The protocols of [Bar02] work without *any* infrastructure, but provide no privacy guarantee (the outcome of the coin toss is always known to the adversary).

In [BCL⁺05] Barak *et al.* define split functionalities – a variant of ideal functionalities of the UC framework. Split functionalities allow an adversary to partition the set of honest parties P into disjoint “authentication sets”. The idea is that all parties within each set $H_i \subseteq P$ have successfully established an authenticated session. The adversary is then unable to impersonate any party in H_i to any other party in H_i , but can impersonate any party in $P \setminus H_i$ to any party in H_i . Since the adversary determines the authenticated sets, this allows her, for example, to set all authenticated sets to be singletons, in which case no authenticated communication between honest parties is possible. This indeed seems unavoidable since the adversary has complete control over the communication channels. Camenisch *et al.* [CCGS10] define, among other things, split key exchange and give a construction based on the decisional Diffie-Hellman assumption.

Our non-malleable KE can be built from split functionalities and coin toss. Indeed, coin-toss whose output is hidden from Adv is easily achieved assuming secure channels (e.g., Blum’s protocol with a UC-commitment suffices). Then, applying the compilers of [BCL⁺05] or [CCGS10] to such secret coin toss functionality guarantees that either key is hidden from Adv (Adv chose to interact with a single ideal functionality that issues keys secretly to the two honest parties), or that Adv knows both independent keys (Adv interacts with two separate functionalities, playing the role of an honest party in each one). Using the recent commitment protocol of Lindell [Lin11b], the resulting KE protocol requires seven rounds of communication, and sending of a constant number of group elements.

2 Preliminaries

Notation. We write PPT to denote Probabilistic Polynomial Time. When we wish to fix the random bits of a PPT algorithm M to a particular value, we write $M(x; r)$ to denote running M on input x and randomness r . We write $time_n(M)$ to denote the running time of algorithm M on security parameter n . We use $x \in_R S$ to denote the fact that x is sampled according to a distribution S . Similarly, when describing an algorithm we may write $x \leftarrow_R S$ to denote the action of sampling an element from S and storing it in a variable x .

Discrete Logarithm Assumption. Let \mathcal{G} be a probabilistic group generator such that $\mathbb{G} \leftarrow_R \mathcal{G}(1^n)$ is a group of order p where p is a prime of length about n bits. The Discrete Logarithm assumption for \mathcal{G} is that given \mathbb{G}, g, g^x , where $\mathbb{G} \leftarrow_R \mathcal{G}(1^n), g \in_R \mathbb{G}, x \in_R \mathbb{Z}_p$, it is infeasible to find x .

Non-Malleable Public Key Encryption (PKE). Let $\mathcal{PKC} = \langle \text{KeyGen}, \text{Enc}, \text{Dec} \rangle$ be a public key encryption scheme. We say that \mathcal{PKC} is non-malleable if no adversary can distinguish the encryptions of two messages of her choice, even if she is allowed to make a single decryption query after seeing the challenge ciphertext C_* . The decryption query is of course restricted to all strings that are not equal to C_* .

Target Collision Resistant Hash Functions. A family $\mathcal{H} = \{H_k\}_{k \in \{0,1\}^*}$ of hash functions is target collision resistant if no efficient adversary can win the following game: (i) the adversary selects a target input x ; (ii) a random key $k \in_R \{0,1\}^n$ is chosen; (iii) the adversary is given k and must output another input y such that $H_k(x) = H_k(y)$.

3 Definition of Secure Key Exchange Without Credentials

We start with a syntactic definition of a credential-free key exchange (CFKE) protocol. A CFKE protocol is a triple $\mathcal{CFKE} = \langle \text{KEInit}, \text{KE}_A, \text{KE}_B \rangle$ where KEInit takes as input a security parameter 1^n and outputs a common reference string PUB . The protocol itself consists of the actions performed by a role-A party, specified by KE_A , and a role-B party whose actions are specified by KE_B (the roles are assigned to break the symmetry, e.g. to determine who moves first). The pair $\text{KE} = \langle \text{KE}_A, \text{KE}_B \rangle$ is a two party protocol in the common reference string model. We shall use KE to discuss the protocol as a whole, and distinguish between KE_A and KE_B when such a distinction is warranted.

Adversary for a CFKE protocol is a triple of PPT algorithms $\text{Adv} = (A_{ke}, A_{dist}, A_{mal})$. To define security of CFKE we describe three experiments where the first experiment models the interaction of Adv with the protocol KE , and the other experiments capture *privacy* and *non-malleability* properties.

Security Experiments. We start with the description of the experiment ExpCFKE which defines the interaction of the adversary with the protocol: let \mathcal{KE} be a CFKE and let A_{ke} be a PPT algorithm.

Experiment $\text{ExpCFKE}(1^n, \mathcal{KE}, A_{ke})$

1. $\text{KEInit}(1^n)$ is run to obtain public parameters PUB , which are given to A_{ke} .
2. The key exchange protocol is run between two parties A and B , where A acts according to KE_A and B acts according to KE_B . All the communication is routed through A_{ke} . During this process A_{ke} can inject, delete, and modify messages between the two parties at will.
3. After KE concludes, let K_A^{out} and K_B^{out} be the outputs of the protocol, and let $view(A_{ke})$ be the view of A_{ke} during its execution. The view consists of the randomness of A_{ke} , and all the (potentially modified) messages exchanged between A and B during the execution of the protocol.
4. The outcome of the experiment is the tuple $(K_A^{out}, K_B^{out}, view(A_{ke}))$.

Intuitively, we wish to achieve the following security guarantee: either the two parties agree on a key and the adversary knows nothing about it, or the adversary is forced to perform two independent key exchanges (one with each party) resulting in the parties outputting independently random keys. We capture this intuition by describing two security experiments for CFKE: privacy and non-malleability.

The first experiment ExpCFKEInd requires the adversary to distinguish a key agreed upon by the two parties in the protocol from a random key. If the two parties do not agree on a key (i.e. $K_A^{\text{out}} \neq K_B^{\text{out}}$) then the adversary automatically loses in the privacy experiment. This is enforced by setting the outcome of the experiment by flipping an unbiased coin. Let $b \in \{0, 1\}$, the privacy experiment is defined as follows:

Experiment $\text{ExpCFKEInd}(1^n, \mathcal{KE}, A_{ke}, A_{ind}, b)$

1. Run experiment $\text{ExpCFKE}(1^n, \mathcal{KE}, A_{ke})$ to obtain $(K_A^{\text{out}}, K_B^{\text{out}}, \text{view}(A_{ke}))$.
2. If $K_A^{\text{out}} \neq K_B^{\text{out}}$, flip an unbiased coin $b' \in_{\mathcal{R}} \{0, 1\}$ and set the outcome of the experiment to b' .
3. Else, let $K_0 = K_A^{\text{out}}$, $K_1 \in_{\mathcal{R}} \{0, 1\}^n$. Let $b' \leftarrow_{\mathcal{R}} A_{ind}(\text{view}(A_{ke}), K_b)$. The outcome of the experiment is b' .

Definition 1. Let $\mathcal{KE} = (\text{KEInit}, \text{KE})$ be a credential-free key exchange. We say that \mathcal{KE} is private if for every CFKE adversary $\text{Adv} = (A_{ke}, A_{dist}, A_{mal})$, there exists a negligible function $\text{neg}(\cdot)$, such that for all $n \in \mathbb{N}$

$$\begin{aligned} & |\Pr[\text{ExpCFKEInd}(1^n, \mathcal{KE}, A_{ke}, A_{ind}, 0) = 1] - \\ & \Pr[\text{ExpCFKEInd}(1^n, \mathcal{KE}, A_{ke}, A_{ind}, 1) = 1]| \leq \text{neg}(n) \end{aligned}$$

In the second experiment ExpCFKENMal the goal of the adversary is to make the two parties output different keys with some correlation that may depend on the adversary's view. We require that for every adversary there exists a simulator such that the view of the adversary in an interaction that causes the parties to output two different keys K_A, K_B is simulatable given random keys K_A, K_B . The simulated view should be indistinguishable from the real one even given K_A and K_B .

This captures the intuition that the only way the adversary can make the parties output different keys is by making them output independent random keys. As we discussed in the introduction, we cannot completely prevent the adversary from influencing the output of the parties since she always has the option to omit the message that determines that outcome, unless the outcome satisfies some property (e.g. the key K_A has zero as its first bit). We require that this is essentially the only way the adversary can influence the outputs of the parties. This is captured by allowing the simulator to sample polynomially many pairs of uniformly random keys, and pick one pair for the output. This essentially grants the simulator exactly the ability to try a new key unless the current key satisfies a relatively likely property. For $b \in \{0, 1\}$ and a simulator S the non-malleability experiment is:

Experiment $\text{ExpCFKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, S, b)$

1. Run experiment $\text{ExpCFKE}(1^n, \mathcal{KE}, A_{ke})$ to obtain $(K_A^{\text{out}}, K_B^{\text{out}}, \text{view}(A_{ke}))$.
2. If $K_A^{\text{out}} = K_B^{\text{out}}$, flip an unbiased coin $b' \in_{\mathcal{R}} \{0, 1\}$ and set the outcome of the experiment to b' .
3. Else, run simulator $S(1^n)$ and allow S to sample a polynomial number of uniformly random key pairs $(K_A, K_B) \in (\{0, 1\}^n)^2$. Let view' be the output of the simulator and (K_A, K_B) be the last pair of keys sampled by S .
4. Set $Y_0 = (K_A^{\text{out}}, K_B^{\text{out}}, \text{view}(A_{ke}))$, and $Y_1 = (K_A, K_B, \text{view}')$, and let $b' \leftarrow_{\mathcal{R}} A_{mal}(Y_b)$. The outcome of the experiment is b' .

Definition 2. Let $\mathcal{KE} = (\text{KEInit}, \text{KE})$ be a credential-free key exchange. We say that \mathcal{KE} is non-malleable if for every CFKE adversary $\text{Adv} = (A_{ke}, A_{dist}, A_{mal})$, there exists an expected PPT simulator S , and a negligible function $\text{neg}(\cdot)$, such that for all $n \in \mathbb{N}$

$$\begin{aligned} & |\Pr[\text{ExpCFKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, S, 0) = 1] - \\ & \Pr[\text{ExpCFKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, S, 1) = 1]| \leq \text{neg}(n) \end{aligned}$$

Finally, we say that a CFKE protocol is secure if it satisfies both properties described above.

Definition 3. Let $\mathcal{KE} = (\text{KEInit}, \text{KE})$ be a credential-free key exchange. We say that \mathcal{KE} is secure if it satisfies privacy, and non-malleability.

On concurrent CFKE executions. In contrast to many definitions of KE with credentials, we do not need to include any additional players in our model. This is because in our setting the adversary can simulate all parties, except the two that he decides to target. There are no secrets associated with the communicating parties. Thus, there is no advantage in corrupting an honest party, since none of the other parties are even aware of its existence. Further, allowing for multiple protocol runs confers no advantage to the adversary, as there are no secrets that are re-used across runs the participant.

Universally composable CFKE. To allow for formal composability of CFKE protocols, we also provide a UC definition and construction of the CFKE notion. Due to limited space, this is presented in Appendix D.

4 Two Credential-Free Key Exchange Protocols

We present two constructions of CFKE protocols. Our main protocol is shown to be secure in the standard model, and requires two simultaneous rounds of communication. The second protocol that we present is what is commonly known as ‘‘Hashed Diffie-Hellman’’. We show that HDH is *not* a secure CFKE protocol in the standard random oracle model. However, we show that if one is willing to assume that HDH uses its own random oracle, that is not later used by other protocols, it is a secure CFKE protocol.

4.1 Protocol 1: Standard Model

We now present our main protocol (see Section 1.2 for its intuition). For our construction, we rely on a non-interactive equivocal commitment scheme with a specialized non-malleability property. Commitments are discussed in detail in Appendix B. We note that while our definition is specialized to allow us to prove the security of our CFKE protocol, it may be of independent interest as a property of non-interactive commitment schemes.

Let $\mathcal{COM} = \langle \text{ComInit}, \text{Commit}, \text{Decommit} \rangle$ be a 2-strongly non-malleable commitment scheme (discussed in detail in Appendix B), and let $\mathcal{PKE} = \langle \text{KeyGen}, \text{Enc}, \text{Dec} \rangle$ be a non-malleable public key encryption scheme. We construct a two-flow credential-less key exchange protocol $\mathcal{CFKE}_1 = \langle \text{KEInit}, \text{KE}_A, \text{KE}_B \rangle$ based on \mathcal{COM} and \mathcal{PKE} .

The public parameters generating algorithm KEInit on input security parameter 1^n runs $\text{ComInit}(1^n)$ to obtain a common reference string PUB for \mathcal{COM} , and outputs PUB as the public parameters of the key exchange protocol. The protocol KE consists of two rounds where in each round Alice and Bob send a message to each other. In our protocol, the actions of the parties are symmetric, and so the

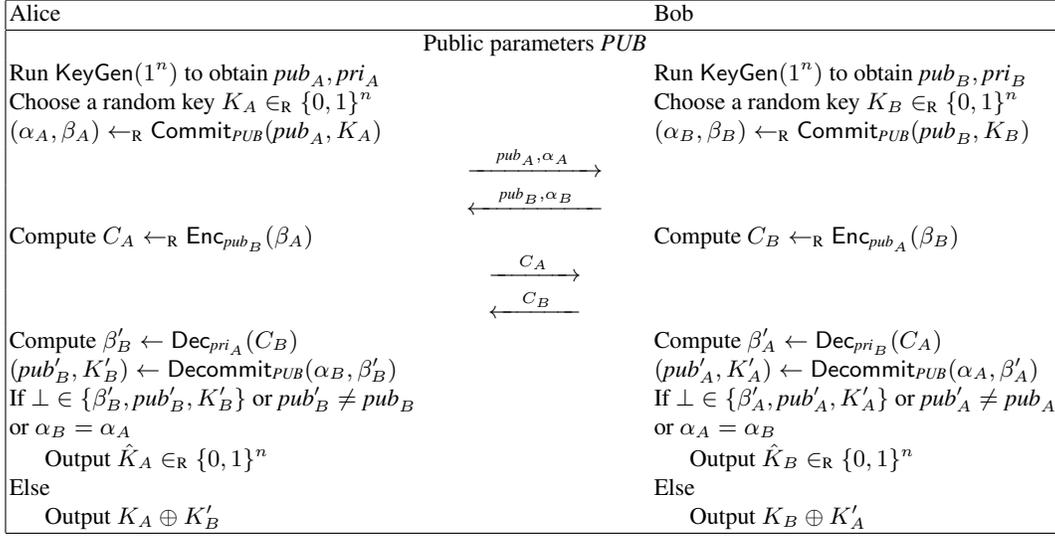


Fig. 1. The CFKE Protocol \mathcal{CFKE}_1

messages at each round can be sent in parallel, without either side waiting for the other to send first. The complete description of the protocol is given in Figure 1.

Privacy. To achieve privacy, the protocol must guarantee that if the two parties agree on a key (i.e. output the same string) then the adversary is unable to distinguish that key from a random one. we prove the following theorem:

Theorem 1 *If $\mathcal{PK}\mathcal{E}$ is a non-malleable public key encryption scheme and \mathcal{COM} is a computationally binding, 2-strongly non-malleable commitment scheme, then the protocol \mathcal{CFKE}_1 is private according to Definition 1.*

Intuitively, this follows from the following two facts: (i) if the adversary modifies A 's a public key (the case of B is symmetric), then because of the strong non-malleability of the commitment scheme, she must choose a new nonce to accompany the modified public key. However, if she does so, the probability that A and B output the same key is negligible. On the other hand, if she allows A and B to exchange public keys, then privacy follows from the semantic security of the encryption scheme. Due to limited space we present the details of the proof in Appendix C.1.

Non-Malleability. To show non-malleability, we must describe a simulator that can simulate the view of the adversary given a choice of one of polynomially many pairs of random outputs for the two parties. We prove the following theorem:

Theorem 2 *If \mathcal{COM} is a 2-strongly non-malleable commitment scheme then the protocol \mathcal{CFKE}_1 is non-malleable according to Definition 2.*

Proof sketch. We next present an overview of our proof. The complete details are given in Appendix C.2. Our proof is structured as follows. We first describe a simulator that achieves a weaker notion of security, where the simulator is allowed to lock an output for each party. That is, the simulator can sample pairs of output keys as before, but now it has the additional ability to lock one key in the pair, and continue randomizing the other key. Once a key is locked the simulator has committed to making that key the output of the corresponding party. Once we describe a simulator Sim in the weaker model, achieving simulation according to the actual notion of security is straightforward: sample a

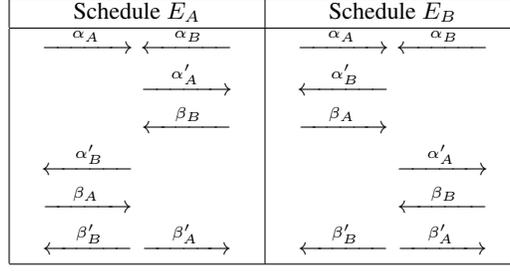


Fig. 2. Two possible schedules for adversarial commitments

pair of random keys \hat{K}_A and \hat{K}_B and guess at which iteration Sim will lock each key. Then, generate the rest of the keys randomly. This procedure is repeated until the guess is correct.

The simulator Sim itself is described in Figure 4 of Appendix C. On a high level the simulator works as follows. Let us denote by E_A (E_B) the event that the adversary submits both α'_A and α'_B before seeing an encryption of β_A (β_B). Note that since each party outputs the encrypted decommitment after obtaining the commitment of the other party, at least one of these events must always occur (see Figure 2 for an illustration of the two possible schedules). Specifically, according to schedule E_A the adversary submits both α'_A and α'_B before seeing an encryption of β_A . Similarly, according to schedule E_B the adversary submits both commitments before obtaining β_B . The simulator first generates a tuple γ of the form $(PUB, pub_A, \alpha_A, pub_B, \alpha_B, r_{adv})$. This commits the adversary to one of the two schedules described in Figure 2 (see proof of Claim C.1). Assuming that E_A is the schedule that the adversary follows conditioned on γ , the simulator proceeds to extract a decommitment for α'_A by simulating the interaction of the adversary with the protocol to completion (the case when the schedule induced by γ is E_B is symmetric). Now, by making use of the strong non-malleability of the commitment scheme, and ignoring (for the moment) the possibility that the adversary chooses to provide an invalid decommitment for α'_A , we know that there is a unique value K'_A that α'_A will be decommitted to. Therefore, the decommitment β'_A obtained by the simulator, together with the commitment α'_A , allow the simulator to obtain K'_A .

At this point, the simulator fixes a key \hat{K}_B to be output by Bob, and rewinds the adversary to the point where she submitted her commitment α'_A . The simulator then uses the equivocability of the commitment scheme to decommit α_B to $K_B \stackrel{def}{=} \hat{K}_B \oplus K'_A$. A similar extract-then-adjust procedure is repeated with the Alice side of the interaction: fixing $(\gamma, \alpha'_A, \beta_B)$ the simulator obtains a commitment α'_B from the adversary and simulates the protocol to completion to obtain a decommitment β'_B . Again, ignoring invalid commitments and applying the strong non-malleability property of the commitment scheme, there is only a single value to which the adversary can decommit α'_B . That value, K'_B , is obtained by the simulator from α'_B and β'_B . The simulator then rewinds the adversary to the point where $(\gamma, \alpha'_A, \beta_B, \alpha'_B)$ are fixed, fixes a key \hat{K}_A to be output by Alice, and decommits α_A to $K_A \stackrel{def}{=} \hat{K}_A \oplus K'_B$. As a result, K_A and K_B are properly distributed in the transcript of the protocol – uniformly random, and the simulator successfully forces the correct outputs \hat{K}_A and \hat{K}_B for Alice and Bob respectively.

Under the simplifying assumptions that the adversary always decommits a given commitment α'_A or α'_B to a unique value, and that no invalid decommitments are ever generated, the simulator described above perfectly simulates the view of the adversary in the non-malleability experiment. The main technical difficulty is caused by the fact that after fixing the commitment α'_A or α'_B the

adversary still has a choice whether to decommit to K'_A (K'_B) or to produce an invalid decommitment. To accommodate this possibility the actual simulator may rewind the adversary many times until the “right” kind of commitment is produced. For example, if the adversary first decommits α'_A to K'_A then after rewinding the simulator will keep trying new random keys \hat{K}_B for Bob until the adversary decommits α'_A to K'_A again. This repeated rewinding is what causes the running time of our simulator to be expected rather than strict polynomial. The only computational part of our argument concerns the inability of the adversary to decommit a single commitment α'_A or α'_B to more than one non- \perp value. If this were not the case then our simulator would potentially not be able to predict the value K'_A to which α'_A will be decommitted, and therefore fail to set β'_A appropriately. However, no efficient adversary can violate this requirement without breaking the strong non-malleability of the commitment scheme.

We give the complete details in Appendix C.2.

4.2 Protocol 2: Hashed Diffie-Hellman as CFKE

We next analyze the hashed Diffie-Hellman (HDH) [DH76] protocol (cf. Figure 3) in the context of CFKE. Note, HDH (and natural variants) are insecure in the sense of Definition 2. Indeed, a MIM adversary in HDH can learn the pre-images under the hash function H of the keys K_A and K_B by running a separate instance of the protocol with each party. This violates the non-malleability requirement of CFKE. One can then design (contrived) session protocols that become insecure when the adversary has this information. For example, a secure CFKE protocol can stop encrypting and authenticating messages if Alice receives a message containing the pre-image of her key under H .

At the same time, protocols such as HDH are natural and useful CFKE protocols, and should be allowed by the definition (especially since “incompatible” CFSS protocols can be naturally excluded – see below). A simple amendment to the definition of non-malleability resolves this. To accommodate protocols that output a hash of a value as the final key, we let the simulator in experiment ExpCFKENMal fix the output key pair to be the outcome of the last two queries that the simulator makes to the random oracle before terminating. More precisely, we modify step 3 in the experiment as follows:

- 3'. Run simulator $S(1^n)$ and allow S to query the random oracle H . Let view' be the output of the simulator and (K_A, K_B) be the last two values returned by H as responses to queries made by S .

Definition 4. Let $\mathcal{KE} = (\text{KEInit}, \text{KE})$ be a credential-free key exchange in the random oracle model. We say that \mathcal{KE} is secure if it satisfies privacy, and amended non-malleability.

We note that the original version of Step 3 can be simulated in the above variant simply by querying the random oracle on pairs of random inputs.

As discussed above, Definition 4 requires excluding “incompatible” CFSS protocols, namely those that may query the same RO used in CFKE. Hence, a simple way to ensure security of composed CFKE (Definition 4 version) and CFSS is to require that different RO are used for the two types of protocols. This is achieved in practice by using a protocol name as a prefix in all hash function calls.

Theorem 3 Suppose that the hash function H in the description of CFKE_2 is modeled as a random function Then, CFKE_2 is a secure CFKE according to Definition 4.

The proof of Theorem 3 is straightforward: privacy follows from the decisional Diffie-Hellman assumption (DDH), similarly to the standard Diffie-Hellman key exchange protocol. Non-malleability is shown by having the simulator query the random oracle on the two values queried last by the two parties.

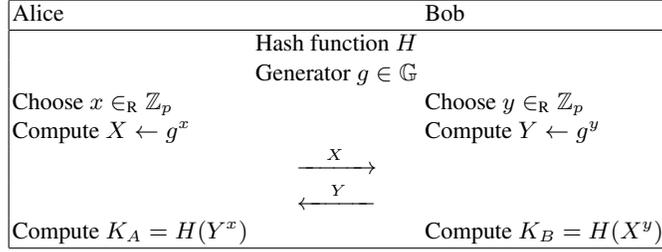


Fig. 3. The Hashed Diffie-Hellman protocol \mathcal{CFKE}_2

5 Definition and Construction of Credential-Free Secure Sessions

The notion of secure session protocol (with a supposedly shared session key) appears intuitive, and is often omitted from formal discussion. However, existing formalizations of secure sessions and secure channels [Sho99,CK02] show that subtleties arise even in these “simple” settings. Further, as we pointed out in Section 1.3, standard definitions (and even constructions!) of secure sessions don’t work in our credential-free scenario. In this section, we justify and formalize the new notion.

We simplify presentation by only considering one-way sessions where A communicates a long message to B . This message arrives in “pieces” to A , which are encrypted (using encryption function ENC) and sent to B one at a time. The adversary sees the encryptions and chooses the pieces in a very adaptive manner: he chooses the first piece, sees its encryption, chooses the second piece, sees its encryption, etc. B decrypts the pieces one at a time (using DEC). Each such decryption will involve an integrity test; if any such test fails, we say that B outputs a special symbol FAIL for the piece, and for simplicity we will assume that B must then output FAIL for all succeeding pieces.

Formally, a session protocol is a tuple $\langle ENC, DEC \rangle$, which satisfies correctness in the absence of an adversary. That is, if ENC and DEC are given the same key K , and if ENC is given a sequence of message pieces m_0, m_1, \dots, m_w , and if the resulting encryptions are fed into DEC , then DEC will output message pieces m_0, m_1, \dots, m_w . A standard secure session must satisfy the standard notions of *integrity* and *privacy*.

Definition of Credential-Free Secure Sessions (CFSS). We say a session protocol is a CFSS, if it satisfies: *Integrity*, *Privacy*, and *MIM-Integrity*. We omit formalization of the first two standard properties. To define MIM-integrity (and hence CFSS), fix a session protocol (ENC, DEC) .

MIM-Integrity: Consider an adversary Adv on input security parameter 1^n . Adv is a probabilistic algorithm that runs in time polynomial in n . We define an experiment that begins with two random n -bit strings K_A and K_B being chosen; Adv sees both K_A and K_B ; K_A is given to A (who uses ENC) and K_B is given to B (who uses DEC).

Adv interactively chooses m_0, m_1, \dots, m_w while seeing e_0, e_1, \dots, e_w . (Note that since A – that is ENC – is allowed to be probabilistic, Adv might not be able to compute e_0, e_1, \dots, e_w on his own.) Adv then computes and sends e'_0, e'_1, \dots, e'_j to B , where $0 \leq j \leq w$ and $e'_j = e_i$ for some i , $0 \leq i \leq w$. If B doesn’t output FAIL in response to e'_j , we say that Adv *wins*. Let $p_{Adv}(n)$ be the probability that Adv wins.

MIM-Integrity means that for every such Adv for every c , for sufficiently large n , $p_{Adv}(n) \leq 1/n^c$.

A CFSS Protocol. It is not hard to create a session protocol that satisfies these three concepts. It is straightforward to verify that the following construction is a CFSS.

Construction. Say that the session key consists of two parts: an n bit privacy key k_1 and an n bit integrity key k_2 . Assume we have two pseudo-random function generators F and F' , where $F_{k_1} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $F'_{k_2} : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$. A encrypts the i th n -bit piece m_i , by computing $\alpha = F_{k_1}(\bar{i}) \oplus m_i$ and letting the encryption be $e_i = \alpha F'_{k_2}(\bar{i}\alpha)$. (Here, \bar{i} denotes the n -bit representation of i .) B decrypts in the obvious way: given the i -th encryption $e'_i = \alpha'\beta'$, B FAILS if $\beta' \neq F'_{k_2}(\bar{i}\alpha')$, and otherwise outputs $F_{k_1}(\bar{i}) \oplus \alpha'$. It is easy to see that this satisfies integrity and privacy, but it does not necessarily satisfy MIM-integrity. In order to satisfy MIM-integrity, we add to e_i the string $F_{k_1}(1^n)$, and we add to B the additional stipulation that given $e'_i = \alpha'\beta'\gamma'$, B FAILS if $\gamma' \neq F_{k_1}(1^n)$. Alternatively, we could add a dedicated part k_3 of the key to each message, however, adding $F_{k_1}(1^n)$ allows for shorter keys.

5.1 Composing CFKE and CFSS

In this section we informally argue that composing CFKE and CFSS provides the guarantee that “the adversary must remain continuously active on the channel throughout the entire session to avoid detection”.

This is indeed easy to see. In one case, CFSS (namely, its standard integrity and privacy properties) guarantees that if the keys are random and unknown to Adv, then channel is fully secure in the standard strong sense. In the other case, Adv knows the keys of the players, and the keys are random and independent of each other, the MIM-integrity property of CFSS guarantees that as soon as Adv allows an unmodified message to pass between the players, it is immediately detected. Finally, the CFKE definition is tailored to explicitly guarantee that the keys that the players output fall under one of the two of the above cases.

References

- ALE09. J. Arkkio, V. Lehtovirta, and P. Eronen. RFC 5448: Improved extensible authentication protocol method for 3rd generation authentication and key agreement (EAP-AKA'). <http://tools.ietf.org/html/rfc5448>, May 2009.
- Bar02. Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, pages 345–355. IEEE Computer Society, 2002.
- BCL⁺05. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377. Springer, 2005.
- BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *SP '92: Proceedings of the 1992 IEEE Symposium on Security and Privacy*, page 72, Washington, DC, USA, 1992. IEEE Computer Society.
- Can00. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. <http://eprint.iacr.org/>.
- CCGS10. Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 255–276. Springer, 2010.
- CDSVW12. Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 449–466, Darmstadt, Germany, May 21–23, 2012. Springer, Berlin, Germany.
- CF01. R. Canetti and M. Fischlin. Universally composable commitments. pages 19–40. Springer, 2001.
- CHK⁺05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, 2005.

- CK02. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351, London, UK, 2002. Springer.
- CKOS01. Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In *EUROCRYPT*, pages 40–59, 2001.
- DCIO98. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 141–150. ACM, 1998.
- DDN91. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552. ACM, 1991.
- DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.
- DG03. I. Damgard and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, page 437. ACM, 2003.
- DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
- FF00. Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. In *CRYPTO*, pages 413–431, 2000.
- GL01. Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 408–432, London, UK, 2001. Springer.
- GL03. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543, 2003.
- KOY01. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494, 2001.
- KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011*, volume 6597, page 293, 2011.
- Lin11a. Y. Lindell. Highly-efficient universally-composable commitments based on the ddh assumption. *Advances in Cryptology–EUROCRYPT 2011*, pages 446–466, 2011.
- Lin11b. Yehuda Lindell. Highly-efficient universally-composable commitments based on the ddh assumption. In *Advances in cryptology, EUROCRYPT’11*, pages 446–466, Berlin, Heidelberg, 2011. Springer-Verlag.
- NT94. B.C. Neuman and T. Ts’o. Kerberos: an authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, sep 1994.
- Ped91. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
- PR05. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, page 542. ACM, 2005.
- RS84. Ronald L. Rivest and Adi Shamir. How to expose an eavesdropper. *Commun. ACM*, 27:393–394, April 1984.
- Sho99. Victor Shoup. On formal models for secure key exchange. Technical Report RZ 3120 (#93166), IBM, 1999.

A Probabilistic Lemma

We make use of the following simple fact about sampling from conditional distributions.

Lemma 1. *Let D be a probability distribution over some sample space X . Let $E \subseteq X$ be any event. Then, the following procedure outputs a random element sampled according to the distribution $D|E$:*

1. *Sample x from X according to D .*
2. *If $x \in E$ output x , otherwise repeat.*

Furthermore, the expected number of samples until a value is output is $1/\Pr[E]$.

Before presenting our main construction in Section 4.1, we present a building block that we construct – Strongly Non-Malleable Commitments.

B Strongly Non-Malleable Commitments

A non-interactive *commitment scheme* consists of a triple of PPT algorithms: (ComlNit, Commit, Decommit) and a length parameter $d \in \mathbb{N}$. For $n \in \mathbb{N}$, $m \in \{0, 1\}^{n^d}$, ComlNit(1^n) outputs a public parameter string PUB , Commit($1^n, PUB, m$) outputs a commitment α and decommitment β , and Decommit($1^n, PUB, \alpha, \beta$) outputs a string $m' \in \{0, 1\}^d$ or a special string \perp .

Definition 5. A *commitment scheme* (ComlNit, Commit, Decommit) is *correct* if for all $n \in \mathbb{N}$, $m \in \{0, 1\}^d$, random tapes $r_1, r_2, r_3 \in \{0, 1\}^{\text{poly}(n)}$, Decommit($1^n, PUB, \alpha, \beta; r_1$) = m if $PUB = \text{ComlNit}(1^n; r_2)$ and $(\alpha, \beta) = \text{Commit}(1^n, PUB, m; r_3)$.

Non-interactive commitment schemes come in two flavors: “perfectly binding”, where each commitment can be decommitted to a unique value, and that value is computationally hidden without the decommitment string; and “perfectly hiding”, where a commitment string contains no information about the underlying message, but a computationally bounded adversary is only able to decommit to one value. In this work we use perfectly hiding commitment schemes that are also equivocal. Informally, a commitment scheme is equivocal if there is an alternative method of generating the public parameters PUB that provides a trapdoor, and given that trapdoor any commitment string can be decommitted to any message. Furthermore, if the alternatively generated PUB has the same distribution as the actual PUB then the commitment scheme is *perfectly* equivocal. We now define perfect equivocability formally:

Definition 6. A *commitment scheme* (ComlNit, Commit, Decommit) is *perfectly equivocal* if there exists a triple of PPT algorithms (Elnit, ECom, EDecom) such that

1. Elnit(1^n) outputs (PUB, s) , where s is trapdoor information about PUB .
2. For all $m \in \{0, 1\}^d$ the following two random variables are identically distributed:

$$\begin{aligned} & \{(PUB, \alpha, \beta) | (PUB, s) \leftarrow \text{Elnit}(1^n); \alpha \leftarrow \text{ECom}(1^n, PUB); \beta \leftarrow \text{EDecom}(s, \alpha, m)\} \\ & \{(PUB, \alpha, \beta) | PUB \leftarrow \text{ComlNit}(1^n); (\alpha, \beta) \leftarrow \text{Commit}_{PUB}(m)\} \end{aligned}$$

Strong Non-Malleability Another property of commitment schemes that we require is non-malleability. Two definitions of non-interactive non-malleable commitments have appeared in the literature, both aiming to capture variants of the following intuitive property: no efficient adversary should be able, given a commitment α to a message m , to produce a commitment to a related message m' . One definition, non-malleability with respect to *commitment* [DDN91], requires exactly this. That is, that no efficient adversary can produce a commitment to a related message. The second definition, non-malleability with respect to *opening* [DCIO98, FF00], requires the adversary to successfully decommit to the related message after producing the commitment. We note that for statistically hiding commitment schemes, non-malleability with respect to commitment is not a meaningful notion since a commitment can be decommitted to many different messages (in the information theoretic sense). Consequently, non-malleability with respect to opening is considered the right notion for statistically and perfectly hiding commitments.

In both definitions described above the requirement that m is related to m' is modeled by assuming the existence of a polynomial time computable relation R such that $R(m, m')$ is significantly more likely to hold than $R(m, m'')$ where m'' is chosen according to some distribution that does not depend on the commitment α . However, this approach does not rule out a scenario where m' depends on the decommitment string β that decommits α to m . In particular, it is possible that by seeing different valid decommitments of α , possibly all to the same message m , or to different messages, the

adversary is able to produce decommitments of α' to different messages. We believe that in many natural applications of non-malleable commitments the adversary will, in fact, be able to choose a decommitment for α' after seeing a decommitment for α . One such application is our credential-free key exchange protocol.

In this paper we propose a new, stronger, definition of non-interactive non-malleable commitment schemes. Our definition is described as a game and does not require specifying any particular relation that must hold between the strings in question. Instead, we require that no efficient adversary can win in the following game: the adversary is given an equivocal commitment α , and must produce a different commitment α' such that given two decommitments for α that decommit to different messages the adversary must produce decommitments for α' that decommit to different messages. For our analysis, we actually require an even stronger type of non-malleability: where the adversary is unable to win the above game even if given equivocation of multiple commitments. Somewhat counter-intuitively, security with respect to a single commitment does not imply security relative to multiple commitments (we direct the reader to [DDN00] for a discussion). It may be instructive to first provide a definition with respect to one commitment, and then extend to multiple commitments. However, to avoid confusing the reader with numerous definitions we directly provide the definition of strong non-malleability with respect to multiple commitments, and merely state that the single commitment variant is a special case.

Formally, strong non-malleability with respect to k commitments is defined through the following experiment:

Experiment $\text{ExpSNMal}(1^n, k, Adv)$

1. Elnit is run to obtain a tuple (PUB, s) and ECom is run k times to obtain commitments $\alpha_1, \dots, \alpha_k$. PUB and $(\alpha_1, \dots, \alpha_k)$ are given to the adversary.
2. The adversary submits a commitment string α and $2k$ messages m_1, \dots, m_k and m'_1, \dots, m'_k . Let w be the current state of the adversary.
3. $2m$ decommitments are computed $\beta_i \leftarrow \text{EDecom}(s, \alpha_i, m_i)$ and $\beta'_i \leftarrow \text{EDecom}(s, \alpha_i, m'_i)$.
4. Two copies of the adversary are run in parallel with state w . One copy is given β_1, \dots, β_k . The other copy is given $\beta'_1, \dots, \beta'_k$. Each copy then outputs a decommitment for α . Let β and β' be the decommitments produced by the first and second copies of the adversary respectively.
5. Let $m = \text{Decommit}_{PUB}(\alpha, \beta)$ and $m' = \text{Decommit}_{PUB}(\alpha, \beta')$. The outcome of the experiment is 1 if $m \neq m'$ and $\perp \notin \{m, m'\}$. Otherwise, the outcome of the experiment is 0.

Definition 7. For $k \in \mathbb{N}$, a perfectly equivocal commitment scheme $(\text{ComInit}, \text{Commit}, \text{Decommit}, \text{Elnit}, \text{ECom})$ is k -strongly non-malleable if for every PPT adversary Adv there exists a negligible function $neg(\cdot)$ such that for all $n \in \mathbb{N}$, $\Pr[\text{ExpSNMal}(1^n, k, Adv) = 1] \leq neg(n)$.

In the following section we present an adaptation of the commitment scheme of [CKOS01] that is 2-strongly non-malleable.

A Strongly Non-Malleable Commitment Scheme We show that an adaptation of the discrete logarithm based commitment scheme of Di Crescenzo *et al.* [CKOS01] is 2-strongly non-malleable. Our proof essentially follows the proof in [CKOS01], and perhaps surprisingly is somewhat simpler. We start with a full description of the commitment scheme. The scheme makes use of a computational group generator \mathcal{G} , and consists of three PPT algorithms $\mathcal{COM} = \langle \text{ComInit}, \text{Commit}, \text{Decommit} \rangle$, which are defined as follows:

Setup. Run $\mathbb{G} \leftarrow_{\mathcal{R}} \mathcal{G}(1^n)$, generate $g_1, g_2, g_3, g_4 \leftarrow_{\mathcal{R}} \mathbb{G}$, and let $H : \mathbb{G} \rightarrow \mathbb{Z}_p$ be a target collision resistant hash function. $PUB = (\mathbb{G}, g_1, g_2, g_3, g_4, H)$.

Commitment. Generate $r_1, r_2, r_3, r_4, r_5 \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, set $A \leftarrow g_1^{r_1} g_2^{r_2} g_3^{r_3} g_4^{r_4} \gamma \leftarrow H(A)$, $D \leftarrow g_1^{\gamma^2} g_2^{\gamma} g_3$, $B \leftarrow D^m g_4^{r_5}$, $\sigma \leftarrow \text{MAC}_{r_1, r_2}(B)$. The commitment is $\alpha = (A, B, \sigma)$. The decommitment is $\beta = (m, r_1, r_2, r_3, r_4, r_5)$. Output (α, β)

Decommitment. $\text{Decommit}(PUB, \alpha, \beta)$: Parse $\alpha = (A, B, \sigma)$, $\beta = (m, r_1, r_2, r_3, r_4, r_5)$. If the parsing fails output \perp . Verify $A \stackrel{?}{=} g_1^{r_1} g_2^{r_2} g_3^{r_3} g_4^{r_4}$; $B \stackrel{?}{=} (g_1^{\gamma^2} g_2^{\gamma} g_3)^m g_4^{r_5}$; $\sigma \stackrel{?}{=} \text{MAC}_{r_1, r_2}(B)$ If verification succeeds output m . Otherwise, output \perp .

Theorem 4 *The commitment scheme COM is perfectly hiding, computationally binding, perfectly equivocal, and 2-strongly non-malleable.*

B.1 Proof of Theorem 4

Perfect hiding and equivocability are self evident. The computational binding property follows by a simple reduction to the binding property of the Pedersen commitment scheme [Ped91]. We concentrate on an adaptation of the non-malleability proof from [CKOS01] to our scheme and the stronger security definition. Let $\alpha_1 = (A_1, B_1, \sigma_1)$ and $\alpha_2 = (A_2, B_2, \sigma_2)$ be the two commitments given to the adversary, and let (A', B', σ') be the commitment produced by the adversary. Following the proof in [CKOS01], we consider the following cases:

Case 1: $A' = A_1$ or $A' = A_2$. Suppose without loss of generality that $A' = A_1$. Then, the argument proceeds as in [CKOS01], that is, if the adversary later produces a decommitment where $(r_1, r_2, r_3, r_4) = (r'_1, r'_2, r'_3, r'_4)$ then it can be used to break the binding property of the Pedersen commitment scheme. If $(r_1, r_2, r_3, r_4) \neq (r'_1, r'_2, r'_3, r'_4)$ then with overwhelming probability the decommitment will result in \perp since the MAC key (r_1, r_2) is information theoretically hidden given (α_1, α_2) .

Case 2: $A' \neq A_1$ and $A' \neq A_2$ but $\gamma' = \gamma_1$ or $\gamma' = \gamma_2$. Again, similarly to [CKOS01], in this case we are able to find a collision in H .

Case 3: $A' \neq A_1$, $A' \neq A_2$, $\gamma' \neq \gamma_1$ and $\gamma' \neq \gamma_2$. We show that if an adversary is able to successfully decommit α' to two different messages then we can use it to produce two different valid decommitments to a related Pedersen commitment. Then, using the reduction described in [CKOS01] we can compute the discrete logarithm of a given group element. More precisely, we describe a new adversary Adv' that is given g_1 and g_4 , and proceeds as follows:

- Generate $r_i, s_i, r_{2,i}, r_{3,i}, u_i \in_{\mathbb{R}} \mathbb{Z}_p$ for $i \in \{1, 2\}$, and generate $t, t' \in_{\mathbb{R}} \mathbb{Z}_p$.
- Set $A_i = g_1^{r_i} g_4^{s_i}$, $B_i = u_i$, $\sigma_i = \text{MAC}_{r_{1,i}, r_{2,i}}(B_i)$.
- Set $r_{1,i} = (\gamma_1 + \gamma_2)r_{2,i} - \gamma_1\gamma_2r_{3,i} + r_i$ and $r_{4,i} = -tr_{2,i} - t'r_{3,i} + s_i$.
- Set $PUB = (g_1, g_2, g_3, g_4)$, $\alpha_i = (A_i, B_i, \sigma_i)$ and $s = (r_{1,1}, r_{1,2}, r_{2,1}, r_{2,2}, r_{3,1}, r_{3,2}, r_{4,1}, r_{4,2}, t, t', u_1, u_2)$.

The tuple $(PUB, \alpha_1, \alpha_2)$ is given to Adv , which in turn submits a commitment $\alpha' = (A', B', \sigma')$ and two pairs of messages $(m_{1,1}, m_{1,2})$ and $(m_{2,1}, m_{2,2})$. Adv' then computes decommitments for $1 \leq i, j \leq 2$

$$\beta_{i,j} = (r_{1,j}, r_{2,j}, r_{3,j}, r_{4,j}, u_j - m_{i,j}(t\gamma_j + t'))$$

Let w be the state of Adv at this point. Adv' runs two copies of Adv in parallel starting from state w . Copy i receives decommitments $(\beta_{i,1}, \beta_{i,2})$. It is easy to verify that the public parameters, commitments, and decommitments produced by Adv' are properly distributed. Note that for $i \in \{1, 2\}$

$$D_i = g_1^{\gamma_i^2} g_2^{\gamma_i} g_3 = g_4^{t\gamma_i + t'}$$

There are two subcases to consider:

Case 3.1: $D' = g_1^{(\gamma')^2} g_2^{\gamma'} g_3 = g_4^{t\gamma_i+t'}$ for some $i \in \{1, 2\}$. In this case, if it also holds that $B' = B_i$, then α' can be decommitted using any decommitment for α_i . However, we argue that no efficient adversary can construct a commitment that matches this case: if $D' = g_4^{t\gamma_i+t'}$ then

$$g_1^{(\gamma')^2 - (\gamma_1 + \gamma_2)\gamma' + \gamma_1\gamma_2} g_4^{t\gamma' + t'} = g_4^{t\gamma_i + t'}$$

Since $\gamma' \notin \{\gamma_1, \gamma_2\}$, it is not a root of the polynomial in the exponent of g_1 , this allows us to recover the discrete logarithm of g_4 relative to g_1 by solving a linear equation.

Case 3.2: $D' = g_1^{(\gamma')^2} g_2^{\gamma'} g_3 \neq g_4^{t\gamma_i+t'}$ for both $i \in \{1, 2\}$. In this case, similarly to [CKOS01] we obtain two different valid decommitments for a Pedersen commitment using generators g_4 and $g_1^{(\gamma')^2 - (\gamma_1 + \gamma_2)\gamma' + \gamma_1\gamma_2} g_4^{t\gamma' + t'}$. Using the reduction in [Ped91], we can now compute the discrete logarithm of g_4 relative to g_1 .

C Analysis of CFKE Protocol

C.1 Privacy

In this section we prove Theorem 1. We reduce the security of our construction to the chosen ciphertext security of $\mathcal{PK}\mathcal{E}$ and the security of the commitment scheme \mathcal{COM} . Our proof proceeds through a series of hybrid experiments, where the initial experiment is the original experiment of privacy, and the final experiment is such that the transcript of the protocol contains no information about the key that A and B agree on (if they do indeed agree on some key). We show that the adversary cannot distinguish between participating in the original and final experiments, and conclude that she must have almost identical advantage in breaking privacy in these experiments. This immediately implies the privacy property since the advantage of the adversary in the final experiment is zero.

We start with a description of the hybrid experiments. In each experiment i we denote by X_i the random variable which is 1 if the adversary successfully guesses the bit b , and 0 otherwise.

Game 1. Game 1 is the original privacy security game.

Game 2. Game 2 proceeds identically to Game 1, except that in the beginning of the game a random string $z_A \in_{\mathcal{R}} \{0, 1\}^n$ is chosen, and if, after the second round, C'_B decrypts to z_A then A outputs $K_A \oplus K_B$.

The modification in Game 2 introduces a secret string that allows A to output the correct key without ever receiving a decommitment for the commitment α_B that was sent by B in the first round. The two games proceed identically unless the adversary manages to produce an encryption C'_B of z_A . However, since z_A is uniformly random, and the adversary is given no information about it, this event occurs with negligible probability. The proof of the following claim is trivial, and is omitted.

Claim. Let F_2 be the event that the adversary produces a ciphertext C'_B that decrypts to z_A . Then, $|\Pr[X_1 = 1] - \Pr[X_2 = 1]| \leq \Pr[F_2]$, and $\Pr[F_2] = \frac{1}{2^n}$.

Game 3. Game 3 proceeds identically to Game 2 with the following exception. In game 3, if the adversary submits a public key $pub'_A = pub_A$ then C_B is modified to be an encryption of z_A instead of β_B .

We now show that no adversary can distinguish between games 2 and 3. Intuitively, this follows from the non-adaptive chosen-ciphertext security of $\mathcal{PK}\mathcal{E}$. If the adversary leaves the public key pub_A

unmodified then she either has to forward the ciphertext C_B to A without changing it, or compute a new ciphertext C'_B with an unrelated plaintext. Suppose that an adversary can distinguish between games 2 and 3. This gives a way to break the security of $\mathcal{PK}\mathcal{E}$: if the adversary does not modify C_B , then in both games A will output $K_A \oplus K_B$, therefore the adversary must be distinguishing the content of C_B in game 2 (the decommitment β_B) from its content in game 3 (the string z_A). If the adversary produces a modified ciphertext $C'_B \neq C_B$, then we can decrypt C'_B using the decryption oracle. This allows us to simulate the adversary perfectly.

Claim. Let $\varepsilon = |\Pr[X_2 = 1] - \Pr[X_3 = 1]|$. Then, we can construct an adversary that breaks the non-malleability of the public key encryption scheme $\mathcal{PK}\mathcal{E}$ with advantage ε .

Proof. We construct an adversary A_{pke} that simulates Adv , and breaks the non-malleability of $\mathcal{PK}\mathcal{E}$. Initially A_{pke} is given a public key pub . A_{pke} then proceeds as follows. It sets $pub'_A = pub$, and generates the values $(K_A, K_B, \alpha_A, \beta_A, \alpha_B, \beta_B, pri_B, pub_B)$ from the appropriate distributions, and chooses $z_A \in_{\mathcal{R}} \{0, 1\}^n$. A_{pke} then submits β_B and z_A as its two messages in the security experiment of $\mathcal{PK}\mathcal{E}$, and obtains a challenge ciphertext C^* .

The adversary Adv then proceeds to submit either (pub'_A, α'_A) or (pub'_B, α'_B) . If the adversary submits (pub'_B, α'_B) first then A_{pke} computes $C_A = \text{Enc}_{pub'_B}(\text{Dec}_A)$ and gives C_A to Adv . When the adversary submits (pub'_A, α'_A) , A_{pke} checks whether $pub'_A \neq pub_A$, and if so stops the simulation and flips a coin in the security experiment of $\mathcal{PK}\mathcal{E}$. Note that if $pub'_A \neq pub_A$ then games 2 and 3 proceed identically, and so Adv will behave identically in both cases.

Suppose that Adv submits (pub'_A, α'_A) where $pub'_A = pub_A$. Then, A_{pke} sets $C_B = C^*$, and gives C_B to Adv . Adv then submits C'_B and stops. At this point A_{pke} proceeds in one of two ways: if $C'_B = C_B$ then C'_B is either an encryption of β_B or z_A . In both cases, the output of A is $K_A \oplus K_B$ in both game 2 and 3. If $C'_B \neq C_B$ then A_{pke} submits C'_B to its decryption oracle to obtain β'_B , and proceeds to compute K_A^{out} according to the protocol.

Finally, A_{pke} uses the private key pri_B to decrypt C'_A and compute K_B^{out} . If $K_A^{out} = K_B^{out}$ then K_A^{out} is given to Adv , and A_{pke} outputs whatever Adv outputs. Otherwise, A_{pke} flips a fair coin and outputs the outcome. It is now easy to see that A_{pke} simulates Adv perfectly in game 2 if C^* is an encryption of β_B , and in game 3 if it is an encryption of z_A . The claim follows.

We now define games 4 and 5 analogously for B :

Game 4 Game 4 proceeds identically to Game 3, except that in the beginning of the game a random string $z_B \in_{\mathcal{R}} \{0, 1\}^n$ is chosen, and if, after the second round, C'_A decrypts to z_B then B outputs $K_A \oplus K_B$.

Game 5. Game 5 proceeds identically to Game 4 with the following exception. In game 5, if the adversary submits a public key $pub'_B = pub_B$ then C_A is modified to be an encryption of z_B instead of β_A .

The proofs of the following two claims are almost identical to the proofs of Claim C.1 and Claim C.1, and are omitted.

Claim. Let F_4 be the event that the adversary produces a ciphertext C'_A that decrypts to z_B . Then, $|\Pr[X_3 = 1] - \Pr[X_4 = 1]| \leq \Pr[F_4]$, and $\Pr[F_4] = \frac{1}{2^n}$.

Claim. Let $\varepsilon = |\Pr[X_4 = 1] - \Pr[X_5 = 1]|$. Then, we can construct an adversary that breaks the non-malleability of the public key encryption scheme $\mathcal{PK}\mathcal{E}$ with advantage ε .

Game 6. Game 6 proceeds identically to Game 5 except that if $pub'_A \neq pub_A$ but $\alpha'_A = \alpha_A$ then B always outputs a random key \hat{K}_B , regardless of what C'_A decrypts to.

Intuitively, an adversary that manages to modify pub_A without changing the commitment α_A is able to equivocate α_A . We use this fact to break the computational binding property of \mathcal{COM} . Let F_6 be the following event:

$$F_6 = \{pub'_A \neq pub_A; \alpha'_A = \alpha_A; \text{Decommit}_{PUB}(\alpha_A, \text{Dec}_{pri_B}(C'_A)) \neq \perp; pub''_A = pub'_A\}$$

Claim. Let $\varepsilon = |\Pr[X_6 = 1] - \Pr[X_5 = 1]|$. Then, we can construct an adversary that breaks the binding property of the commitment scheme \mathcal{COM} with advantage ε .

Proof. We first show that unless F_6 occurs, games 5 and 6 proceed identically. Clearly, this is true if $pub'_A = pub_A$ or $\alpha'_A \neq \alpha_A$. Suppose that $\text{Decommit}_{PUB}(\alpha_A, \text{Dec}_{pri_B}(C'_A)) = \perp$ or $pub''_A \neq pub'_A$. Then, in both games B outputs a random key \hat{K}_B .

Now, let $\varepsilon = \Pr[F_6]$. We construct an adversary A_{bind} that simulates Adv and violates the computational binding property of \mathcal{COM} with probability ε . A_{bind} initially receives PUB , and has to produce a commitment α , and two different decommitments that decommit to different, non \perp values.

A_{bind} proceeds as follows. It performs a perfect simulation of Adv in game 5 up to the point where B is about to output a key K_B^{out} . At this point, A_{bind} can tell whether event F_6 occurred. If it did, A_{bind} outputs α_A as its commitment, and (β_A, β'_A) as the decommitments. We know that β_A decommits α_A to a string of the form (pub_A, K_A) , and that β'_A decommits to (pub'_A, K'_A) . The fact that $pub_A \neq pub'_A$ implies that A_{bind} has successfully broken the binding property.

Game 7. Game 7 proceeds identically to Game 6 except that if $pub'_B \neq pub_B$ but $\alpha'_B = \alpha_B$ then A always outputs a random key \hat{K}_A , regardless of what C'_B decrypts to.

Event F_7 , and Claim C.1 are symmetric to event F_6 and Claim C.1.

$$F_7 = \{pub'_B \neq pub_B; \alpha'_B = \alpha_B; \text{Decommit}_{PUB}(\alpha_B, \text{Dec}_{pri_A}(C'_B)) \neq \perp; pub''_B = pub'_B\}$$

Claim. Let $\varepsilon = |\Pr[X_7 = 1] - \Pr[X_6 = 1]|$. Then, we can construct an adversary that breaks the binding property of the commitment scheme \mathcal{COM} with advantage ε .

Game 8. Proceeds identically to Game 7 except that if $\alpha_A = \alpha'_A$ or $\alpha_B = \alpha'_B$ then the outcome of the experiment is determined by flipping a fair coin.

Claim. $\Pr[X_7 = 1] = \Pr[X_8 = 1] \leq \frac{1}{2^n}$

Proof. We show that if $\alpha_A = \alpha'_A$ then the outcome of the experiment is 1 with probability $1/2$ in both game 7 and 8. The case of $\alpha_B = \alpha'_B$ is symmetric. There are two cases to consider: $pub_A = pub'_A$ and $pub_A \neq pub'_A$. If $pub_A = pub'_A$ then according to the modification introduced in game 3 the ciphertext C_B sent from Bob to Alice is an encryption of z_A , which is a uniformly random string independent from K_B . Consequently, the adversary never sees (information theoretically) a decommitment to K_B . Since the commitment scheme is perfectly hiding, the adversary never gains any information about K_B , and so the value $K_A \oplus K_B$ is a uniformly random string. Therefore, if $pub_A = pub'_A$ the adversary wins in game 7 with probability $1/2$.

If $pub_A \neq pub'_A$ then Bob always outputs a random key \hat{K}_B . Therefore, in the privacy experiment the key K_A^{out} output by Alice will be equal to \hat{K}_B with probability $\frac{1}{2^n}$ (recall that if the keys are not equal, the outcome of the experiment is determined by flipping a fair coin).

Game 9. Proceeds identically to Game 8 except that the public parameters PUB , commitment strings α_A and α_B and the decommitments β_A and β_B are generated using the equivocal algorithms EInit , ECom and EDecom instead of the regular commitment algorithms. More formally, the following values are sampled:

$$\begin{aligned} (PUB, s) &\leftarrow_{\text{R}} \text{EInit}(1^n); \alpha_A \leftarrow_{\text{R}} \text{ECom}_{PUB}(s); \alpha_B \leftarrow_{\text{R}} \text{ECom}_{PUB}(s) \\ (pub_A, pri_A) &\leftarrow_{\text{R}} \text{KeyGen}(1^n); (pub_B, pri_B) \leftarrow_{\text{R}} \text{KeyGen}(1^n); K_A, K_B \leftarrow_{\text{R}} \{0, 1\}^n \\ \beta_A &\leftarrow_{\text{R}} \text{EDecom}(\alpha_A, s, (K_A, pub_A)); \beta_B \leftarrow_{\text{R}} \text{EDecom}(\alpha_B, s, (K_B, pub_B)) \end{aligned}$$

The values $PUB, pub_B, pub_A, \alpha_A, \alpha_B, \beta_A, \beta_B$ are then to generate the communication between Alice and Bob. By the perfect equivocability of the commitment scheme we get

Claim. $\Pr[X_8 = 1] = \Pr[X_9 = 1]$

Game 10. Game 10 proceeds identically to Game 9 except that if $\{\alpha_A, \alpha_B\} \cap \{\alpha'_A, \alpha'_B\} \neq \emptyset$ then the outcome of the experiment is determined by flipping a fair coin.

Claim. Let $\varepsilon \stackrel{def}{=} |\Pr[X_9 = 1] - \Pr[X_{10} = 1]|$. Then, we can construct an adversary that breaks the 2-strong non-malleability property of \mathcal{COM} with advantage $\frac{\varepsilon^2}{2} - \frac{1}{2^{n+1}}$.

Proof. First, let us consider the case where $\alpha'_B = \alpha_A$ or $\alpha'_A = \alpha_B$. Then, according to the protocol a random key is chosen for the party in question. Consequently, games 9 and 10 proceed identically in this case. Let F_9 be the event that $\alpha_A \neq \alpha'_A, \alpha_B \neq \alpha'_B, K_A^{out} = K_B^{out}$, and let $\varepsilon = \Pr[F_9]$. Also, let us denote by E_A (E_B) the event that the adversary submits both α'_A and α'_B before seeing an encryption of β_A (β_B). Note that since each party outputs the encrypted decommitment after obtaining the commitment of the other party, at least one of these events must always occur (see Figure 2 for an illustration of the two possible schedules).

Consider tuples of the form $(PUB, \alpha_A, \alpha_B, r_{adv})$ where PUB are the global public parameters, α_A and α_B are the non-adversarial commitments of Alice and Bob respectively, and r_{adv} are the random bits used by the adversary. A tuple γ of this form determines whether event E_A, E_B , or $E_A \wedge E_B$ occurs. Let us denote $\varepsilon_\gamma = \Pr[F_9 | \gamma]$.

We shall now describe an adversary A' that uses A_{ke} to break the 2-strong non-malleability of the commitment scheme. A' starts by receiving PUB, α^* , and flipping a fair coin $c_1 \in_{\text{R}} \{0, 1\}$ to guess whether event E_A or E_B will occur. Suppose without loss of generality that A' guesses that E_A occurs. A' then generates two random keys $K_{A,0}, K_{A,1} \in_{\text{R}} \{0, 1\}^n$, a random commitment α_B , random bits r_{adv} , and sets $\alpha_A = \alpha^*$. Let γ be the tuple $(PUB, \alpha_A, \alpha_B, r_{adv})$. A' then verifies that γ causes event E_A to occur by checking that A_{ke} , when given PUB, α_A, α_B outputs α'_A (note that the adversary always has to submit α'_B before she is given β_A , therefore submitting α'_A guarantees that event E_A occurs) or a pair (α'_A, α'_B) . If this is not the case then A' aborts.

Suppose that E_A occurs. The adversary A_{ke} has now committed to mauling the commitment α_A , but is still free to decide which of the two commitments α'_A and α'_B she will equivocate. In fact, the adversary may make that decision adaptively based on the decommitment that she receives for α_A . It is therefore instructive to recall that the two messages chosen by A' are the random keys $K_{A,0}$ and $K_{A,1}$. Consequently, conditioning on the tuple γ , the distributions on the decommitment given to A' are identical for both messages.

We shall now argue that if, conditioned on γ , A_{ke} causes event F_9 to occur with probability ε_γ then in two independently generated continuations of γ , A_{ke} is likely to decommit at least one of α'_A or α'_B to two different values. The values that are missing from γ to completely determine the execution

of the protocol are only β_A and β_B . The decommitment β_B does not affect the value that α_A is decommitted to. Therefore, we can fix it for both executions: let $K_B \in_{\mathcal{R}} \{0, 1\}^n$ and let β_B be a random decommitment of α_B to K_B . We now focus on the extended tuple $\gamma' = (PUB, \alpha_A, \alpha_B, \beta_B, r_{adv})$. Let $\varepsilon_{\gamma'}$ be the probability that F_9 occurs conditioned on γ' , where the probability is over K_A and the randomness needed to generate a decommitment β_A of α_A to K_A . Consider two independently generated transcripts δ and δ' that are continuations of γ' . Since the transcripts are independently generated using γ' as a starting point we have

$$\Pr[\delta, \delta' \in F_9] = \varepsilon_{\gamma'}^2$$

We now show that if $\delta, \delta' \in F_9$ and, in addition, $K_{A,0} \neq K_{A,1}$, then the adversary A_{ke} has to decommit at least one of the commitments α'_A or α'_B to two different values given decommitments of α_A to $K_{A,0}$ and $K_{A,1}$ respectively. Let us denote by $K_{A,b}^{out}$ and $K_{B,b}^{out}$ the keys output by A and B respectively due to the transcripts δ and δ' . Then we get

$$\begin{aligned} K_{A,0}^{out} &= K_{A,0} \oplus K'_{B,0}; & K_{B,0}^{out} &= K_B \oplus K'_{A,0} \\ K_{A,1}^{out} &= K_{A,1} \oplus K'_{B,1}; & K_{B,1}^{out} &= K_B \oplus K'_{A,1} \end{aligned}$$

The event $\delta, \delta' \in F_9$ implies that $K_{A,b}^{out} = K_{B,b}^{out}$ for $b \in \{0, 1\}$. Therefore, if $K_{A,0} \neq K_{A,1}$ then we get that either $K'_{B,0} \neq K'_{B,1}$ or $K'_{A,0} \neq K'_{A,1}$. Let us denote these events by Equiv_B and Equiv_A respectively. Clearly, $\Pr[K_{A,0} = K_{A,1}] = 1/2^n$. Consequently, the above argument shows that

$$\Pr[\text{Equiv}_A \vee \text{Equiv}_B | \gamma'] \geq \varepsilon_{\gamma'}^2 - \frac{1}{2^n}$$

Consequently, we can assume without loss of generality that

$$\Pr[\text{Equiv}_A | \gamma'] \geq \frac{\varepsilon_{\gamma'}^2}{2} - \frac{1}{2^{n+1}} \quad (1)$$

Getting back to the construction of A' , A' submits $(\alpha'_A, K_{A,0}, K_{A,1})$ in the strong non-malleability experiment. At this point two copies of A' are run independently. One is given a decommitment $\beta_{A,0}$ that decommits α_A to $K_{A,0}$. The other is given a decommitment $\beta_{A,1}$ that decommits to $K_{A,1}$. A' now continues to simulate A_{ke} to obtain the decommitments $\beta'_{A,0}$ and $\beta'_{A,1}$ in the two independent copies. Note that A' generated the public keys pub_A and pub_B herself and can therefore decrypt the ciphertexts C'_A and C'_B to obtain the decommitments. Each copy of A' then verifies that the protocol concludes successfully (without any values being set to \perp) and that $K_A^{out} = K_B^{out}$. If the verification fails then A' aborts. Otherwise, the two copies of A' output $\beta'_{A,0}$ and $\beta'_{A,1}$ respectively. It is now easy to verify that the success probability of A' conditioned on γ' is exactly $\Pr[\text{Equiv}_A | \gamma']$. Let us now calculate the overall probability of success of A' :

$$\begin{aligned} \Pr[A' \text{ wins}] &= \frac{1}{2} \sum_{\gamma'} \Pr[\text{Equiv}_A | \gamma'] \Pr[\gamma'] \\ &\geq \sum_{\gamma'} \left(\frac{\varepsilon_{\gamma'}^2}{2} - \frac{1}{2^{n+1}} \right) \Pr[\gamma'] \\ &\geq \frac{\mathbb{E}_{\gamma'}[\varepsilon_{\gamma'}^2]}{2} - \frac{1}{2^{n+1}} \\ &\geq \frac{\varepsilon^2}{2} - \frac{1}{2^{n+1}} \end{aligned}$$

where the first equality holds because A' has to guess correctly (through the coin c_1) which schedule among E_A and E_B is followed, and the last inequality holds because $\mathbb{E}[\varepsilon_{\gamma'}^2] \geq (\mathbb{E}[\varepsilon_{\gamma'}])^2 = \varepsilon^2$.

C.2 Non-Malleability

Inputs: A security parameter 1^n , an infinite sequence of keys $\hat{K}_A \in \{0, 1\}^n$ and $\hat{K}_B \in \{0, 1\}^n$.

Setup: $PUB, s \leftarrow_{\text{R}} \text{Elnit}(1^n)$. Set PUB as the public reference string.

Phase 1: Generate a tuple γ of the form $(PUB, pri_A, pub_A, \alpha_A, pri_B, pub_B, \alpha_B, r_{adv})$ where α_A and α_B are the non-adversarial commitments of Alice and Bob respectively, and r_{adv} are the random bits used by the adversary.

Phase 2: Starting from γ simulate the interaction of A_{ke} with the protocol to completion. Suppose that E_A occurs (the description of the simulator for the case when E_B occurs is symmetric), and let $K'_A, K'_B \in \{0, 1\}^n \cup \{\perp\}$ be the values obtained by decommitting the adversarial commitments α'_A and α'_B using β'_A and β'_B respectively, and let $view$ be the view of A_{ke} at the end of the execution. Sample a new pair of output keys (\hat{K}_A, \hat{K}_B) .

Phase 3: The simulator proceeds according to the following cases:

1. $K'_B = \perp$ and $K'_A = \perp$: Output $view$.
2. $K'_B \neq \perp$ and $K'_A = \perp$: Let $\gamma' = (\gamma, \alpha'_A, \beta_B, \alpha'_B)$. Fix \hat{K}_B . Repeat the following: sample new \hat{K}_A ; set $K_A = \hat{K}_A \oplus K'_B$; $\beta_A \leftarrow_{\text{R}} \text{EDecom}(s, \alpha_A, K_A)$; conclude the protocol. Let $(K'_{B,i}, K'_{A,i}, view_i)$ be the keys decommitted to by A_{ke} and the view of A_{ke} during repetition i . If for any i , $K'_{B,i} \notin \{K'_B, \perp\}$ abort. Stop repeating when $(K'_{B,i}, K'_{A,i}) = (K'_B, \perp)$. Output $view_i$.
3. $K'_A \neq \perp$: Let $\gamma' = (\gamma, \alpha'_A)$. Repeat the following: sample new (\hat{K}_A, \hat{K}_B) ; set $K_B = \hat{K}_B \oplus K'_A$; $\beta_B \leftarrow_{\text{R}} \text{EDecom}(s, \alpha_B, K_B)$; conclude the protocol. Let $(K'_{B,i}, K'_{A,i}, view_i)$ be the keys decommitted to by A_{ke} and the view of A_{ke} during repetition i . There are three sub-cases:
 1. $K'_{A,i} \neq K'_A$: If $K'_{A,i} \neq \perp$ abort. Otherwise, repeat.
 2. $K'_{A,i} = K'_A$ and $K'_{B,i} = \perp$: Terminate and output $(\hat{K}_A, \hat{K}_B, view_i)$.
 3. $K'_{A,i} = K'_A$ and $K'_{B,i} \neq \perp$: Let $\gamma'' = (\gamma, \alpha'_A, \beta_B, \alpha'_B)$. Fix \hat{K}_B . Repeat the following: sample new \hat{K}_A ; set $K_A = \hat{K}_A \oplus K'_{B,i}$; $\beta_A \leftarrow_{\text{R}} \text{EDecom}(s, \alpha_A, K_A)$; conclude the protocol. Let $(K'_{B,j}, K'_{A,j}, view_j)$ be the keys decommitted to by A_{ke} and the view of A_{ke} during repetition j . If for any j , $K'_{B,j} \notin \{K'_{B,i}, \perp\}$ or $K'_{A,j} \notin \{K'_A, \perp\}$ abort. Stop repeating when $(K'_{B,j}, K'_{A,j}) = (K'_{B,i}, K'_A)$. Output $view_j$.

Fig. 4. The Simulator Sim

In this section we prove Theorem 2. The structure of the proof can be summarized as follows: we describe a simulator Sim and show that it terminates in expected polynomial time. The simulator Sim makes use of an additional ability, which is not available to a simulator in the non-malleability experiment ExpCKENMal : to fix one of the output keys in the sampled random key pairs and keep randomizing the other key. We then describe a variant Sim' of Sim , for which we have no bound on the running time, but it is guaranteed to produce a perfect simulation of the view of the adversary. We then show that the outputs of the first and second simulators are statistically indistinguishable unless a certain bad event occurs. If the bad event occurs with high probability then we can create an adversary that breaks the non-malleability of the commitment scheme \mathcal{COM} with a similar probability. Finally, we describe how to convert Sim to a simulator \widehat{Sim} that does not lock output keys and is compatible with ExpCKENMal . Combining these steps, we get that our simulator \widehat{Sim} performs a statistically indistinguishable simulation of the interaction of the adversary with the protocol, and it does so in expected polynomial time.

The complete description of the simulator is given in Figure 4. Our analysis of Sim consists of three parts. First, we describe a modified variant Sim' of the simulator Sim which never aborts. Instead, in the cases where Sim aborts, Sim' enters another iteration of the loop. We then show that with all but negligible probability Sim and Sim' proceed identically. Otherwise, we can construct an adversary that breaks the 2-message strong non-malleability of the underlying commitment scheme. Then, we show that the distribution of the transcripts produced by Sim' is identical to the distribution

of the transcripts in the real interaction between the adversary and the protocol. Finally, we show that Sim runs in expected polynomial time. Combining the three claims together we obtain Theorem 2.

Hybrid Experiments. In order to compare the behavior of A_{mal} when given either real or simulated transcripts we define a hybrid experiment where the transcript is simulated, but the simulator never aborts. Instead, it keeps trying until it produces a transcript from the correct distribution. Specifically, we define the following experiments:

Experiment Hyb₁. Is the experiment $\text{ExpCKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, 1)$ where the transcript given to the distinguisher is generated by the simulator Sim .

Experiment Hyb₂. Is the hybrid experiment where the transcript is generated by a modified simulator Sim' , which is defined as follows. Sim' to proceed identically to Sim except that whenever Sim aborts, Sim' performs another iteration of the appropriate step of Sim . More precisely, when Sim would abort in step 2 of phase 3, Sim' performs another iteration of step 2. Similarly, when Sim aborts at the top level of step 3, Sim' performs another iteration of step 3, and when Sim aborts in step 3.3, Sim' performs another iteration of step 3.3.

Experiment Hyb₃. Is the experiment $\text{ExpCKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, 0)$ where the transcript given to the distinguisher is generated by an interaction of A_{ke} with the protocol.

We now show that the outcomes of experiments Hyb₁ and Hyb₂ are closely distributed, and that the transcripts produced by Sim' and by actual interactions of A_{ke} with the protocol are identically distributed. The closeness in both cases is statistical, therefore, we obtain that the outcomes of experiments Hyb₁ and Hyb₃ are close. Finally, we show that Sim runs in expected polynomial time.

Lemma 1 *Experiments Hyb₁ and Hyb₂ proceed identically with probability $\geq 1 - \varepsilon_{com-nmal}^{1/3}$.*

Proof. Let abort be the event that Sim aborts at some point during its execution, and let $\varepsilon = \Pr[\text{abort}]$. Clearly, unless abort occurs Sim and Sim' proceed identically. We describe an adversary Adv that uses Sim to break the 2-message strong non-malleability of the underlying commitment scheme with probability close to ε . The main difficulty stems from the fact that Sim may rewind A_{ke} many times, each time equivocating one of the commitments α_A or α_B to a different value. In contrast, our adversary essentially gets one chance to rewind in the strong non-malleability experiment (through the two independent copies of Adv). We are saved by the fact that an abort in Sim is always caused by a pair of transcripts that are independent from all other pairs of transcripts that Sim generates. Moreover, there are only two ways that such a pair of transcripts can be generated: first, by fixing a tuple of the form $\gamma' = (\gamma, \alpha'_A, \beta_B, \alpha'_B)$ and generating two independent continuations of γ' where in each continuation α_A is decommitted to a new random string \hat{K}_A . An abort that occurs due to transcripts generated this way can occur either in case 2 or case 3.3 of phase 3. The second type of pairs of transcripts is generated by fixing a tuple of the form $\gamma' = (\gamma, \alpha'_A)$ and generating two independent continuations of γ' where in each continuation *both* α_A and α_B are decommitted to new random strings \hat{K}_A and \hat{K}_B . An abort in case 3.1 of phase 3 is always caused by such a pair. Consequently, what our adversary does is to guess what type of transcript pair will cause the abort, and then use the two branches of the strong non-malleability experiment to generate the two transcripts.

Formally, our new adversary Adv proceeds as follows: given public parameters PUB and two commitments α_A, α_B , Adv generates (pri_A, pub_A) and (pri_B, pub_B) from the appropriate distributions, as well as random bits r_{adv} for A_{ke} . Adv then tosses a fair coin c as a guess about the type of transcript pair that will cause the abort. Adv also guesses whether the schedule E_A or E_B will be followed by the adversary. Below we describe the algorithm of Adv for the case when E_A is guessed. The case of E_B is symmetric. Assuming that Adv guessed that schedule E_A is followed, it proceeds according to the following cases:

Case 1: $c = 0$. In this case Adv will generate a pair of transcripts of the second type described above, where (γ, α'_A) are fixed and both α_A and α_B are decommitted to random strings in the two branches of Adv . Specifically, Adv simulates A_{ke} on inputs $PUB, pub_A, \alpha_A, pub_B, \alpha_B$. A_{ke} then has a choice whether to follow schedule E_A or E_B . If the adversary follows E_B then Adv aborts. Otherwise, A_{ke} submits a commitment α'_A . Adv then submits α'_A in the non-malleability experiment. At this point two parallel copies of Adv are run where copy i is given two decommitments $(\beta_{A,i}, \beta_{B,i})$ for (α_A, α_B) . Adv then proceeds to simulate a complete interaction of A_{ke} with the protocol by setting $\beta_A = \beta_{A,i}$ and $\beta_B = \beta_{B,i}$. Eventually A_{ke} submits a decommitment $\beta'_{A,i}$, which Adv outputs in copy i .

Case 2: $c = 1$. In this case Adv will generate a pair of transcripts of the first type described above, where α_B is decommitted to a single value in both branches, and α_A is decommitted to two independently chosen random strings. Adv starts by replacing the commitment α_B with a new commitment $\hat{\alpha}_B$ for a random $K_B \in_{\mathcal{R}} \{0, 1\}^n$. More precisely, Adv generates $K_B \in_{\mathcal{R}} \{0, 1\}^n$ and $(\hat{\alpha}_B, \hat{\beta}_B) \leftarrow_{\mathcal{R}} \text{Commit}_{PUB}(K_B)$. Adv then simulates A_{ke} with inputs $(PUB, \alpha_A, pub_A, \hat{\alpha}_B, pub_B)$ to obtain a commitment α'_A (again, if A_{ke} does not follow schedule E_A then Adv aborts). Adv then continues simulating A_{ke} using the decommitment $\hat{\beta}_B$ to obtain a commitment α'_B . At this point Adv guesses whether α'_A or α'_B will be equivocated by tossing a coin c' . If $c' = 0$, Adv submits α'_A in the non-malleability experiment. Otherwise, Adv submits α'_B . Two copies of Adv are then run in parallel where copy i is given a pair of decommitments $(\beta_{A,i}, \beta_{B,i})$. Adv ignores $\beta_{B,i}$ and proceeds to complete the interaction of A_{ke} with the protocol using decommitments $\beta_A = \beta_{A,i}$ and $\beta_B = \hat{\beta}_B$. At the end of the simulation Adv outputs the decommitment β'_A produced by A_{ke} if $c' = 0$, and β'_B if $c' = 1$.

This concludes the description of Adv (assuming schedule E_A). We now show that Adv succeeds in breaking strong non-malleability with a probability which is close to the probability of abort during an execution of Sim . Let r_{sim} be a random variable representing the number of repetitions performed by the simulator, where a repetition occurs each time Sim restarts in cases 2, 3, and 3.3 of phase 3. This is also the number of opportunities for the simulator to abort. From Lemma 2 we know that $\mathbb{E}[r_{sim}] = 3$. Let abort_k be the event that Sim aborts on or before the k th repetition. We then get by Markov's inequality that for every integer $k > 0$,

$$\Pr[\text{abort}] \leq \Pr[\text{abort} | r_{sim} < k] + \frac{\mathbb{E}[r_{sim}]}{k} = \Pr[\text{abort} | r_{sim} < k] + \frac{3}{k} \quad (2)$$

and so $\Pr[\text{abort}_k] \geq \Pr[\text{abort}] - 3/k$. We shall now relate the probability of abort_k occurring to the advantage of Adv in the strong non-malleability experiment. There are four cases where Sim aborts in phase 3:

1. Event E_2 : in case 2, due to equivocation of α'_B . Let γ' be the tuple fixed at the beginning of case 2, let δ be the continuation of γ' that was generated in phase 2, and let δ' be the continuation generated during the repetition that caused the abort in case 2. Then, δ and δ' are distributed identically. Moreover, δ and δ' have the same distribution as the transcripts generated by the two branches of Adv when $c = 1$ and $c' = 1$.
2. Event $E_{3.1}$: in case 3.1, due to equivocation of α'_A . Let γ' be the tuple fixed at the beginning of case 3, let δ be the continuation of γ' that was generated in phase 2, and let δ' be the continuation generated during the repetition that caused the abort in case 3.1. Then, δ and δ' are identically distributed and have the same distribution as the transcripts generated by the two branches of Adv when $c = 0$.
3. Events $E_{3.3a}$ and $E_{3.3b}$: in case 3.3, due to equivocation of either α'_A or α'_B . Let γ'' be the tuple fixed when case 3.3 was first entered, let δ be the continuation of γ'' that was generated at the

top level of case 3 (and caused Sim to enter case 3.3), and let δ' be the continuation generated during the repetition that caused the abort in case 3.3. Then, if the abort is caused by the rule $K'_{B,j} \notin \{K'_{B,i}, \perp\}$, δ and δ' have the same distribution as the transcripts generated by the two branches of Adv when $c = c' = 1$. Similarly, if the abort is caused by the rule $K'_{A,j} \notin \{K'_A, \perp\}$ then δ and δ' have the same distribution as the transcripts generated by the two branches of Adv when $c = 1$ and $c' = 0$.

Clearly, $\text{abort}_k = (\text{abort}_k \wedge E_2) \vee (\text{abort}_k \wedge E_{3.1}) \vee (\text{abort}_k \wedge E_{3.3a}) \vee (\text{abort}_k \wedge E_{3.3b})$, and so for some $E \in \{E_2, E_{3.1}, E_{3.3a}, E_{3.3b}\}$, $\Pr[\text{abort}_k \wedge E] \geq \Pr[\text{abort}_k]/4$. Let us now calculate the probability of success of Adv . An abort occurs due to a pair of transcripts that matches one of the cases listed above. We can think of Adv as guessing which pair of transcripts will cause the abort, and then generating the pair from the appropriate distribution. The adversary “guesses” correctly with probability $\geq 1/4k^2$ (that is, the values c, c' match the event E and the two repetitions where the abort causing transcripts were generated are guessed correctly). We therefore obtain that Adv succeeds with probability:

$$\Pr[\text{ExpUSNMAL}(1^n, 2, Adv) = 1] \geq \frac{\Pr[\text{abort}_k]}{16k^2} \geq \frac{\Pr[\text{abort}] - 3/k}{16k^2}$$

and so $\varepsilon \leq 16k^2\varepsilon_{com} + 3/k$. This expression is minimized when $k = \left(\frac{3}{32\varepsilon_{com}}\right)^{1/3}$, and we get $\varepsilon \leq \varepsilon_{com}^{1/3}$.

The following lemma shows that the modified simulator Sim' , which does not abort, performs a perfect simulation of the interaction of the adversary A_{ke} with the protocol.

Lemma 2 *Let $\langle A_{ke}, A_{mal} \rangle$ be any PPT adversary for the CKE non-malleability experiment ExpCKENMal . Then $\Pr[\text{ExpCKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, Sim', 0) = 1] = \Pr[\text{ExpCKENMal}(1^n, \mathcal{KE}, A_{ke}, A_{mal}, Sim', 1) = 1]$.*

Proof. To prove the lemma we show that the cases of phase 3 of the simulator partition the space of transcripts into disjoint sets. Then, we argue that each case is selected with the right probability, and that inside each case the actual outcome is sampled with the right probability, conditioned on the set. We start our analysis by noting that the tuple γ generated in phase 1 is fixed throughout the simulation, and is always part of the final simulated view. Also recall that a tuple γ determines which of the schedules E_A or E_B is followed. We describe the analysis for the case of E_A . The analysis for E_B is symmetric. The entire argument is concluded by noting that the schedules E_A and E_B partition the transcripts into disjoint sets, and that each of the schedules is followed with the right probability in views produced by the simulator.

For a given γ let us define $E_{\perp, \perp}, E_{*, \perp}, E_{\perp, *}, E_{*, *}$ to be the events that the simulator enters cases 1, 2, 3.2, and 3.3 respectively. These events commit the simulator to producing transcripts where the adversarial commitments (α'_B, α'_A) are decommitted to $(\perp, \perp), (K'_B, \perp), (\perp, K'_A)$, and (K''_B, K'_A) for some string K''_B (not necessarily equal to K'_B). It is therefore clear that the sets of transcripts produced in each sub-case are disjoint. We now consider each case separately:

Case 1: This case is followed when $K'_A = K'_B = \perp$ and causes the simulator to output the view *view* that was generated in phase 2. Since phase 2 is simply a simulation of a complete interaction of the adversary with the protocol, and because Alice and Bob output random keys when $E_{\perp, \perp}$ occurs, we get that for every outcome $\delta_0 = (\hat{K}_A, \hat{K}_B, \text{view})$, $\Pr_{\delta \leftarrow Sim'}[\delta = \delta_0 | E_{\perp, \perp}] = \Pr_{\delta \leftarrow Adv}[\delta = \delta_0 | E_{\perp, \perp}]$.

Case 2: In this case $K'_B \neq \perp$. Therefore, the output of Alice must be $K'_B \oplus K_A$. However, in phase 2 the simulator chose a random K_A , without first seeing K'_B . Consequently, it is unlikely that the value $K'_B \oplus K_A$, where K_A is Alice's ephemeral key from $view$, will be equal to any of the keys \hat{K}_A that the simulator can sample in polynomial time. The simulator therefore proceeds to resample K_A together with \hat{K}_A , while fixing both adversarial commitments and K_B . During this repeated sampling \hat{K}_A is sampled first and then K_A is set to $\hat{K}_A \oplus K'_B$. However, since \hat{K}_A and K_A are still independently chosen random strings the transcripts $view_i$ that are generated during the sampling are correctly distributed. The simulator terminates when $(K'_{B,i}, K'_{A,i}) = (K'_B, \perp)$, which according to Lemma 1 implies $\Pr_{\delta \leftarrow Sim'}[\delta = \delta_0 | E_{K'_B, \perp}] = \Pr_{\delta \leftarrow Adv}[\delta = \delta_0 | E_{K'_B, \perp}]$ for every $K'_B \in \{0, 1\}^n$.

The argument for cases 3.2 and 3.3 is similar and is omitted.

Next we show that the actual simulator Sim runs in expected polynomial time. The proof proceeds by standard calculation. In particular, we show that each case in phase 3 of the simulation will require only a constant number of repetitions in expectation.

Lemma 3 *Let t_{prot} be an upper bound on the running time of the protocol (with commitments generated using the equivocal algorithms) and adversary, and let t_{sim} be a random variable representing the running time of Sim . Then, $\mathbb{E}[t_{sim}] \leq 3 \cdot t_{prot}$.*

Proof. We start by observing that $\mathbb{E}[t_{sim}] = \sum_{\gamma} \Pr[\gamma] \mathbb{E}[t_{sim} | \gamma]$, where γ is the tuple generated in phase 1. We shall now calculate $\mathbb{E}[t_{sim} | \gamma]$. Let us define $E_1, E_2, E_3, E_{3.1}, E_{3.2}, E_{3.3}$ to be the events that the simulator enters into the corresponding sub-case in phase 3. We further break down our calculation as follows:

$$\mathbb{E}[t_{sim} | \gamma] = \sum_{E \in \{E_1, E_2, E_3\}} \Pr[E | \gamma] \mathbb{E}[t_{sim} | E, \gamma] \quad (3)$$

For E_1 , there are no repetitions and so $\mathbb{E}[t_{sim} | E_1, \gamma] = 1$. For E_2 , the expected running time is as follows:

$$\begin{aligned} \mathbb{E}[t_{sim} | E_2, \gamma] &= \sum_{\gamma'} \frac{\Pr[\gamma' | E_2, \gamma]}{\Pr[E_2 | \gamma']} = \sum_{\gamma'} \frac{\Pr[E_2 | \gamma', \gamma] \Pr[\gamma' | \gamma] / \Pr[E_2 | \gamma]}{\Pr[E_2 | \gamma']} \\ &= \sum_{\gamma'} \frac{\Pr[\gamma' | \gamma]}{\Pr[E_2 | \gamma]} = \frac{1}{\Pr[E_2 | \gamma]} \end{aligned}$$

For E_3 the calculation is somewhat more involved due to the nested loop:

$$\begin{aligned} \mathbb{E}[t_{sim} | E_3, \gamma] &= \frac{1}{\Pr[E_3 | \gamma]} + \Pr[E_{3.3} | \gamma] \sum_{\gamma'} \frac{\Pr[\gamma' | E_{3.3}, \gamma]}{\Pr[E_{3.3} | \gamma']} \\ &= \frac{1}{\Pr[E_3 | \gamma]} + \Pr[E_{3.3} | \gamma] \sum_{\gamma'} \frac{\Pr[E_{3.3} | \gamma', \gamma] \Pr[\gamma' | \gamma] / \Pr[E_{3.3} | \gamma]}{\Pr[E_{3.3} | \gamma']} \\ &= \frac{1}{\Pr[E_3 | \gamma]} + \Pr[E_{3.3} | \gamma] \sum_{\gamma'} \frac{\Pr[\gamma' | \gamma]}{\Pr[E_{3.3} | \gamma]} = \frac{1}{\Pr[E_3 | \gamma]} + 1 \end{aligned}$$

Going back to (3) we get:

$$\mathbb{E}[t_{sim} | \gamma] = \Pr[E_1 | \gamma] + \Pr[E_3 | \gamma] + 2 \leq 3 \quad (4)$$

Which implies that $\mathbb{E}[t_{sim}] \leq 3t_{prot}$. We note that one might suspect that, due to the nested loop structure that triggers event E_3 in the simulator, the expected running time should have a quadratic term. This is not the case. Indeed, the external loop of phase 3, step 3 is never repeated once step 3.3 is entered, which avoids a quadratic term in equation (4).

This concludes the analysis of Sim . We are left with one final task: to achieve simulation in the setting where the simulator cannot lock the keys of Alice and Bob individually, but rather must randomize both keys at the same time. However, this is quite straightforward: We describe a new simulator \widehat{Sim} that uses Sim as a black box. \widehat{Sim} will run Sim many times. Each time, \widehat{Sim} will use its oracle to sample a new pair of outputs (\hat{K}_A, \hat{K}_B) for Alice and Bob and guess at which iterations Sim will lock each of the two keys. To simulate the key sampling and locking oracle of Sim , \widehat{Sim} will generate answers randomly except at the iterations where we guessed that Sim will lock a key. At these iterations \widehat{Sim} will use the appropriate key from the pair (\hat{K}_A, \hat{K}_B) . That is, if we guessed that Sim will lock Alice's output at iteration i then \widehat{Sim} will reply to Sim 's query with \hat{K}_A , and similarly for Bob. Formally, \widehat{Sim} works as follows:

The simulator \widehat{Sim} . Repeat the following steps until the first success:

1. Obtain a new pair of random output keys (\hat{K}_A, \hat{K}_B) .
2. Choose randomly two indexes $i, j \in_{\mathbb{R}} [n]$.
3. Simulate Sim for at most n repetitions as follows: at all repetitions $k \notin \{i, j\}$, if Sim asks for a new output key for Alice or Bob, choose one at random and use it as a response to Sim 's query. At repetition i , if Sim asks for a key for Alice and locks it then provide \hat{K}_A , otherwise restart (go back to step 1). Similarly, at repetition j if Sim asks for a key for Bob and locks it, provide \hat{K}_B . Otherwise restart.
4. If we haven't restarted and Sim has output a view $view$ within n repetitions, output $(\hat{K}_A, \hat{K}_B, view)$. Otherwise, restart.

Lemma 4 *The simulator \widehat{Sim} runs in expected polynomial time and produces transcripts from the same distribution as Sim .*

Proof (Proof Sketch). Using Markov's inequality and Lemma 2 we obtain that Sim terminates within n repetitions with probability at least $1 - 3/n$. \widehat{Sim} guesses the repetitions in which the keys are locked correctly with probability $1/n^2$. Therefore, the expected running time of \widehat{Sim} is $\frac{n^3}{n-3}$. Finally, we observe that the distribution of transcripts output by Sim does not depend on the repetition in which the transcript is output. In particular, unless Sim aborts, it always produces a perfect simulation of the adversary's view.

Lemmas 1, 2, 3, and 4 together give us Theorem 2.

D Universally Composable Credential-free Key Exchange

D.1 Background on Split Functionalities

We define what it means to realize an ideal functionality in an unauthenticated network without any setup. As sketched in the Introduction, realizing a functionality in the unauthenticated model is defined by requiring that the protocol actually realizes a relaxed variant of the functionality, called a "split functionality." (More motivational discussion on this choice appears there.) Recall that split functionalities enable the ideal-model adversary to split the uncorrupted parties into disjoint sets, called authentication sets, in an adaptive way. The parties in each authentication set H then run a

separate ideal execution with the trusted party where in each execution the adversary plays the roles of all the parties not in H .

A little more precisely, we assume that the parties know in advance the set U of parties with which they wish to interact. In each execution of an authentication set H the adversary plays the roles of the parties in $U - H$. (Throughout, we use the term “party P ” to mean “an ITI with PID P ”, where the SID is clear from the context.) We wish to make the following guarantees:

1. An authentication set must be fixed before any computation in the set begins (and thus an authentication set cannot be chosen on the basis of the inputs of the uncorrupted parties in that set);
2. The authentication sets are disjoint. This guarantees that all the parties in an authentication set have consistent views of the interaction. In particular, each party participates in only one execution.
3. The computation within each set is secure in the standard sense, i.e. as in the case that authenticated channels are assumed. In particular it is independent of the computations in other sets, except for the inputs provided by the adversary, which can be correlated to the outputs that it has received from computations with other authentication sets that have already been completed.

For simplicity, we assume that the authentication sets given by the adversary do not include corrupted parties. Alternatively, we modify the disjointment requirement to say that the intersection of any two authentication sets contains only corrupted parties.

Functionality $\mathbb{S}\mathbb{F}$

Given functionality \mathbb{F} , functionality $\mathbb{S}\mathbb{F}$ proceeds as follows:

Initialization:

1. Upon receiving an input $(\text{Init}, \text{sid})$ from a party P (i.e., from an ITI with SID sid and PID P), interpret $\text{sid} = (U, \text{sid}')$, where U is a set of IDs that includes P . Send $(\text{Init}, \text{sid}, P)$ to the adversary.
2. Upon receiving a message $(\text{Init}, \text{sid}, P', H, \text{sid}_H)$ from the adversary, verify that $H \subseteq U$, that $P' \in H$, and that for all previously recorded sets H' , it holds that either **(1)** $H \cap H'$ contains only corrupted parties and $\text{sid}_H \neq \text{sid}_{H'}$, or **(2)** $H = H'$ and $\text{sid}_H = \text{sid}_{H'}$. If any condition fails then do nothing. Otherwise, record the pair (H, sid_H) , output $(\text{Init}, \text{sid}, \text{sid}_H)$ to P' , locally initialize a new instance of the original functionality \mathbb{F} with session identifier sid_H , and provide this instance of \mathbb{F} , denoted \mathbb{F}_H , with the current set of corrupted parties. From now on let the adversary play the role of the parties in $U - H$ in \mathbb{F}_H .

Computation:

1. Upon receiving an input $(\text{Input}, \text{sid}, v)$ from party P' , find the set H such that $P' \in H$, and forward the message v from P' to \mathbb{F}_H . If no such H is found then ignore the input.
2. Upon receiving a message $(\text{Input}, \text{sid}, H, P', v)$ from the adversary, if \mathbb{F}_H is initialized and $P' \in U - H$, then forward v to \mathbb{F}_H as if coming from party P' . Otherwise, ignore the message.
3. When an instance \mathbb{F}_H generates an output v for party $P \in H$, output v to P . When the output is for a party $P' \in U - H$ or for the adversary, send the output to the adversary.

Corruption:

1. Upon receiving a $(\text{corrupt}, P)$ message from the adversary, record P as corrupted and forward this message to all currently active instances of \mathbb{F} .

Fig. 5. The split version of ideal functionality \mathbb{F}

The split functionality sF . We now proceed to formalize the above. Let \mathbb{F} be an ideal functionality. We define the relaxation of \mathbb{F} , called **split- \mathbb{F}** or sF , in Figure 5. For convenience, we let the SID of sF explicitly contain the list of parties (i.e., PIDs) that the initiator of this instance wished to interact with.

In the initialization stage of sF , the adversary adaptively chooses subsets of uncorrupted parties. (The adaptivity relates to the fact that an authentication set can be chosen and even a full execution completed, before the next authentication set is chosen). The adversary can choose any subsets that it wishes, as long as they are *disjoint* (or, rather, as long as the intersection of any two authentication sets contains only corrupted parties).

sF requires the adversary to provide a unique identifier, sid_H , for each authentication set H . This identifier is used to differentiate between the various copies of \mathbb{F} . Furthermore, this identifier is outputted explicitly to all the parties in this set. This is an important security guarantee: while the parties do not know, of course, which are the authentication sets, they have “evidence” of the set they are in. In particular, a global entity that sees the outputs of all parties can determine the authentication sets from the outputs alone. (From a technical point of view, this provision forces the adversary in the ideal process to mimic the same partitioning to authentication sets as the adversary in the real protocol execution.)

In the computation stage of the functionality, each set H is provided with a different and independent copy of \mathbb{F} . This means that each set H essentially runs a separate ideal execution of \mathbb{F} . In each such execution, the parties $P \in H$ provide their own inputs, and the adversary provides the inputs for all $P' \in U - H$. This reflects the fact that in each execution, the roles of the parties *outside* of the authentication set are played by the adversary. Similarly, the parties $P \in H$ all receive their specified outputs as computed by their copy of \mathbb{F} . The adversary receives all of its own outputs, as well as the outputs of the parties $P \in U - H$. We stress that there is no sharing of state between the different copies of \mathbb{F} run by sF .

Note that each instance of sF has a session identifier, sid , which is separate from the identifiers of the various authentication sets. This convention makes sure that all participants agree on the set U of intended participants. It is also consistent with the UC framework, which requires each instance of a functionality or a protocol to have its own unique identifier.

Also, when initializing a new copy of \mathbb{F} , functionality sF lets this copy know the current set of corrupted parties. This too is done for consistency with the UC framework, which models party corruption in the ideal process via special messages sent from the adversary to the ideal functionality.

D.2 UC Definition in The Split Functionalities Setting

Functionality \mathcal{F}_{CT}
<ol style="list-style-type: none"> 1. Upon activation with input (Init, sid) from party with PID P_0, where $sid = (P_0, P_1, sid')$, choose a value $k \leftarrow_R \{0, 1\}^n$, and forward (sid, k) to the adversary. 2. When receiving $(\text{Deliver}, sid)$ from the adversary, output (Init, sid) to the party with SID sid and PID P_1. 3. When receiving (Key, P) from the adversary, where $P \in \{P_0, P_1\}$, output (Key, k) to the party with SID sid and PID P.

Fig. 6. The Coin Toss functionality, \mathcal{F}_{CT} .

The secret coin toss functionality \mathcal{F}_{SCT} is identical to \mathcal{F}_{CT} except that the adversary does not learn the output value k .

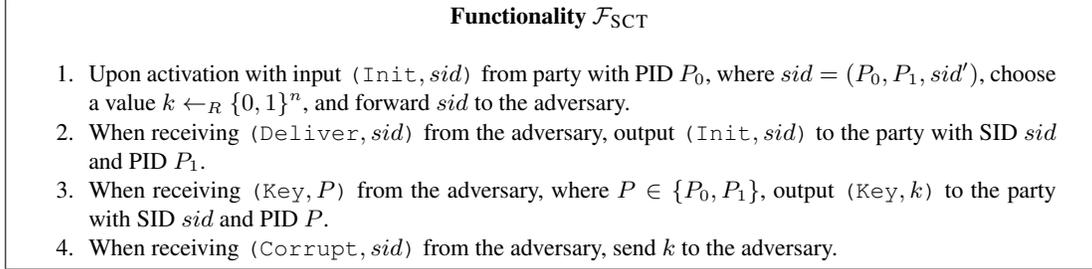


Fig. 7. The Secret Coin Toss functionality, \mathcal{F}_{SCT} .

In Figure 6 we review the commonly used definition of UC coin toss, and describe a simple modification that provides privacy of the output (given in Figure 7). The credential-less key exchange functionality $\mathcal{F}_{\text{CFKE}}$, given in Figure 8 is essentially a reformulation of the split version of \mathcal{F}_{SCT} – a coin toss functionality with privacy of the output. That is, the adversary can decide whether it wants to disrupt the interaction or to let it continue un-disrupted. If the interaction is not disrupted, then both intended parties obtain the same ideally random key, which remains unknown to the adversary. If the adversary decides to disrupt the interaction, then an independent random key is chosen for each one of the two parties. The adversary can then learn the key associated with each party, and then decide, based on the value of this key, whether to allow this party to obtain this key as output. When either party is corrupted, the adversary learns the corresponding key.

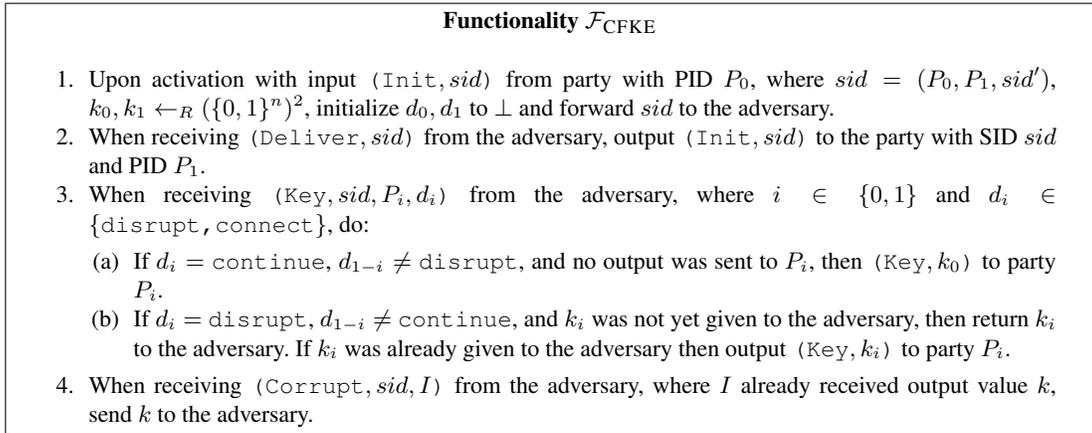


Fig. 8. The Credential-Less Key Exchange functionality, $\mathcal{F}_{\text{CFKE}}$.

D.3 A UC Secure CFKE Protocol

In this section we sketch a very simple transformation using existing tools that implements the UC version $\mathcal{F}_{\text{CFKE}}$ of CFKE. The UC based solution stems from the following observation. A coin

toss protocol where the messages are encrypted and authenticated has both properties that we need: (i) the outcome is hidden from the adversary, and (ii) if the adversary were to corrupt an honest party, he will still be unable to influence the outcome too much. The problem, of course, is that encryption and authentication are unavailable in our setting. This is where the compilers of [BCL⁺05] and [CCGS10] are very useful. Intuitively, a split functionality compiler transforms the “private coin toss” described above (and formally in Figure 7) into a protocol that achieves one of two scenarios: either the adversary allows the two honest parties to interact with a *single* \mathcal{F}_{SCT} functionality, or the adversary must interact with two *independent* functionalities $\mathcal{F}_{\text{SCT}}^A$ and $\mathcal{F}_{\text{SCT}}^B$, where honest party A interacts with $\mathcal{F}_{\text{SCT}}^A$ and party B with $\mathcal{F}_{\text{SCT}}^B$. Now the secrecy part of the functionality is no longer relevant (since the adversary plays the role of B when interacting with $\mathcal{F}_{\text{SCT}}^A$ and the role of A when interacting with $\mathcal{F}_{\text{SCT}}^B$). However, the coin toss part of the functionality guarantees that the key that adversary agrees on with A is a uniformly random string, independent from the random key on which he agrees with B .

For concreteness, if we instantiate Blum’s coin toss protocol with a recent UC commitment by Lindell [Lin11a], and apply the transformation of [CCGS10] to obtain a split functionality, we obtain a seven round $\mathcal{F}_{\text{CFKE}}$ protocol (in comparison to our first protocol, which requires two simultaneous rounds).