A PKI-based Framework for Establishing Efficient MPC Channels

Daniel Masny
Gaven Watson
VISA Research, USA

ABSTRACT

The Transport Layer Security (TLS) protocol is a fundamental building block for ensuring security on Internet. It provides an easy to use framework for the purposes of establishing an authenticated and secure channel between two parties that have never physically met. Nevertheless, TLS only provides a simple cryptographic functionality compared to more advanced protocols such as protocols for secure multiparty computation (MPC).

In this work, we provide a framework for efficiently establishing channels for MPC over the Internet. We focus on MPC protocols in the oblivious transfer (OT) hybrid model such that it is sufficient to establish OT correlations for such a channel. We revisit and combine different notions of UC security proposed in both the MPC and authenticated key exchange settings. Through this work, we show how an OT protocol can be composed with a secure authenticator to ensure the authenticity of messages sent during the OT.

In addition, we adapt and analyse non-interactive OTs based on dense key encapsulation mechanisms (KEMs) in the random oracle model, where the first message, i.e. public key, can be reused. These KEMs can be instantiated based on CDH, RSA and LWE and after a performance and security evaluation, it turns out that the resulting OT protocols are very competitive with the state of the art and are able to leverage existing PKIs.

CCS CONCEPTS

• **Security and privacy → Public key (asymmetric) techniques**.

KEYWORDS

MPC; PKI; Oblivious Transfer; Authentication; UC

ACM Reference Format:
Daniel Masny and Gaven Watson. 2021. A PKI-based Framework for Establishing Efficient MPC Channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3460120.3484806

1 INTRODUCTION

The Transport Layer Security (TLS) protocol [RD08, Res18] provides a robust and easy to use framework for the establishment of confidential and authenticated channels. Nevertheless, this protocol only provides a basic cryptographic functionality, that is, it enables two parties that trust each other to share confidential information while protecting said information from other non-trusted parties.

Secure multi-party computation (MPC) allows a more nuanced setting in which one does not want to entrust confidential information to the other participants in the protocol but still enable them to collaborate. This collaboration takes the form of computing a joint function on their own respective confidential inputs.

It is a natural task to build an efficient and easy to use framework that allows us to execute MPC protocols with arbitrary parties over the internet. Similar to the TLS setting, we cannot assume that authenticated or confidential channels between the parties are already established.

A common starting point for many MPC protocols is the OT-hybrid model in which parties have access to an oblivious transfer (OT) functionality [Yao82, Yao86, GMW87, Kil88, IPS08, IKO+11, CvT95, BL18, GS18]. An oblivious transfer (OT) [Rab81, EGL82], is a protocol between a sender and a receiver with the goal to establish OT correlations between them, i.e. the sender holds k_0, k_1 and the receiver b, k_b for a choice bit b. By the security of OT, k_{1-b} is only known to the sender, b only to the receiver and k_b to the sender and receiver but no other party.

To efficiently establish OT correlations that later could be used for an arbitrary MPC protocol in the OT-hybrid model, i.e. a channel for MPC, we need an efficient OT protocol. Unlike in a key-exchange protocol where only one key needs to be established for a confidential channel, an MPC protocol usually needs a large number of OT correlations. Fortunately, by using a technique called OT extension [Bea96, IKNP03, OOS17, ALSZ15, KOS15], it is sufficient to establish an amount of OT correlations that is identical to the security parameter. Nevertheless, to minimize the overhead compared to a key exchange, it is important to leverage synergies between the OT correlations we establish.

An efficient notion of OT is the notion of non-interactive OT by Bellare and Micali [BM90]. In their notion, the receiver generates a public key that depends on his choice bit. Given that the sender has access to this public key, he only needs to send a single message. Even more efficient are the OTs of Chou and Orlandi [CO15], a variant of Masny and Rindal [MR19], Canetti, Sarkar and Wang [CSW20] and McQuoid, Rosulek and Roy [MRR20], where [CO15] does not accomplish UC security [GIR17, HL17]. In these OT constructions, two rounds are sufficient to establish random OT correlations. The sender starts the protocol by sending a message to the receiver. This message is independent of the OT correlation which opens the possibility of reusing it for multiple OT correlations and hence, leverage synergies. This brings a significant advantage over [BM90] where using the same public key multiple times leads to OT correlations with the same choice bit. The OT of Gertner et al. [GKM+00] based on public-key encryption (PKE) with dense ciphertexts also has this property, though it is only semi-honest

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea.

ACM ISBN 978-1-4503-8454-4/21/11.
https://doi.org/10.1145/3460120.3484806

secure.

Security for MPC and OT is typically defined in the UC framework [Can01]. The advantage of the UC framework is that it allows us to universally compose protocols without negatively impacting security and it ensures maximal confidentiality for the inputs of an MPC protocol execution. The UC framework has also been used to model security for authenticated key exchange [BCK98, CSV16], which covers both, authenticity and confidentiality. The authenticity in UC is typically established using an authenticator. Such an authenticator transforms a protocol that is secure in a setting that assumes authenticated channels to a protocol that is secure even when the channels are not authenticated.

In practice, a public-key infrastructure (PKI) is needed for the purposes of establishing authenticated channels and it is a non-trivial task to integrate it into the UC framework. The work of Canetti, Shahaf and Vald [CSV16] have proposed a UC notion with PKI for the authenticated key exchange setting.

1.1 Our Contribution

We provide a framework with the goal to make it easier to use MPC protocols in the OT-hybrid model. We have two main objectives. The first objective is to construct an efficient and secure OT protocol that follows a TLS-like design and can leverage existing PKI infrastructure, i.e. by enabling the reuse of public keys. The second objective is to provide suitable security definitions for authenticating messages sent during the OT protocol.

NIOT. We start with the first objective for which the technical details can be found in Section 4. We consider two constructions of non-interactive OTs (NIOT) in which a sender generates a pk, sk pair that is independent of its input or generated OT correlations. Therefore, pk can be reused to generate independent OT correlations. Given that a receiver knows pk, he can generate a single message to generate the correlations. This changes the traditional paradigm of OT where usually the sender chooses two strings. In this notion of OT, [MR19] has called it receiver-chosen OT, the receiver can choose string k_b for his choice bit b. The string k_{1-b} is implicitly generated by R and is indistinguishable from uniform given the view of R.

The two constructions, that we refer to as OT_1, OT_2, use a cryptographic hash function and a one-way secure dense key-encapsulation mechanism. Dense KEMs are well known from CDH, RSA or LWE with a superpolynomial modulus to noise ratio. The latter is unfortunately not very efficient and hence we put more focus on the CDH and RSA-based dense KEMs.

In the CDH setting, OT_1 is a slight variation of an OT proposed by Masny and Rindal [MR19] and OT_2 is a slight variation of an OT proposed by McQuoid, Rosulek and Roy [MRR20]. Both variants have minimal overhead over [MR19, MRR20] and therefore remain competitive with the state of art. The two most efficient alternatives that allow a non-interactive OT functionality are the OTs of Chou and Orlandi [CO15] and, Canetti, Sarkar and Wang [CSW20]. We provide a comparison in Figure 1. Different to [CO15, MR19, MRR20, CSW20], we also cover the setting where the public keys are reused across different sessions in our security model. We model this setting such that it closely aligns with the notion of UC with joint state (JUC) [CR03]. Proving the security in this setting is quite challenging. We prove it in the algebraic group model (AGM) [FKL18] under the discrete-log assumption. We provide the details in Theorems in Section 4 and proofs in Appendix D and E. We emphasize that the JUC security, where the joint state is the public key, allows composability with any other protocol that is JUC secure with the public key as joint state. Further, this is also compatible with the generalized UC (GUC) [CDPW07] modeling of PKIs by [CSV16] such that the public key could be registered in a PKI and used for different protocols that are GUC secure or JUC secure with the public key as joint state. Further, this also allows to leverage existing PKIs and reuse registered public keys for the OT_1, OT_2 protocols. We refer for more details of the compatibility of JUC and GUC to the technical overview.

	UC	JUC	CRS	Com	Mul	GH
[CO15]	✗	✗	✗	$\log \lvert\mathcal{G}\rvert$	3	0
[MR19]	iDDH	?	✗	$2\log \lvert\mathcal{G}\rvert$	4	3
[CSW20]	DDH	?	✓	$2\log \lvert\mathcal{G}\rvert$	5	2
[MRR20]	KA	?	✗	$\lvert\mathsf{ct}\rvert + \kappa$	-	-
[MRR20]	CDH	?	✗	$\log \lvert\mathcal{G}\rvert + \kappa$	4	3
OT_1	KEM	?	✗	$2\lvert\mathsf{ct}\rvert$	-	-
OT_2	KEM	?	✗	$\lvert\mathsf{ct}\rvert + \kappa$	-	-
OT_1	CDH	AGM	✗	$2\log \lvert\mathcal{G}\rvert$	4	3
OT_2	CDH	AGM	✗	$\log \lvert\mathcal{G}\rvert + \kappa$	4	3

Figure 1: We compare the OT_1 and OT_2 construction proposed in this paper which are based on [MR19, MRR20] with previous works in terms of UC security, UC with joint state (JUC), i.e. public-key reuse, whether it needs a CRS (CRS), communication (Com), elliptic-curve multiplications (Mul) and hash operations into the group (GH). For Com, Mul and GHash, we ignore the costs caused by the public key generation and communication. iDDH stands for interactive DDH [MR19].

Both OT_1 and OT_2 allow a very simple and efficient OT from a dense KEM based on RSA. RSA based OTs have to the best of our knowledge not received much attention even though the OT construction of McQuoid, Rosulek and Roy [MRR20] is generic enough to capture RSA based OTs. Since RSA is a viable option for TLS, we instantiate OT_1 and OT_2 based on RSA to give an alternative to cyclic group based OTs. Unfortunately, our results using the AGM do not translate to the RSA setting so that we do not have a formal security analysis for the JUC security of the RSA based OT_1 and OT_2. Nevertheless, we are unaware of any attack and it would be rather surprising given the generic nature of OT_1 and OT_2 that JUC security would hold in the cyclic group setting but not in the RSA setting.

We implement the CDH and RSA-KEM instantiations of OT_1 and OT_2. The details can be found in Section 5 which provides a fairer comparison to the works of Chou and Orlandi [CO15] and, Canetti, Sarkar and Wang [CSW20] than Figure 1. In the RSA setting, we are unaware of any similarly efficient OT protocol and thus we only provide its performance.

Authentication. Our second objective is to define secure authentication for our OT and realize it efficiently. Similar to TLS, we want

to use a public key infrastructure (PKI) for authentication. Since we want to provide secure channels for MPC protocols we shall use UC definitions, the de facto standard for MPC. We build on the UC definitions and work for authentication in key exchange protocols by Bellare, Canetti and Krawczyk [BCK98] and Canetti and Krawczyk [CK02] provide. In these prior works the authors propose the use of *authenticators* that can establish authenticated channels for any protocol.

More recent work in the UC model by Canetti, Shahaf and Vlad [CSV16] propose a technique to model PKI in the UC setting. Their work focused on the used of signatures but not on KEMs and message authentication codes (MACs). We follow their approach by defining certification functionalities that make it easy to build authenticators. We realize these certification functionalities based on signatures or the KEM and MAC-based authentication mechanism. Further, we define a session certification functionality that allows to certify a whole session rather than individual messages. This enable modeling of an efficient KEM and MAC-based approach, where using a single MAC is more realistically. Our analysis uses techniques and ideas from the JUC framework.

Our final authenticators follow the standard approach of authentication based on either signatures or, KEMs and MACs as proposed in [BCK98] (using a nonce to prevent replay attacks). A key differentiator between this paper and prior works, is that we analyze their security in a more realistic PKI-based setting and where the same MAC key may be used for a whole session. Our more modular approach allows us to analyze the security of unilateral and mutual authentication, where previous works using the UC model typically only consider mutual authentication. We also discuss the differences between implicit and explicit authentication in our setting. In key exchange, upgrading implicit authentication to explicit authentication can be done using a key confirmation step [FGSW16]. However, in the OT setting this is much more challenging. We do not have a solution that accomplishes key confirmation in the OT setting that is sufficiently efficient to make it viable compared to straightforward authentication based on signatures. We provide the technical details in Section 3.

We emphasize that KEM and MAC-based authentication can be very attractive in combination with the KEM based OT_1 and OT_2 protocols since significant synergy effects can be leveraged. We analyze the GUC security where the KEM public keys are organized in a PKI. Since the GUC and JUC notions are compatible in our context, we can use the same public key for both, the authentication mechanism and the OT construction.

1.2 Related Works

Concurrent to our work, McQuoid, Rosulek and Roy [MRR21] addressed the issue when reusing the first message when generating a batch of OT correlations. They focus on OTs in cyclic groups and found that the OTs of [MR19, MRR20, CSW20] can be securely batched if the hash that defines the final OT string includes a counter. They prove security based on the oracle Diffie-Hellman (ODH) assumption which holds in the generic group model combined with the ROM. If this counter is not included, a malicious party could force the OT correlations of the whole batch to be identical and in this case, an OT extension using this batch would be insecure. In this sense, [MR19, MRR20, CSW20] are not secure when reused in a naive manner.

Compared to their result, we base the security on the discrete logarithm in the algebraic group model combined with the ROM which is weaker than the generic group model combined with the ROM. In our setting, where the first message serves as a public key which is e.g. registered in a PKI, the counter solution does not work straightforwardly. We resolve the issue by including the public key and intermediate transcript in the hash that defines the final OT strings. This does not immediately guarantee that the OT correlations are different when used in a batch, though transcripts that force OT correlations to be identical can be efficiently identified and then rejected. Our ideal NIOT functionality reflects that OT correlations could be force to be identical by allowing an ideal adversary to indicate that the OT correlations will be identical to a previous session such that the ideal functionality will reuse these correlations.

1.3 Technical Overview

Authenticators for UC secure Key Exchange. Bellare, Canetti and Krawczyk [BCK98] propose a UC definition for authenticating key exchange as well as other protocols. They define the AM and UM model. In the AM model, all channels are authenticated while in the UM model, an adversary is allowed to drop, inject or alter messages. They construct authenticators that can transform any protocol secure in the AM model to a protocol that is secure in the UM model. Their authenticators use either a signature or a KEM (to exchange a MAC key) and for any message from a party A to a party B, party B first sends a random nonce. After receiving the nonce, party A certifies the message and nonce using either a signature scheme or MAC and sends the message and certificate. B can now easily check the authenticity of the message. The random nonce plays a crucial role since it prevents replay attacks.

Authentication in the Global-PKI Model. Canetti, Shahaf and Vlad [CSV16] proposed a UC model for authentication with a global functionality that models a PKI using the global UC (GUC) model [CDPW07]. The works of [BCK98, CK02] do not allow such a global functionality. In their work, Canetti et al. do not use authenticators but instead a certification functionality that model signature schemes with a public key certified by a certification authority (CA) in a PKI. To authenticate a message, they sign the message and the session id, a unique id for each session which is used in the UC model to identify a session. This allows them to authenticate messages more efficiently than prior works [BCK98, CK02]. Though it also has the drawback that the availability of such a unique session id seems to be a very strong assumption in practice. One way to generate such a session id would be secure coin tossing which would incur a significant overhead.

Authenticators in the global-PKI model. We revisit these previous works [CSV16, BCK98, CK02] in order to build authenticators secure in a global-PKI model. In order to achieve this we must adjust the previous definitions appropriately. We adapt the certification functionality of [CSV16] such that it does not prevent replay attacks on its own. We also define a session certification functionality using the JUC model [CR03]. This models a KEM and MAC based

certification process where the same MAC key is used for all messages during a session. In Section 3.1, we show how to realize the session certification based on a CPA secure KEM and a MAC when for every session new certified KEM keys are used. In Appendix C, we show how to use a signature to realize message certification and a CCA KEM and a MAC to realize session certification with the same certified KEM key for multiple sessions. Further, we show that this can also be accomplished when replacing the CCA secure KEM with a CDH based KEM where the security holds in the AGM.

The certification functionalities are global functionalities that can be used in a straightforward way to build authenticators. The realization of our authenticators follows the proposed constructions in [BCK98] by using a nonce rather than a session id. This is less convenient but a more realistic countermeasure against replay attacks and used by TLS in a similar fashion. Other than [BCK98], we define a hybrid model between the AM and UM model that we denote with AM^+. In the AM^+ model, the adversary is only allowed to drop, inject or alter messages sent by specific parties but not necessarily all parties. This allows us to provide a model to prove unilateral authentication which is in some settings sufficient. We also discuss implicit authentication for OT. Other than in the key exchange setting, it seems not straightforward to upgrade it efficiently to explicit authentication by using key confirmation [FGSW16].

Receiver based Non-Interactive OT. Our goal for the OT protocol is to accomplish an ideal OT functionality that we call $\mathcal{F}_{\mathsf{mNIOT}}$. $\mathcal{F}_{\mathsf{mNIOT}}$ consists of two phases. In the first phase, a sender sends a query to the functionality. In an actual protocol, the sender would send a public key during this phase. In the second phase, the receiver chooses a key k_b and sends b, k_b to the functionality. The functionality outputs $\mathsf{k}_0, \mathsf{k}_1$ to the sender.

OT from Dense KEMs. A dense KEM is a KEM for which encapsulations are uniformly random over some group from which we can efficiently sample elements. This allows us to hash into this group and the hash is indistinguishable from an encapsulation even given the secret key.

The OT_1 and OT_2 constructions follow the works of [MR19, MRR20] and use the concept of programmable once public functions (POPF) [MRR20]. On the high level, they work as follows. The sender samples a public, secret key pair $(\mathsf{pk}, \mathsf{sk})$ and sends pk to the receiver. The receiver generates a message from which two encapsulations $\mathsf{ct}_0, \mathsf{ct}_1$ can be derived and for which he can know at most one of the encapsulated keys denoted with k_b. ct_{1-b} will be a uniform and simulatable (in the ROM) group element.

In both constructions, the choice bit of the receiver is hidden, because ct_0 and ct_1 have the same distribution thanks to the density of the KEM. At the same time the key k_{1-b} is hidden from the receiver due to the security of the KEM. When arguing UC security, we replace the hash function with a random oracle. By observing the order of the random oracle queries, a simulator can extract the choice bit b of a malicious receiver. He additionally can extract k_b by using his secret key. Through programming the random oracle a simulator can control both encapsulations ct_0, ct_1 and therefore both keys when interacting with a malicious sender.

Dense KEMs from CDH, trapdoor functions (RSA) and LWE are well known and in case of CDH and RSA very efficient. In the LWE case, obtaining correctness is very costly and requires a superpolynomial modulus to noise ratio. Unfortunately, we cannot improve the efficiency using a reconciliation mechanism in our case, since this mechanism is not known to be simulateable, especially not when the secret key or random coins of the encapsulation are known to a corrupted party.

Multi-use of Public Keys. An advantage of our OT is that the sender's message can potentially be reused. This however can create security complications since it allows a malicious receiver R^* to open several sessions under the same public key. When arguing security against such a receiver, we would like to replace encapsulation ct_{1-b} with a challenge encapsulation ct^* to argue that he cannot learn the key k_{1-b}. Unfortunately, UC security allows R^* to open a fresh session under the same public key and send ct^* to the sender who will decapsulate it, send it to the environment which then forwards it to R^*. Even if the second session is also simulated, R^* can choose a message such that ct^* is the encapsulation corresponding to his choice bit for this session. Thus, the simulator needs to extract k^* from ct^*. One solution might be to detect that ct^* is a challenge encapsulation and not decapsulate it faithfully. This does not seem to work, since all known dense KEMs have a strong form of malleability. R^* could alter ct^* to $\widehat{\mathsf{ct}^*}$ such that $\widehat{\mathsf{ct}^*}$ looks uniform given ct^* but encapsulates a key $\mathsf{k}^* + r$, where r is known to R^*. This security issue seems non-trivial to resolve. Therefore, even though a scheme might functionally allow key reuse, it is unclear whether the security still holds. To hedge our solution against such an attacker, we hash the encapsulated key together with parts of the transcript.

We prove this stronger notion of security, i.e. JUC with public keys as joint state, when instantiating OT_1 or OT_2 with a CDH-based KEM in the algebraic group model (AGM) [FKL18] in Section 4 and Appendices D and E. The AGM is weaker than the generic-group model and security typically does not hold unconditionally. In our case, security holds under the discrete-log assumption which is equivalent to CDH in the AGM. The AGM imposes a strong constraint on potential adversaries, especially in our case where we assume in addition a random oracle that hashes into the group. This helps us to prove security when public keys are reused. A key difference is that now, we are able to decapsulate encapsulations generated by R^* without using the secret key as well as detect altered challenge encapsulations.

Composability of our JUC and GUC results. We emphasize that in our setting, we consider JUC model where the joint states are public keys and therefore can be easily translated to the GUC setting in a similar fashion as modeling PKI in GUC. The public keys, i.e. joint states, could be registered in a global bulletin board functionality as proposed by [CSV16] in the context of authentication. Whenever a protocol reuses a joint state, it would simply request it from the global bulletin board. Since the joint states are public keys and part of the public transcript, publishing them in a bulletin board does not affect the JUC security of the protocols. This simple observation also explains why the proofs of GUC security of the session authentication using the CDH KEM in the AGM and the JUC security of OT_1 and OT_2 instantiated with the CDH KEM in the AGM follow a very similar approach.

As a consequence, we can use the same PKI or global bulletin board to store public keys. By the global composability, a public key can be used and reused for different authentication methods and OT protocols simultaneously. This however, might require to instantiate protocols with different random oracles when the protocols are not proven secure in the global random oracle model as it is the case for OT_1 and OT_2.

2 PRELIMINARIES

2.1 Notation

Let κ denote the security parameter. For $n \in \mathbb{N}$, we define the following set $[n] := \{1, \ldots, n\}$. By $x \leftarrow X$ we denote the the sampling of x uniformly from set X. We use $\Pi^{\mathsf{A},\mathsf{B}}$, Π when A and B are clear from the context, to denote a protocol between two parties A and B. For an algorithm $\mathsf{A}(x; r)$, we use x to denote the input and r to denote the random coins. We use $(\mathsf{A}(a), \mathsf{B}(b))_\Pi$ to denote the joint output distribution of A and B when interacting in protocol Π with inputs a and b respectively. For an ideal functionality $\mathcal{F}$ and malicious party $\mathcal{A}$, we denote $(\mathcal{F}, \mathcal{A})_{\Pi^{\mathsf{A},\mathsf{B}}}$ as the joint output of $\mathcal{A}$ and $\mathcal{F}$, where $\mathcal{A}$ produces the output of party A and $\mathcal{F}$ produces the output of party B respectively. When clear from the context, we omit $\Pi^{\mathsf{A},\mathsf{B}}$.

For a cyclic group of order p, we use the bracket notation to denote its elements, i.e. let g be a generator, then $[\![1]\!]_g := g$, $[\![a]\!]_g + b[\![1]\!]_g := g^{a+b}$. We omit g when clear from context.

Definition 2.1 (Random Oracle). *A* random oracle *over a set of domains and an image is a collection of functions* H *that maps an element* q *within one of the domains to a uniform element* $\mathsf{H}(q)$ *in the image.*

2.2 Key Encapsulation Mechanism

Definition 2.2 (Key Encapsulation Mechanism (KEM)). *A* Key Encapsulation Mechanism *consists of three ppt algorithms (*KGen, Enc, Dec*) and a key space* K *with the following syntax:*

KGen**:** *The key generation algorithm takes as input* 1^κ *and outputs a public-key secret-key pair* $(\mathsf{pk}, \mathsf{sk})$.

Enc**:** *The encapsulation takes as input* pk *and outputs an encapsulation* ct *and a key* k.

Dec**:** *The decapsulation takes as inputs* sk *and* ct *and outputs a key* k.

Further, we require correctness and one-way security (OW-CPA), indistinguishability under chosen-plaintext attacks (IND-CPA) or indistinguishability under chosen-ciphertext attacks (IND-CCA):

Correctness:

$$\Pr[\mathsf{k} = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})] \geq 1 - \mathsf{negl},$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$ *and* $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Enc}(\mathsf{pk})$.

OW-CPA **Security:** *For any ppt adversary* $\mathcal{A}$ *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathcal{A}(z, \mathsf{pk}, \mathsf{ct}) = \mathsf{k}] \leq \mathsf{negl},$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$ *and* $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Enc}(\mathsf{pk})$.

IND-CPA **Security:** *For any ppt distinguisher* D *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathsf{D}(z, \mathsf{pk}, \mathsf{ct}, \mathsf{k}) = 1] - \Pr[\mathsf{D}(z, \mathsf{pk}, \mathsf{ct}, u) = 1]| \leq \mathsf{negl},$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$, $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Enc}(\mathsf{pk})$ *and* $u \leftarrow \mathsf{K}$.

IND-CCA **Security:** *For any ppt distinguisher* D *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathsf{D}(z, \mathsf{pk}, \mathsf{ct}^*, \mathsf{k}) = 1] - \Pr[\mathsf{D}(z, \mathsf{pk}, \mathsf{ct}^*, u) = 1]| \leq \mathsf{negl},$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$, $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Enc}(\mathsf{pk})$ *and* $u \leftarrow \mathsf{K}$. *Further,* D *has access to a decapsulation oracle* $\mathsf{Dec}(\mathsf{sk}, \cdot)$ *that allows him to decapsulate arbitrary encapsulations except* ct^*.

Definition 2.3 (Dense KEM). *We call a KEM dense over a group* $\mathcal{G}$ *iff for any ppt distinguisher* $(\mathsf{D}_1, \mathsf{D}_2)$ *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathsf{D}_2(z, \mathsf{st}, \mathsf{ct}) = 1] - \Pr[\mathsf{D}_2(z, \mathsf{st}, u) = 1] \leq \mathsf{negl},$$

where $(\mathsf{st}, \mathsf{pk}) \leftarrow \mathsf{D}_1(1^\kappa)$, $(\mathsf{ct}, \mathsf{k}) \leftarrow \mathsf{Enc}(\mathsf{pk})$ *and* $u \leftarrow \mathcal{G}$.

We can construct a dense OW-CPA KEM from CDH, trapdoor functions and LWE with superpolynomial modulus to noise ratio (see Appendix A.4 for the definition of CDH, LWE and RSA). The three constructions are straightforward. Based on CDH, we get the following dense OW-CPA secure KEM for public parameters consisting of a cyclic group $\mathcal{G}$ of prime order p and a generator $[\![1]\!]$.

$\mathsf{KGen}(1^\kappa)$**:** Sample $a \leftarrow \mathbb{Z}_p \setminus \{0\}$. Output $\mathsf{pk} = [\![a]\!]$ and $\mathsf{sk} = a$.
$\mathsf{Enc}(\mathsf{pk})$**:** Sample $b \leftarrow \mathbb{Z}_p \setminus \{0\}$. Output $\mathsf{ct} = [\![b]\!]$ and $\mathsf{k} = b \cdot \mathsf{pk}$.
$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** Output $\mathsf{k} = \mathsf{sk} \cdot \mathsf{ct}$.

Correctness follows from $\mathsf{k} = \mathsf{sk} \cdot \mathsf{ct} = a \cdot [\![b]\!] = [\![ab]\!] = b[\![a]\!] = b \cdot \mathsf{pk}$. OW-CPA security immediately follows from the hardness of computing $[\![ab]\!]$ given $[\![a]\!]$, $[\![b]\!]$ (CDH).

Let F be a family of trapdoor functions that maps from an efficiently sampleable domain D into an efficiently sampleable group $\mathcal{G}$. Then our KEM is constructed as follows:

$\mathsf{KGen}(1^\kappa)$**:** Sample $(\mathsf{f}, \mathsf{f}^{-1}) \leftarrow \mathsf{F}$. Output $\mathsf{pk} = \mathsf{f}$ and $\mathsf{sk} = \mathsf{f}^{-1}$.
$\mathsf{Enc}(\mathsf{pk})$**:** Sample $x \leftarrow \mathsf{D}$. Output $\mathsf{ct} = \mathsf{f}(x)$ and $\mathsf{k} = x$.
$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** Output $\mathsf{k} = \mathsf{f}^{-1}(\mathsf{ct})$.

Correctness follows from the correctness of F and OW-CPA security follows from the one-wayness of F.

Finally, there is the following LWE-based dense IND-CPA secure KEM using a public parameter $A \leftarrow \mathbb{Z}_p^{m \times n}$.

$\mathsf{KGen}(1^\kappa)$**):** Sample $s \leftarrow \mathbb{Z}_p^n$ and a noise term $e \leftarrow \mathcal{X}$ for some noise distribution $\mathcal{X}$. Output $\mathsf{pk} = As + e$ and $\mathsf{sk} = s$.
$\mathsf{Enc}(\mathsf{pk})$**:** Sample $R \leftarrow \{-1, 0, 1\}^{m \times m}$. Output $\mathsf{ct} = RA$, $\mathsf{k} = \lfloor R\mathsf{pk} \rceil$.
$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** Output $\mathsf{k} = \lfloor \mathsf{ct} \cdot \mathsf{sk} \rceil$.

For achieving correctness, the rounding function $\lfloor \cdot \rceil$ needs to drop all lower significant bits to completely remove the noise. Then $\mathsf{k} = \lfloor \mathsf{ct} \cdot \mathsf{sk} \rceil = \lfloor RAs \rceil = \lfloor RAs + Re \rceil = \lfloor R\mathsf{pk} \rceil$. For the density property, we need that the leftover hash lemma applies. To apply this lemma, we need that A is uniform and $m > n \log p$. The requirement on the rounding function results in rather inefficient parameters and as a result we put less focus on this KEM in the rest of this paper.

2.3 Oblivious Transfer

We follow the simplified UC security framework which is sufficient for full UC security [CCL15]. Figure 2 defines the ideal NIOT functionality $\mathcal{F}_{\mathsf{NIOT}}$.

Definition 2.4 (Oblivious Transfer (OT)). *We call a protocol* Π *between two ppt parties, a sender* S *and a receiver* R*, a* oblivious

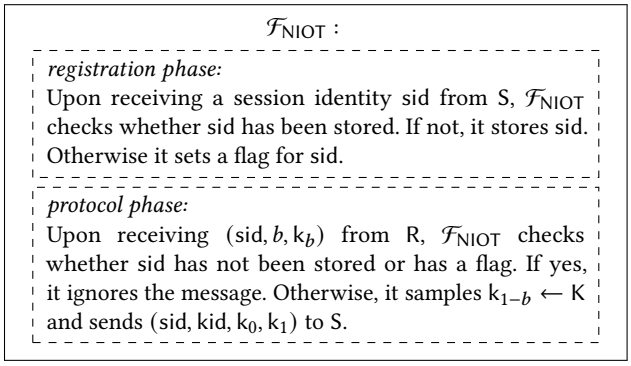

Figure 2: The figure shows the ideal $\mathcal{F}_{\mathsf{NIOT}}$ functionality. The registration phase and protocol phase happen within the same session.

transfer *if at the end,* S *outputs two strings* $(\mathsf{k}_0, \mathsf{k}_1)$ *and* R *outputs* $b \in \{0,1\}$ *and* k_b*. For security, we require two properties with respect to functionality* $\mathcal{F}_{\mathsf{NIOT}}$*.*

Security Against a Malicious Sender: *For any ppt adversary* $\mathcal{A}$*, there exists a ppt adversary* $\mathcal{A}$*' such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\mathsf{D}(z, (\mathcal{A}, \mathsf{R})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}', \mathcal{F}_{\mathsf{NIOT}})) = 1]| = \mathsf{negl},$$

where all algorithms receive input 1^{κ}*.* R *additionally receives input* $\mathbb{S}$*.*

Security Against a Malicious Receiver: *For any ppt adversary* $\mathcal{A}$*, there exists a ppt adversary* $\mathcal{A}$*' such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\mathsf{D}(z, (\mathsf{S}, \mathcal{A})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{F}_{\mathsf{NIOT}}, \mathcal{A}')) = 1]| = \mathsf{negl},$$

where all algorithms receive input 1^{κ}*.*

3 AUTHENTICATION

When studying OT protocols we typically assume that they are run over channels which ensure the integrity and safe delivery of message between the expected parties. However, when deploying such protocols in practice such channels may not exist. Building channels that provide these properties is a vital aspect in the study of secure key-exchange protocols. In fact, the Universal Composability (UC) framework of our security definitions thus far was initially defined with a view towards settings such as secure and authenticated channels, [BCK98, CK02]. Here we revisit this work and extend it to our setting.

First we recall the authenticated and unauthenticated links models, AM and UM respectively. In the AM model there exist n parties $P_1, ..., P_n$ running a protocol Π. An adversary is able to activate instances of Π between parties and control the corresponding communication. In this model it is assumed that the adversary will deliver messages faithfully, that is, if a message is intended for party P_i it will be delivered correctly to that party. In the UM model the adversary is permitted to drop, inject and alter the messages exchanged between parties. In the following, we use $\mathcal{A}_{\mathsf{AM}}$ or $\mathcal{A}$ to denote an adversary in the AM model and $\mathcal{A}_{\mathsf{UM}}$ for an adversary in the UM model.

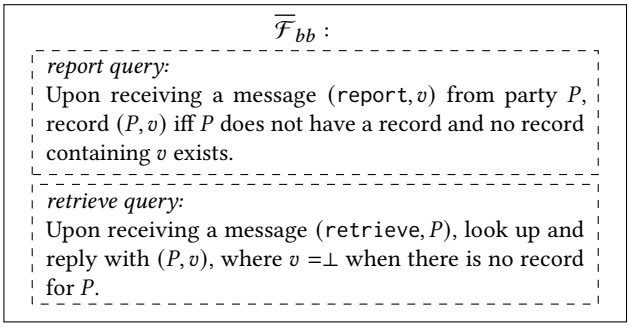

Figure 3: The figure shows the $\overline{\mathcal{F}}_{bb}$ functionality that models a PKI.

Definition 3.1. *For two protocols* Π *and* Π' *for* n *parties, we say that* Π' *emulates* Π *in the UM model if for any ppt adversary* $\mathcal{A}'_{\mathsf{UM}}$ *in the UM model, there exists a ppt adversary* $\mathcal{A}_{\mathsf{AM}}$ *in the AM model such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{AM}})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})_{\Pi'}) = 1]| = \mathsf{negl},$$

where all algorithms receive input 1^{κ}*.*

To transform any protocol secure in the AM model into one secure in the UM, we make use of an *authenticator* which provides the additional security guarantees required.

Definition 3.2. *[BCK98, Definition 2] A compiler* $\mathcal{C}$ *is an algorithm that takes as input descriptions of protocols and outputs descriptions of protocols. An* authenticator *is a compiler* $\mathcal{C}$ *where for any protocol* Π*, the protocol* $\mathcal{C}(\Pi)$ *emulates* Π *in the UM model.*

An authenticator ensures the authenticity of all protocol messages. In the following, we pursue a more modular approach in which we define a certification functionality enables the authentication individual messages. From the certification functionality, we can easily construct an authenticator.

3.1 Certification Functionalities from PKI

The model of Canetti et al. [CSV16] studies authenticated channels in the context of a global PKI. They define a global functionality $\overline{\mathcal{F}}_{bb}$ (see Figure 3) that they call a global bulletin board which models a PKI in the global UC (GUC) setting [CDPW07]. Further, they define a global certification functionality $\overline{\mathcal{F}}_{\mathsf{cert}}$ that allows us to certify and verify messages. We use an adapted version which is defined in Figure 4. $\overline{\mathcal{F}}_{\mathsf{cert}}$ can be instantiated using a signature scheme [CSV16]. Unfortunately, for their realization of $\overline{\mathcal{F}}_{\mathsf{cert}}$ based on signatures, they sign the session id sid. While this works fine in the UC model and helps to prevent replay attacks, in a real protocol the security crucially relies on the uniqueness of the session id among all sessions. This might not be easy to accomplish. One could use secure coin tossing to generate a string that functions as an sid but this causes an additional overhead.

We use sid only to link different parts of a protocol together such that the parties can link a message to a message in the protocol description. This brings the advantage that the security does not depend on the availability of a unique session id. The disadvantage of

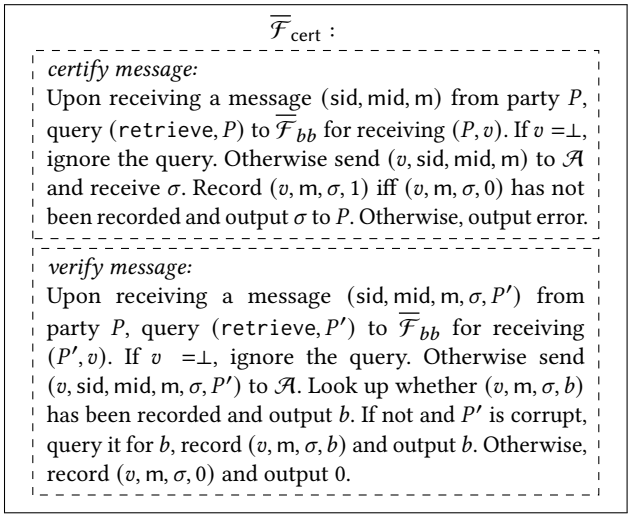

Figure 4: The figure shows the $\overline{\mathcal{F}}_{\text{cert}}$ functionality.

this approach is that $\overline{\mathcal{F}}_{\text{cert}}$ allows certifying messages but does not prevent replay attacks. We follow Bellare et al. [BCK98] to resolve this issue. We use authenticators that use message certification as a building block to achieve full security in the UM model. These authenticators prevent replay attacks by sending a challenge nonce N.

Using $\overline{\mathcal{F}}_{\text{cert}}$ for message certification leads to transferable or undeniable authentication since everyone can verify messages [CSV16]. In Figure 5, we define a session-based certification functionality $\overline{\mathcal{F}}_{\text{cs}}$. $\overline{\mathcal{F}}_{\text{cs}}$ does not have the transferability property since only the sender and receiver of a message can verify the certificate. We follow the UC with joint state (JUC) [CR03] model. In this model, there are subsessions that share a joint state. This is useful for an efficient KEM and MAC-based authenticator, where the same MAC key is used for authenticating multiple messages. Instead of calling it a subsession, we use a message id mid which is unique for each message. Again mid only serves as a method to link messages of a protocol together. In Figure 6, we show a protocol that realizes $\overline{\mathcal{F}}_{\text{cs}}$ based on a KEM and a MAC.

Theorem 3.3. *Let* Π *be the protocol in Figure 6,* KEM *be* IND-CPA *secure and correct and* MAC EU-CMVA *secure and correct. Then, for any ppt adversary* $\mathcal{A}$ *in the UM model, there exists a ppt adversary* $\mathcal{A}$*' in the UM model such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\text{D}(z, (\mathcal{A}_{\text{UM}})_{\Pi}) = 1] - \Pr[\text{D}(z, (\mathcal{A}'_{\text{UM}})_{\overline{\mathcal{F}}_{\text{cs}}}) = 1]| = \text{negl},$$

where all algorithms receive input 1^{κ}.

Proof. Given $\mathcal{A}$, we construct $\mathcal{A}$'. First notice that environment D can only distinguish $\mathcal{A}$ from $\mathcal{A}$' in two ways: First, when $\mathcal{A}$ creates a valid pair (m, τ) for which m, σ with $\sigma = \tau$ has not been recorded as $(v, w, \text{m}, \sigma, 1)$ by $\overline{\mathcal{F}}_{\text{cs}}$ (where v is a public key and w an encapsulation of the random coins of the MAC key generation). Second, when the correctness of KEM or MAC fails.

In case $\mathcal{A} = P_c$ or $\mathcal{A} = P_v$, constructing $\mathcal{A}'$ is easy. We let $\mathcal{A}'$ interact with $\mathcal{A}$. $\mathcal{A}'$ either chooses pk and sends it to $\mathcal{A}$ ($\mathcal{A} = P_v$)

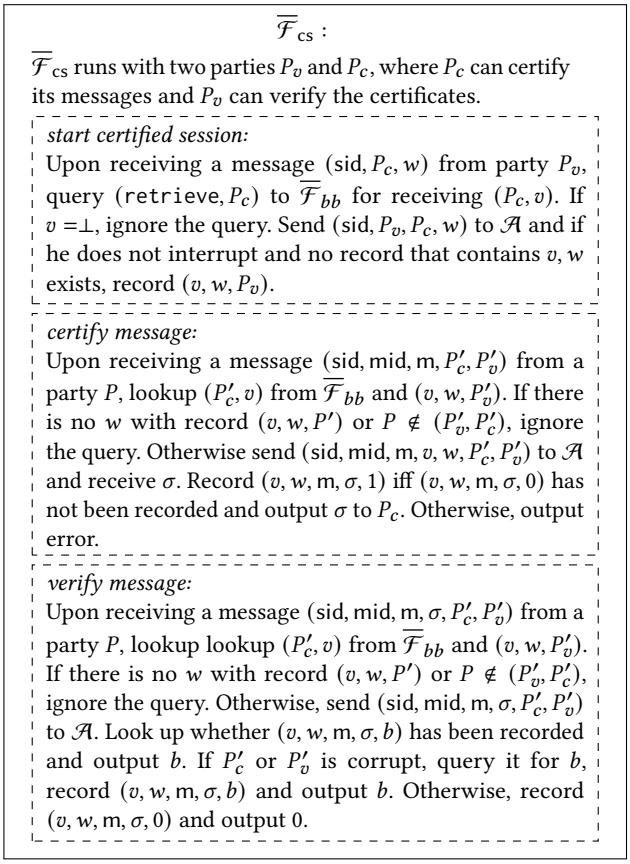

Figure 5: The figure shows the $\overline{\mathcal{F}}_{\text{cs}}$ functionality.

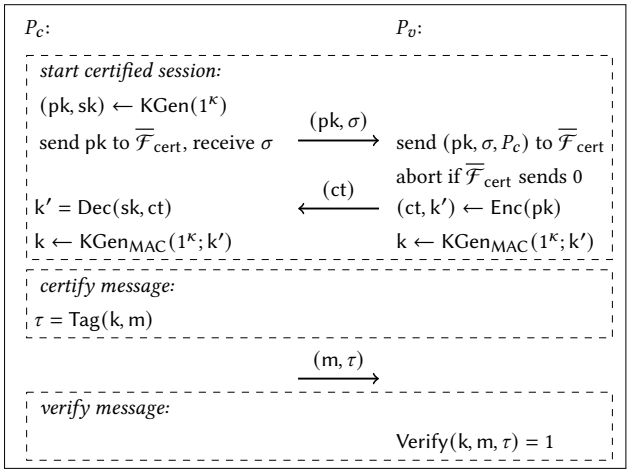

Figure 6: The figure shows a certified session based on a KEM, MAC and $\overline{\mathcal{F}}_{\text{cert}}$. For the sake of simplicity, we omit sid **and** mid**, which only serve to identify messages and link them to a session.**

or $\mathcal{A}'$ chooses and sends the encapsulation ($\mathcal{A} = P_c$). In both cases $\mathcal{A}'$ knows k, can verify (m, τ) and force $\overline{\mathcal{F}}_{\mathsf{cs}}$ to record $(v, w, \mathsf{m}, \tau, 1)$ when (m, τ) is valid. This only fails when the correctness fails.

The non-trivial case is when $\mathcal{A} \notin (P_c, P_v)$. In this case, we define a sequence of hybrids $\mathcal{A}'_1, \mathcal{A}'_2$. $\mathcal{A}'_1$ follows the protocol by following the description of P_c and P_c. $\mathcal{A}'_2$ is identical to $\mathcal{A}'_1$ except that he replaces ct, which encapsulates k, with an encapsulation of a uniform key k_u. Notice that pk is authenticated using $\overline{\mathcal{F}}_{\mathsf{cert}}$. Therefore $\mathcal{A}$ cannot manipulate pk.

Let KEM be ϵ_{KEM} IND-CPA secure, then, conditioned on $\mathcal{A} \notin (P_c, P_v)$,

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{UM}})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{2,\mathsf{UM}})_{\Pi}) = 1]| \leq \epsilon_{\mathsf{KEM}}.$$

Now, the communication transcript between P_c and P_v is independent of MAC key k and we can exploit the EU-CMVA security of MAC. In the EU-CMVA game, we can request arbitrary tags and win if we can forge a new tag. Our final adversary $\mathcal{A}'$ generates the start certified session messages as $\mathcal{A}'_2$ does with the exception that he looks up pk from $\overline{\mathcal{F}}_{bb}$. For the certify message and verify message phase, he uses the tag and verification queries in the EU-CMVA game to simulate P_c and P_v. More specifically, whenever P_c requests a tag for m from $\overline{\mathcal{F}}_{\mathsf{cs}}$, he makes a tag query for m and sends $\sigma = \tau$ to $\overline{\mathcal{F}}_{\mathsf{cs}}$. Further, he uses (m, τ) as message for $\mathcal{A}$. Whenever P_v makes a query (m, τ). $\mathcal{A}$' checks whether τ has been a response to a tag query. If not, he makes a verification query. If the verification fails, $\overline{\mathcal{F}}_{\mathsf{cs}}$ answers also with 0 since this query has not been recorded. If the verification succeeds, D might distinguish $\mathcal{A}'$ from $\mathcal{A}$ but $\mathcal{A}'$ breaks the EU-CMVA security.

Let MAC be ϵ_{MAC} EU-CMVA secure. Then, conditioned on $\mathcal{A} \notin (P_c, P_v)$,

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{2,\mathsf{UM}})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})_{\overline{\mathcal{F}}_{\mathsf{cs}}}) = 1]| \leq \epsilon_{\mathsf{MAC}}.$$

Let MAC and KEM both be δ-correct. Then

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{UM}})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})_{\overline{\mathcal{F}}_{\mathsf{cs}}}) = 1]| \leq \epsilon_{\mathsf{MAC}} + \epsilon_{\mathsf{KEM}} + 2\delta.$$

□

In Appendix C, we provide realizations of $\overline{\mathcal{F}}_{\mathsf{cert}}$ based on signatures and realizations of $\overline{\mathcal{F}}_{\mathsf{cs}}$ based on MACs and CCA secure KEMs or MACs and the CDH based KEM, where security is proven in the AGM.

3.2 Party-Specific Authenticators

In existing UC models for secure channels all parties are assumed to be equal. That is any party can have the role of the protocol initiator. In protocols for entity authentication it may be the case that we only wish to ensure unilateral authentication, or in the case of mutual authentication, the method of authentication in either direction may be different.

We extend the previous model for authenticated and unauthenticated channels to consider both unilateral and mutual authentication. Within a single session of a protocol, one party will be designated the role of initiator and the other the responder. In the case of unilateral authentication, we split the set of all parties into those that are authenticated and those that are unauthenticated. We

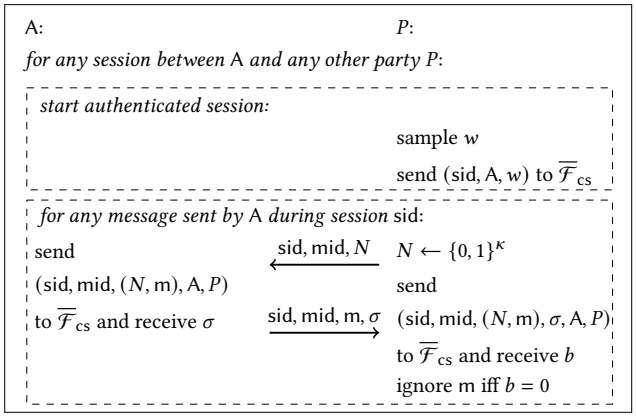

Figure 7: The figure shows an authenticator C^+_{A} that authenticates all messages sent by A during all sessions based on $\overline{\mathcal{F}}_{\mathsf{cs}}$.

all this model the AM^+ model. We use $\mathsf{AM}^+_{\mathbb{P}}$ to denote the setting in which the adversaries delivers the messages of parties $P \notin \mathbb{P}$ faithfully. For any party $P \in \mathbb{P}$, the adversary is allowed to drop, inject and alter messages send by this party.

First, we adapt Definition 3.1 to our AM^+ model.

Definition 3.4. *For two protocols Π and Π' between parties $(P_1, \ldots, P_n)$, we say that Π' emulates Π in the $\mathsf{AM}^+_{\mathbb{P}}$ model where $\mathbb{P} \subseteq (P_1, \ldots, P_n)$ if for any ppt adversary $\mathcal{A}$' in the $\mathsf{AM}^+_{\mathbb{P}}$ model, there exists a ppt adversary $\mathcal{A}$ in the AM model such that for any ppt environment D and any polynomial size auxiliary input z*

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{AM}})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{AM}^+_{\mathbb{P}}})_{\Pi'}) = 1]| = \mathsf{negl},$$

where all algorithms receive input 1^{κ}.

We define a party-specific authenticator following Definition 3.2 to transform any protocol secure in the AM model into one secure in the AM^+ model, we make use of an *authenticator* which provides the additional security guarantees required.

Definition 3.5. *A compiler $C^+_{\mathbb{P}}$ is an algorithm that takes for input descriptions of protocols and outputs descriptions of protocols. An* authenticator *is a compiler $C^+_{\mathbb{P}}$ where for any protocol Π with parties $(P_1, \ldots, P_n)$ secure in the AM model and $\mathbb{P} \subseteq (P_1, \ldots, P_n)$, the protocol $C^+_{\mathbb{P}}(\Pi)$ emulates Π in the $\mathsf{AM}^+_{\mathbb{P}}$ model.*

Notice that when $\mathbb{P} = (P_1, \ldots, P_n)$, the $\mathsf{AM}^+_{\mathbb{P}}$ model is equivalent with the UM model while when $\mathbb{P} = \emptyset$ it is equivalent with the AM model. In Figure 7 and Figure 8, we show party specific compilers for a party A. One authenticates individual messages while the other authenticates all messages sent by A during a whole session which causes less overhead. Both authenticators are basically the authenticators of [BCK98] adapted to our certification functionalities. As in [BCK98], the round complexity can be compressed by sending the nonce together with the previous message.

Theorem 3.6. *Let Π be a protocol between parties $(P_1, \ldots, P_n)$ that is secure in the $\mathsf{AM}^+_{\mathbb{P}}$ model where $\mathbb{P} \subseteq (P_1, \ldots, P_n)$. Let $\mathsf{A} = P_i$ for some $i \in [n]$ and C^+_{A} be one of the two authenticators defined in Figure 7 or 8. Then $C^+_{\mathsf{A}}(\Pi)$ emulates Π in the $\mathsf{AM}^+_{\mathbb{P}\cup\mathsf{A}}$ model.*

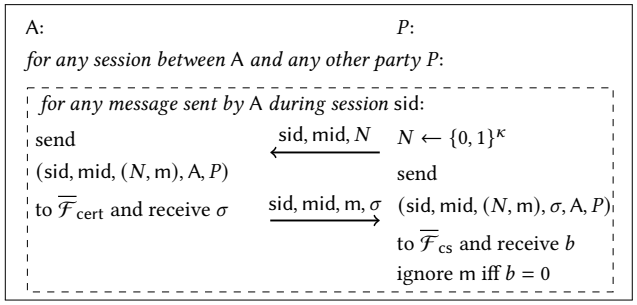

Figure 8: The figure shows an authenticator C^{+}_{A} that authenticates all messages sent by A during all sessions based on $\overline{\mathcal{F}}_{\text{cert}}$.

Proof. The difference between the $\mathsf{AM}^{+}_{\mathbb{P}}$ and $\mathsf{AM}^{+}_{\mathbb{P}\cup\mathsf{A}}$ model is that in the latter an adversary $\mathcal{A}$ can drop, inject or alter messages sent by A. The two models are equivalent, when $\mathcal{A} = \mathsf{A}$ since then he can drop, inject or alter messages of A in both models. Another trivial case is when $\mathcal{A}$ is the receiver of the messages from A. In this case he might accept an injected or altered message. Again, $\mathcal{A}$ can trivially alter or inject messages for himself in both models without breaking security. Therefore, we consider in the following the case when $\mathcal{A}$ is neither A nor the recipient of the message that he wishes to alter or inject.

Since in $C^{+}_{\mathsf{A}}(\Pi)$ all messages that do not have a certificate are ignored, $\mathcal{A}$ needs to either generate certificates for his injected or altered messages or he needs to reuse previous messages with a valid certificate. Let us first consider the authenticator in Figure 8. $\overline{\mathcal{F}}_{bb}$ uniquely binds v to A. Further, $\overline{\mathcal{F}}_{\text{cert}}$ only allows A to enter a record $(v, \mathsf{m}, \sigma, 1)$ through a certify or verify query. Thus, $\mathcal{A}$ cannot generate a new tuple m, σ that will not be ignored by a receiver of the message.

The receiver of a message from A will always sample a random nonce N from $\{0,1\}^{\kappa}$. Except with a negligible probability this nonce will collide with a previous nonce. The receiver will always expect a certificate for (m, N) which forces the adversary to generate a certificate for a new message (except when the nonce collides). Therefore, we have proven the theorem statement for the authenticator of Figure 8.

For the session-based authenticator in Figure 7, the theorem statement follows for similar reasons. Again, the receiver of the message samples a random nonce such that $\mathcal{A}$ is forced to generate a new message that contains this nonce together with a valid certificate. There are only two parties that can enforce a recording of $(v, w, \mathsf{m}, \sigma, 1)$ in $\overline{\mathcal{F}}_{\text{cs}}$. This must either happen through a certify or verify query to $\overline{\mathcal{F}}_{\text{cs}}$. The first party is the party that $\overline{\mathcal{F}}_{bb}$ ties to v which can only be A due to the uniqueness of v. The second party is the party P for which there is a record (v, w, P). There is only one such party which is the receiver of the message from A. Thus, it is straightforward to construct an adversary $\mathcal{A}'$ from $\mathcal{A}$ which can perform the same attack in the $\mathsf{AM}^{+}_{\mathbb{P}}$ model that $\mathcal{A}$ performs in the $\mathsf{AM}^{+}_{\mathbb{P}\mathsf{A}}$ model. □

3.3 Implicit Authentication for KEM-based OT

As mentioned in the Introduction, we provide a construction of OT for which the first message, which is sent from sender to receiver, can be reused. This first message will be the public key of the KEM. It is immediate that one might want to use a PKI to authenticate this public key. Since only the owner of this public key (and corresponding secret key) can decapsulate keys under this public key, this provides *implicit* authentication. In our case, the owner is the OT sender. When a receiver generates a message under a certified public key from a PKI, only the owner of the key can obtain the two strings of the sender. This implicit authentication is strictly weaker than the *explicit* authentication provided by an authenticator. An adversary might alter the OT message of the receiver such that the sender learns different OT strings. The sender will therefore have no guarantees that the correct message was delivered.

In the case of a standard KEM, explicit authentication can be achieved through the addition of a *key confirmation* step, where in the receiver sends back a message using the received key. Key confirmation has been widely studied in game-based definitional settings [FGSW16] but is not explicitly discussed with the UC framework. In the case of OT, key confirmation seems a challenging property to obtain. On one hand, the receiver has only one of the keys and revealing which one to the sender via key confirmation would break his security. On the other hand, the sender could confirm both keys without exposing them. Nevertheless, the receiver is only able to validate one of the two. This opens the possibility for selective abort (selective failure) attacks. The sender/owner (or a man-in-the-middle adversary) can send a faulty confirmation for one of the keys, as this is dependent on the choice bit it will either go unnoticed by the receiver (since he does not know the key), or the receiver will abort since he thinks the key confirmation has failed. As a result, we do not provide a key confirmation step for OT. Implicit authentication will however be enough for many applications. We discuss implicit authentication further in Appendix F.

4 NIOT FROM DENSE KEY ENCAPSULATION

4.1 UC Security

The OT_1 construction of NIOT from a dense KEM is depicted in Figure 9. In Figure 10, we depict the OT_2 construction. Theorem 4.1 and Theorem 4.4 establish their security with respect to ideal functionality $\mathcal{F}_{\mathsf{NIOT}}$. Both constructions can be instantiated with any of the dense KEMs from Section 2.2.

Theorem 4.1. *Given a dense* KEM *that is correct and* OW-CPA*-secure, then the* NIOT *in Figure 9 is UC secure with respect to* $\mathcal{F}_{\mathsf{NIOT}}$ *in the programmable random oracle model.*

The proof of Theorem 4.1 is in Appendix B.

Theorem 4.2. *Given a dense* KEM *that is correct and* OW-CPA*-secure, then the* NIOT *in Figure 10 is UC secure with respect to* $\mathcal{F}_{\mathsf{NIOT}}$ *in the programmable random oracle model.*

The NIOT in Figure 10 is almost identical to the protocol of [MRR20] with the difference, that we include pk and s, T in the hash computation. This does not have any impact on the proof in [MRR20] and we therefore refer to [MRR20, Theorem 11] for the proof of Theorem 4.2.

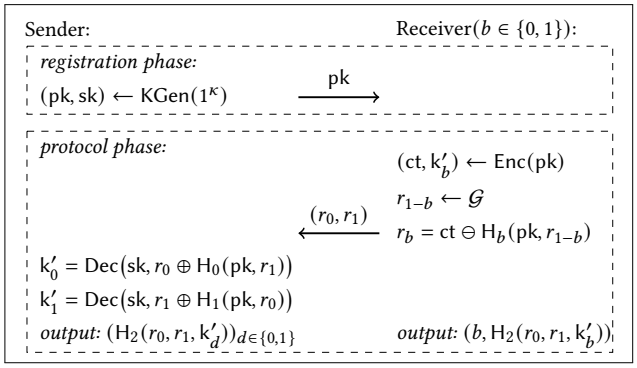

Figure 9: The figure shows the OT_1 protocol based on [MR19] using a dense KEM (KGen, Enc, Dec) **and random oracles H_0, H_1 that hash into a group $\mathcal{G}$ with operations $\oplus$, $\ominus$. H_2 is a random oracle from the keyspace K to $\{0,1\}^\kappa$. The protocol consists of a registration and a protocol phase.**

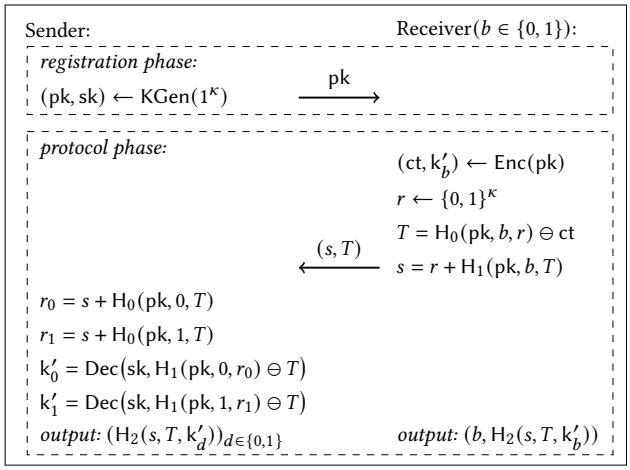

Figure 10: The figure shows the OT_2 protocol which is a variant of [MRR20] with the difference that it uses a dense KEM (KGen, Enc, Dec) **instead of a key agreement and that it includes the public key and the receiver's messages in the hash computations. It uses random oracles H_0, H_1 that hash into a group $\mathcal{G}$ with operations $\oplus$, $\ominus$. H_2 is a random oracle from the keyspace K to $\{0,1\}^\kappa$. The protocol consists of a registration and a protocol phase.**

4.2 Security under Public Key Reuse

We define a new ideal functionality for the setting where the same public keys are used across different sessions. We align our definition with the UC definition with a joint state (JUC) [CR03]. In our case the joint state is the public key which is being reused. As in JUC, a sender can start a joint state by publishing or transferring a public key. A receiver can then start a subsession under this public key to generate OT correlations. We define this functionality in Figure 11 and denote it with $\mathcal{F}_{\mathsf{mNIOT}}$. For the sake of simplicity, $\mathcal{F}_{\mathsf{mNIOT}}$ only considers a single receiver. Our security claims still hold when considering multiple receivers that are different entities.

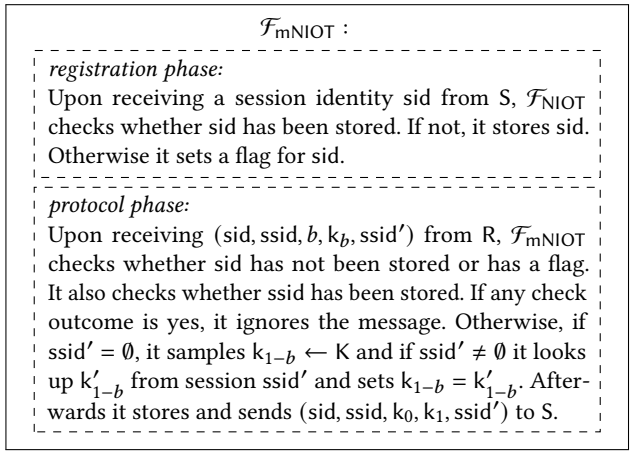

Figure 11: The figure shows the ideal $\mathcal{F}_{\mathsf{mNIOT}}$ functionality. For each session identity, the protocol phase can be executed multiple times and each protocol phase has its own subsession id (ssid**).**

This setting is slightly different from the global UC (GUC) setting [CDPW07] where all protocols can use a single instantiation of a global functionality. In our case, each party can run multiple copies of $\mathcal{F}_{\mathsf{mNIOT}}$ where each copy might have multiple subsessions. Nevertheless, our results in the JUC setting translate to the GUC setting when considering the public keys as part of a PKI that is modeled as a global bulletin board.

In Theorem 4.3, we show that the construction in Figure 9 instantiated with the CDH KEM of Section 2.2 is secure with respect to ideal functionality $\mathcal{F}_{\mathsf{mNIOT}}$ in the AGM.

Theorem 4.3. *Let the CDH assumption hold over group $\mathcal{G}$. Then the* NIOT *in Figure 9 instantiated with the CDH* KEM *in Section 2.2 is UC secure with respect to $\mathcal{F}_{\mathsf{mNIOT}}$ in the programmable random oracle model combined with the algebraic group model.*

We refer for the proof to Appendix D. We show the same statement for the OT_2 construction in Theorem 4.4 and refer for the proof to Appendix E.

Theorem 4.4. *Let the CDH assumption hold over group $\mathcal{G}$. Then the* NIOT *in Figure 10 instantiated with the CDH* KEM *in Section 2.2 is UC secure with respect to $\mathcal{F}_{\mathsf{mNIOT}}$ in the programmable random oracle model combined with the algebraic group model.*

Since both of the protocol use the same public key in the CDH KEM setting, a party can use the same public key to run both protocols and the security follows from the JUC composability. There is one caveat, both protocols are proven in the local random oracle model and local AGM. This technically requires that different random oracles or groups are used. When this is not the case, there could be issues when simulating the random oracle or group. Since in the AGM setting, we do not simulate the group, this should not pose an issue and reusing a group seems less of a concern than reusing the same hash function.

Operation	cycles	running time
Mul	584561	0.277 ms
Sub	1228	0.001 ms
Add	1118	0.001 ms
GHash	245569	0.116 ms
Key Generation	232424	0.110 ms
$OT_{1,R}$	1378214	0.653 ms
$OT_{1,S}$	1867754	0.885 ms
$OT_{2,R}$	1195311	0.566 ms
$OT_{2,S}$	1879205	0.890 ms

Figure 12: We benchmark elliptic curve multiplications (Mul), subtractions (Sub), additions (Add) and hash operations into the group (GH) for curve25519 and average over 25k runs. We use $OT_{1,R}$, $OT_{2,R}$, $OT_{1,S}$, $OT_{2,S}$ to denote the protocol execution of R, S respectively. In $OT_{1,S}$, $OT_{2,S}$, we do not include the key generation. Further, we consider the protocol in the unauthenticated setting.

Operation	cycles	running time
Key Generation	277570659	131.415 ms
RSA Enc	142624	0.068 ms
RSA Dec	3820942	1.810 ms
$OT_{1,R}$	186457	0.089 ms
$OT_{1,S}$	7654668	3.625 ms
$OT_{2,R}$	179813	0.085 ms
$OT_{2,S}$	7640627	3.618 ms

Figure 13: We benchmark our protocol instantiated with the RSA trapdoor permutation with a key length of 2048 by averaging over 100 runs. We use $OT_{1,R}$, $OT_{2,R}$, $OT_{1,S}$, $OT_{2,S}$ to denote the protocol execution of R, S respectively. In $OT_{1,S}$, $OT_{2,S}$, we do not include the key generation. Further, we consider the protocol in the unauthenticated setting.

5 IMPLEMENTATION

We provide an implementation of the OT_1 (based on [MR19]) and OT_2 (based on [MRR20]) protocol when instantiated with the CDH KEM and RSA KEM. We report the cycle and running time amount in a modular fashion such that it is easy to compare our results with previous works based on elliptic curves. We only compare our results with OT protocols in which the first message could be seen as a public key and therefore potentially reused. These two protocols are those of Chou and Orlandi [CO15] and, Canetti, Sarkar and Wang [CSW20]. In Figure 1 one can observe the basic operations required for the three protocols. Using the running time reported in Figure 12, one can roughly estimate the performance and compare their efficiency. We emphasize that these performance estimates are only rough estimates based on a non-optimized implementation.

Our Protocol Implementation. We use the relic toolkit [AGM+] for the basic cryptographic functionalities which includes all elliptic curve operations and RSA key generation, encryption and decryption. To implement our elliptic curve based OT protocol, the operations of multiplication, addition, subtraction, hashing to a curve over an elliptic curve are sufficient. We chose the relic toolkit [AGM+] since it provides all operations over a large set of elliptic curves. Additionally, in provides the hashing to a curve point operation which is a less standard.

We implement our protocol in a straightforward fashion without optimization. We benchmark our performance on an Intel® Core™ i5-10210U CPU @ 1.60GHz × 8 with 15GB memory. In Figure 12, we report the running time and cycles of our protocol and the basic group operations such as exponentiation, division and hashing into an elliptic curve. For the latter operation, we use the operation provided in [AGM+]. For our implementation, we choose curve25519 which has been previously used in the implementations by [CO15].

We do not benchmark our results over a network. Due to the non-interactive nature of our protocol, we estimate that the impact of network delays is less significant and should not vary from previous works such as [CO15] or [CSW20]. We also do not include the authentication mechanism in the benchmark in Figure 12. The authenication mechanism is an orthogonal property that each of the OT protocols can use in a similar fashion.

We use the same hardware settings to benchmark our RSA-based OT. We report the running time and cycle amounts in Figure 13.

Multiplications with the base generator of an elliptic curve group are significantly more efficient that with a random element. This leads to a major discrepancy between the cycle amounts of a multiplication and the key generation reported in Figure 12. This fact benefits in particular the OT construction of Chou and Orlandi and using Figure 1 to estimate their performance leads to an overestimate in terms of cycles and running time. A related hidden cost the OT_1 protocol comes from the requirement to sample a random group element. In Relic this is implemented in a similar fashion as for key generation and therefore causes the same computational complexity. The OT_2 protocol does not generate such a random element which not only decreases the communication complexity but also the running time by roughly 180K cycles.

In the OT_1 protocol, the receiver has to encapsulate a key, sample a random group element, hash to a group and subtract it from the encapsulation. Since the encapsulate a key procedure requires two multiplication of which one is with a random element, we need one multiplication with a random element, two multiplications with the base generator and one hash operation in total. Adding the cycles for these individual operations based on Figure 12 results in roughly 1295K cycles which matches the receivers total running time of 1378K cycles almost, with an overhead of 83K cycles. In OT_2, there are almost the same operations minus generating one random element. Therefore the individual operations sum up to roughly 1146K cycles with a discrepancy of around 50K cycles compared to the cycles reported in Figure 12.

On the sender's side, a sender has to compute two hash to a group operations and two decapsulations. Each decapsulation requires one multiplication with a random group element. This results in two multiplications with a random element and two hash operations which based on Figure 12 totals in 1660K cycles. This has a discrepancy of 207K cycles compared to the sender's actual running time and is roughly twice the discrepancy of 83K cycles on the receiver's side. This can be explained by the fact that the sender has to do roughly twice the amount of group additions (subtractions) and standard hash operations compared to the receiver. This leads

to the conclusion that the estimate of the overall running time of sender and receiver based on the individual group operations provides a realistic estimate when distinguishing different forms of multiplications and sampling random elements.

Comparison with Chou Orlandi [CO15]. The protocol of Chou and Orlandi uses the identical key-generation procedure. On the receiver's side, it requires operations that are identical to encapsulation in the CDH KEM, which totals 816K cycles based on Figure 12. On the sender's side, the protocol can be optimized such that he only needs to compute a single multiplication with a random element which results in around 585K cycles at the cost of requiring a precomputation of another multiplication with a random element. This would increase the computational cost of the key generation to around 816K cycles. This optimization makes sense when the first OT message (the public key) is reused.

Based on these estimates, which rely on the curve25519 implementation of the relic toolkit, the OT protocol of Chou and Orlandi is significantly faster and in case of OT_1 additionally requires less communication. However, it has the drawback that it does not accomplish UC security even when the public key is only used once. In applications where a weaker security notion is sufficient and where the performance is very significant, the Chou and Orlandi OT seems to be the better choice. When UC security is required, which is the de facto standard for MPC, the Chou and Orlandi OT should not be used and the alternative to OT_1 (based on [MR19]) and OT_2 (based on [MRR20]) is the OT protocol of Canetti, Sarkar and Wang [CSW20] which is tailored to the CDH setting.

Comparison with CSW [CSW20]. The CSW OT protocol achieves UC security when the first message (public key) is only used once. Their optimized variant requires a CRS, in addition to a random oracle. In their protocol, the receiver needs to compute two hash into the group operations, one multiplication with the base generator and two with a random element. Based on Figure 12, this leads to 1892K cycles which is a 46% overhead compared to the estimate of the receiver of OT_1 and 65% in case of OT_2. The sender's key generation requires a multiplication with the base generator and one with a random element. This results in 816K cycles and an overhead of 251% compared to the estimates for the OT_1, OT_2 key generation. An optimization of the protocol requires another precomputation by the sender of two multiplications with random elements which results in a total of 1984K cycles of the sender's precomputation including key generation. This would result in an overhead of 755% for the precomputation which is typically not an issue when the first message is reused since it will be amortized over the amount of OT sessions. During the protocol, the sender needs to only compute two multiplications with random elements which results in around 1168K cycles which is only 70% of the estimate for what the OT_1 and OT_2 sender needs to compute during the protocol.

It is not straightforward to compare the CDH-based OT_1 and OT_2 with the CSW OT [CSW20]. The public key size is identical and OT2 has roughly half the communication costs on the side of the receiver. The main difference seems to be that the CSW OT have a faster sender and a slower receiver at the cost of requiring a CRS and slightly more storage to store the precomputed elements. It seems that the best choice between CSW and MRR depends on the application. When the first message is reused, OT_1 and OT_2 are secure under discrete log in the AGM while the security of CSW remains unclear in this setting.

Summary. We are unaware of any efficient OT-protocol based on RSA. The RSA-based OT_1, OT_2 are nice alternatives to a CDH-based OT in settings where RSA is preferred over elliptic-curve cryptography. As expected for RSA-based cryptosystems, the key generation and decryption operations are expensive. The sender needs to perform two decryption operations which we implement using the faster option available in the relic toolkit exploiting the CRT representation. Nevertheless, two decryptions are still very expensive which causes the sender's higher computational costs during the protocol phase. Since the receiver only needs to perform an encryption which is very fast, the receiver computational costs are very low which could be interesting in a setting where the receiver might be a resource constraint device and the sender is a server where computational costs are less an issue. The overhead of OT_1, OT_2 on top of the RSA key generation, encryption and decryption is minimal since we can use standard hashing opposed to hashing to a curve as in the CDH case. Sampling a random element is straightforward and very efficient. This makes it very easy to use wherever RSA is used.

In summary, OT_1, OT_2 seem competitive with the state of art in terms of efficiency while they improve the state of art in terms of security when the first message (public key) is reused. The estimates are dependent on the implementation of the RSA and elliptic curve operations provided by the relic toolkit as well as the specific curve choice. These estimate might significantly vary for different parameter choices and optimized implementations.

REFERENCES

[AGM+] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 673–701. Springer, Heidelberg, April 2015.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *30th ACM STOC*, pages 419–428. ACM Press, May 1998.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018.

[BM90] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557. Springer, Heidelberg, August 1990.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, Heidelberg, April / May 2002.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58. Springer, Heidelberg, August 2015.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, August 2003.

[CSV16] Ran Canetti, Daniel Shahaf, and Margarita Vald. Universally composable authentication and key-exchange with global PKI. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 265–296. Springer, Heidelberg, March 2016.

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 277–308. Springer, Heidelberg, December 2020.

[CvT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 110–123. Springer, Heidelberg, August 1995.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 205–210. Plenum Press, New York, USA, 1982.

[FGSW16] Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *2016 IEEE Symposium on Security and Privacy*, pages 452–469. IEEE Computer Society Press, May 2016.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[GIR17] Ziya Alper Genç, Vincenzo Iovino, and Alfredo Rial. "The simplest protocol for oblivious transfer" revisited. Cryptology ePrint Archive, Report 2017/370, 2017. https://eprint.iacr.org/2017/370.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018.

[HL17] Eduard Hauck and Julian Loss. Efficient and universally composable protocols for oblivious transfer from the CDH assumption. Cryptology ePrint Archive, Report 2017/1011, 2017. https://eprint.iacr.org/2017/1011.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.

[MR19] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 309–326. ACM Press, November 2019.

[MRR20] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Minimal symmetric PAKE and 1-out-of-N OT from programmable-once public functions. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 425–442. ACM Press, November 2020.

[MRR21] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Batching base oblivious transfers. *IACR Cryptol. ePrint Arch.*, 2021:682, 2021.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 381–396. Springer, Heidelberg, February 2017.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Harvard University, 1981.

[RD08] Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A ADDITIONAL DEFINITIONS

A.1 Message Authentication Code

Definition A.1 (Message Authentication Code (MAC)). *A* Message Authentication Code *consists of three ppt algorithms (*KGen, Tag, Verify*) and a key space* K *with the following syntax:*

KGen: *The key generation algorithm takes as input* 1^κ *and outputs a key* k.

Tag: *The tagging algorithm takes as inputs a key* k *and a message* m *and outputs a tag* τ.

Verify: *The verification takes as inputs a key* k*, a message* m*, a tag* τ *and outputs either* 0 *or* 1.

Further, we require correctness and existential unforgeability under chosen message and verification attacks (EU-CMVA):

Correctness:

$$\Pr[\mathsf{Verify}(\mathsf{k}, \mathsf{m}, \mathsf{Tag}(\mathsf{k}, \mathsf{m})) = 1] \geq 1 - \mathsf{negl},$$

where $\mathsf{k} \leftarrow \mathsf{KGen}(1^\kappa)$.

EU-CMVA **Security:** *For any ppt adversary* $\mathcal{A}$ *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathsf{Verify}(\mathsf{k}, \mathsf{m}^*, \tau^*) = 1 \mid \mathcal{A}(z, 1^\kappa) = (\mathsf{m}^*, \tau^*)] \leq \mathsf{negl},$$

where $\mathsf{k} \leftarrow \mathsf{KGen}(1^\kappa)$ *and* $\mathcal{A}$ *can make any polynomial amount of queries to* $\mathsf{Tag}(\mathsf{k}, \cdot)$ *and* $\mathsf{Verify}(\mathsf{k}, \cdot)$ *but* τ^* *must not be a response to a query to* $\mathsf{Tag}(\mathsf{k}, \cdot)$.

A.2 Signature Scheme

Definition A.2 (Signature Scheme (SIG)). *A* Signature Scheme *consists of three ppt algorithms (*KGen, Sign, Verify*) with the following syntax:*

KGen: *The key generation algorithm takes as input* 1^κ *and outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$.

Sign: *The signing algorithm takes as inputs* sk *and a message* m *and outputs a signature* σ.

Verify: *The verification takes as inputs* pk*, a message* m*, a signature* σ *and outputs either* 0 *or* 1.

Further, we require correctness and existential unforgeability under chosen message attacks (EU-CMVA):

Correctness:

$$\Pr[\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \mathsf{Sign}(\mathsf{sk}, \mathsf{m})) = 1] \geq 1 - \mathsf{negl},$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$.

EU-CMA **Security:** *For any ppt adversary* $\mathcal{A}$ *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathsf{Verify}(\mathsf{pk}, \mathsf{m}^*, \sigma^*) = 1 \mid \mathcal{A}(z, 1^\kappa) = (\mathsf{m}^*, \sigma^*)] \leq \mathsf{negl},$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$ *and* $\mathcal{A}$ *can make any polynomial amount of queries to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *but* σ^* *must not be a response to a query to* $\mathsf{Sign}(\mathsf{sk}, \mathsf{m}^*)$.

A.3 Algebraic Group Model

The algebraic group model (AGM) has been introduced by [FKL18] and can be seen as a trade-off between the standard model and generic group model. Security proofs in the AGM typically still require assumptions.

In the AGM, adversaries are considered algebraic. This means that if an adversary ($\mathcal{A}$) has access to group elements $[\![a_1]\!], \ldots, [\![a_n]\!]$, any group element that he outputs will be of the form $\sum_{i=1}^{n} \alpha_i [\![a_i]\!]$ and $\alpha_1, \ldots, \alpha_n$ can be efficiently extracted from $\mathcal{A}$.

In our case, we also give $\mathcal{A}$ access to a random oracle to the group. Outputs of this oracle will be contained in the set of group elements that $\mathcal{A}$ can access. This does not create any complications because the random oracle can for each query sample $u \leftarrow \mathbb{Z}_p \setminus \{0\}$ and output $[\![u]\!]$. The party in control of the oracle can then extract u from the oracle.

Similar to a random oracle, there could be session specific (local) groups or there could be a global group, which requires more care when combining the AGM with the UC model. We are unaware, of any works that combine the AGM with the UC model. Therefore and also for the sake of simplicity, we consider the former case. Nevertheless, this comes at the drawback that it does not accurately model reality where usually the same group is used across different sessions.

A.4 Cryptographic Assumptions

We recap standard assumptions such as discrete log, CDH, LWE and RSA.

Definition A.3 (Discrete Logarithm). *A ppt algorithm* $\mathcal{A}$ *solves the Discrete Logarithm problem for a group G of order p and generator* $[\![1]\!]$ *with probability* ϵ *if*

$$\Pr[\mathcal{A}([\![1]\!], [\![a]\!]) = a] \geq \epsilon,$$

where $a \leftarrow \mathbb{Z}_p$.

Definition A.4 (Computational Diffie-Hellman (CDH)). *A ppt algorithm* $\mathcal{A}$ *solves the Computational Diffie-Hellman (CDH) problem for a group G of order p and generator* $[\![1]\!]$ *with probability* ϵ *if*

$$\Pr[\mathcal{A}([\![1]\!], [\![a]\!], [\![b]\!]) = [\![ab]\!]] \geq \epsilon,$$

where $a \leftarrow \mathbb{Z}_p$, $b \leftarrow \mathbb{Z}_p$.

Definition A.5 (Learning with Errors (LWE)). *A ppt algorithm* $\mathcal{A}$ *solves the Learning with Errors (LWE) problem for parameters* $n, m, q \in \mathbb{N}$ *and noise distribution* $\mathcal{X}$ *with probability* ϵ *if*

$$\Pr[\mathcal{A}(A, As + e) = s] \geq \epsilon,$$

where $\mathsf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $s \leftarrow Z_q^n$ *and* $e \leftarrow \mathcal{X}$.

Definition A.6 (RSA). *A ppt algorithm* $\mathcal{A}$ *solves the RSA problem for parameters* $p, q \in \mathbb{N}$ *with probability* ϵ *if*

$$\Pr[\mathcal{A}(pq, e, M^e \bmod pq) = M] \geq \epsilon,$$

where $e \in \mathbb{N}$ *is a fixed parameter.*

B PROOF OF THEOREM 4.1

By the correctness of the KEM, the protocol will produce the correct outputs when two honest parties interact.

We continue with proving that the NIOT is secure against a malicious sender.

Claim B.1. *Given an* ϵ*-dense* KEM*, then it holds that in the programmable random oracle model for any ppt adversary* $\mathcal{A}$*, there exists a ppt adversary* $\mathcal{A}'$ *such that for any ppt distinguisher* D *and any polynomial size auxiliary input z,*

$$|\Pr[\mathsf{D}(z, (\mathcal{A}, \mathsf{R})_\Pi) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}', \mathcal{F}_{\mathsf{NIOT}})) = 1]| \leq \epsilon + 2^{-\kappa+2},$$

where all algorithms receive input 1^κ *and* R *additionally receives an input b.*

Proof. To prove the claim, we use a sequence of hybrids, i.e an intermediate receiver (R_1) that interacts with $\mathcal{A}$. R_1 behaves like R with the exception that it generates two encapsulations and uses its capability to program the random oracle such that the two recombined encapsulations match the two generated encapsulations.

We define R_1 as follows. When $\mathcal{A}$ makes a random oracle query, R_1 uses the lazy sampling approach and returns a random element in the domain of the random oracle and stores it in a list. If the query has been done before, it returns the output for that query in the list instead.

During the protocol phase, R_1 takes the session public key pk and generates two encapsulations $(\mathsf{ct}_0, \mathsf{k}_0) \leftarrow \mathsf{Enc}(\mathsf{pk})$ and $(\mathsf{ct}_1, \mathsf{k}_1) \leftarrow \mathsf{Enc}(\mathsf{pk})$. It then samples $r_0, r_1 \leftarrow \mathcal{G}$. Due to the high entropy of r_0, r_1 and the uniqueness of sid, the lazily sampled random oracle is undefined for inputs r_0, r_1 except with probability $2 \cdot 2^{-|\mathcal{G}|}$ which is smaller than $2 \cdot 2^{-\kappa}$. Therefore, R_1 can program the random oracles H_0 on point sid, r_1 and H_1 on point sid, r_0 such that $\mathsf{ct}_0 = r_0 \oplus \mathsf{H}_0(\mathsf{sid}, r_1)$ and $\mathsf{ct}_1 = r_1 \oplus \mathsf{H}_1(\mathsf{sid}, r_0)$.

When $\mathcal{A}$ is able to distinguish an interaction with R from R_1, then it must distinguish ct_{1-b} (R_1) from the uniformly distributed ct_{1-b} (R). This would break the density property of the KEM.

Therefore

$$|\Pr[\mathsf{D}(z, (\mathcal{A}, \mathsf{R})) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}, \mathsf{R}_1)) = 1]| \leq \epsilon + 2^{-\kappa+1}.$$

Since R_1 is, during all sessions, independent of the choice bit b (except for its output), we can use it to construct our adversary $\mathcal{A}'$ that interacts with the ideal functionality $\mathcal{F}_{\mathsf{NIOT}}$ and returns the output of $\mathcal{A}$ (by interacting with him as well). $\mathcal{A}$' follows the description of R_1 with the following exceptions. When $\mathcal{A}$ registers pk during the registration phase, $\mathcal{A}$' sends sid to $\mathcal{F}_{\mathsf{NIOT}}$. During the protocol phase, $\mathcal{A}$' receives message $(\mathsf{sid}, b, \mathsf{k}_b)$ from $\mathcal{F}_{\mathsf{NIOT}}$. $\mathcal{A}$' generates $\mathsf{ct}_0, \mathsf{ct}_1$ as R_1 does. The encapsulated keys $\mathsf{k}'_0, \mathsf{k}'_1$ of $\mathsf{ct}_0, \mathsf{ct}_1$ have high entropy and therefore, because sid is unique, the lazily sampled H_2 is undefined on inputs $\mathsf{k}'_0, \mathsf{k}'_1$ except with probability less than $2 \cdot 2^{-\kappa}$. Thus, $\mathcal{A}$' can program H_2 such that $\mathsf{H}_2(\mathsf{sid}, \mathsf{k}'_0) = \mathsf{k}_0$ and $\mathsf{H}_2(\mathsf{sid}, \mathsf{k}'_1) = \mathsf{k}_1$. $\mathcal{A}$' sends r_0, r_1 and then outputs the output of $\mathcal{A}$.

This implies

$$|\Pr[\mathsf{D}(z, (\mathcal{A}, \mathsf{R}_1)) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}', \mathcal{F}_{\mathsf{NIOT}}) = 1]| \leq 2^{-\kappa+1}.$$

□

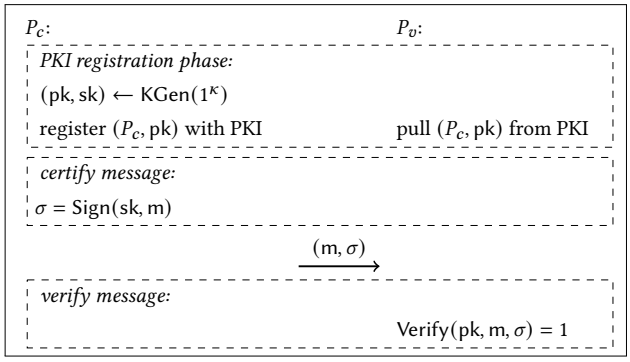

Figure 14: The figure shows a certified message functionality based on a signature scheme and PKI.

We finish the proof of the theorem by showing that the OT protocol is secure against a malicious receiver.

Claim B.2. *Given an ϵ_1-dense* KEM *that is ϵ_2* OW-CPA *secure. Then it holds that in the programmable random oracle model for any ppt adversary $\mathcal{A}$, there exists a ppt adversary $\mathcal{A}'$ such that for any ppt distinguisher* D *and any polynomial size auxiliary input z,*

$$|\Pr[\mathsf{D}(z, (\mathsf{S}, \mathcal{A})_{\Pi}) = 1] - \Pr[\mathsf{D}(z, (\mathcal{F}_{\mathsf{NIOT}}, \mathcal{A}')) = 1]| \leq q_h^2 \epsilon_1 + q_h^5 \epsilon_2,$$

where all algorithms receive input 1^κ and q_h is an upper bound on the amount of random oracle queries of $\mathcal{A}$. [1]

Proof. We use again a sequence of hybrids. This time, the hybrids iterate over all random oracle queries to $\mathsf{H}_0, \mathsf{H}_1$ for the same session sid. We consider a collection of senders $(\mathsf{S}_i)_{i \in [q_h]}$, where S_0 is identical to S.

We define the hybrids in the following way. S_i is identical to S_{i-1} except how he computes a key k given a k′ from a specific set K′ which we will define as follows[2]. Instead of querying the random oracle H_2 on input k′ to compute k, he samples k uniformly at random. Let r be the i-th query to random oracles $\mathsf{H}_0(\mathsf{sid}, \cdot), \mathsf{H}_1(\mathsf{sid}, \cdot)$ and let that oracle be H_d for $d \in \{0, 1\}$. We define the set

$$\mathsf{K}' := \{\mathsf{k}' \mid \mathsf{k}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}), \mathsf{ct} = r \oplus \mathsf{H}_{1-d}(\mathsf{sid}, r_n), i < n \leq q_h\},$$

where r_n is defined as the n-th query to oracle $\mathsf{H}_{1-d}(\mathsf{sid}, \cdot)$.

Now, we show that $\mathcal{A}$ cannot distinguish S_i from S_{i-1} using a hybrid S'_{i-1}. S'_{i-1} is identical to S_{i-1} except that he programs the random oracle $\mathsf{H}_{1-d}(\mathsf{sid}, r_n)$ for all $i < n \leq q_h$ such that ct = $r \oplus H(\mathsf{sid}, r_n)$ is an output of $\mathsf{Enc}(\mathsf{pk})$. He can do this by sampling $(\mathsf{ct}, \mathsf{k}') \leftarrow \mathsf{Enc}(\mathsf{pk})$ and programming $\mathsf{H}_{1-d}(\mathsf{sid}, r_n) = r \ominus \mathsf{ct}$. For every n, he samples a fresh ct. S'_{i-1} programs $\mathsf{H}_{1-d}(\mathsf{sid}, \cdot)$ on r_n whenever $\mathcal{A}$ makes a query r_n to H_{1-d}. Since we assume that $\mathcal{A}$ does not query the same input to the same oracle twice[3], H_{1-d} will be undefined such that the programming does not fail.

If $\mathcal{A}$ can distinguish S_{i-1} and S'_{i-1}, then he can distinguish a uniform ct from an output of Enc and hence break the density property of KEM. Therefore,

$$|\Pr[\mathsf{D}(z, (\mathsf{S}_{i-1}, \mathcal{A})) = 1] - \Pr[\mathsf{D}(z, (\mathsf{S}'_{i-1}, \mathcal{A})) = 1]| \leq q_h \epsilon_1.$$

Notice that for all programmed encapsulations, their encapsulated key is in the set K′. Further, S_i is identical to S'_{i-1} except that for all encapsulated keys k′ of the encapsulations ct that were used to program H_{1-d}, i.e. k′ ∈ K′, S_i defines k as uniform instead of $\mathsf{k} = \mathsf{H}_2(\mathsf{sid}, \mathsf{k}')$. $\mathcal{A}$ can only distinguish S_i from S'_{i-1} if he makes a query a key from K′ to $\mathsf{H}_2(\mathsf{sid}, \cdot)$. Though if that is the case, we can build an adversary $\mathcal{A}_{\mathsf{KEM}}$ that breaks the key recovery security of KEM in the following way. $\mathcal{A}_{\mathsf{KEM}}$ registers the challenge public key and guesses a random query j in $(i, q_h]$ of $\mathcal{A}$ to $\mathsf{H}_{1-d}(\mathsf{sid}, \cdot)$ and programs it to the challenge ciphertext. $\mathcal{A}_{\mathsf{KEM}}$ fails if (r_0, r_1) of the protocol do not correspond to queries (r_i, r_j). Assuming that this is not the case, $\mathcal{A}_{\mathsf{KEM}}$ needs to generate the senders output. He rolls a three sided dice. If one, he guesses that $\mathcal{A}$ has made its query for k_0 to $\mathsf{H}_2(\mathsf{sid}, \cdot)$, if two or three he guesses not. In the first case $\mathcal{A}_{\mathsf{KEM}}$ picks his k_0 randomly from the outputs of $\mathsf{H}_2(\mathsf{sid}, \cdot)$. In the second case, he chooses a uniform k and he picks a random future query to $\mathsf{H}_2(\mathsf{sid}, \cdot)$ that he will program to k. The third case is identical to the second case, except that he does not program $\mathsf{H}_2(\mathsf{sid}, \cdot)$ for any future query. He follows the same strategy for computing k_1. $\mathcal{A}_{\mathsf{KEM}}$ returns protocol output $(\mathsf{k}_0, \mathsf{k}_1)$. After all queries to $\mathsf{H}_2(\mathsf{sid}, \cdot)$ have been finished, $\mathcal{A}_{\mathsf{KEM}}$ returns the preimage of k_{1-d} to the challenger.

We can now bound the probability that S_i and S'_{i-1} can be distinguished as follows.

$$|\Pr[\mathsf{D}(z, (\mathsf{S}_i, \mathcal{A})) = 1] - \Pr[\mathsf{D}(z, (\mathsf{S}'_{i-1}, \mathcal{A})) = 1]| \leq 9 q_h^4 \epsilon_2.$$

For any choice of (r_0, r_1) of $\mathcal{A}$ that have been queried to $\mathsf{H}_0, \mathsf{H}_1$, at least one of the keys in the output of S_{q_h} does not match the actual key. Let this key be k_{1-b} which is unnoticed by $\mathcal{A}$ or D. Therefore, we can use S_{q_h} to construct $\mathcal{A}'$. $\mathcal{A}'$ needs to send $(\mathsf{sid}, b, \mathsf{k}_b)$ to $\mathcal{F}_{\mathsf{NIOT}}$. He follows the description of S_{q_h}. Instead of outputting $(\mathsf{k}_b, \mathsf{k}_{1-b})$ where k_{1-b} is fake, it sends $(\mathsf{sid}, b, \mathsf{k}_b)$ to $\mathcal{F}_{\mathsf{NIOT}}$ and outputs the output of $\mathcal{A}$. The output of S_{q_h} is identical to the output of $\mathcal{F}_{\mathsf{NIOT}}$. Thus,

$$\Pr[\mathsf{D}(z, (\mathsf{S}_{q_h}, \mathcal{A})) = 1] = \Pr[\mathsf{D}(z, (\mathsf{NIOT}, \mathcal{A}')) = 1].$$

□

C SECURE CERTIFICATION DIRECTLY BASED ON PKI

In the first part of this section, we show how to realize $\overline{\mathcal{F}}_{\mathsf{cert}}$ based on signature schemes. During the second part, we show how to obtain secure session certification without relying on $\overline{\mathcal{F}}_{\mathsf{cert}}$. We first show it based on a CCA secure KEM. Second, we also show it based on the CDH KEM in the AGM.

C.1 Signature based Message Certification

In Figure 14 shows a realization of $\overline{\mathcal{F}}_{\mathsf{cert}}$ based on a signature scheme. The construction is basically identical to the construction of Canetti et al. [CSV16] with the difference that we do not sign the session or message identity, which results in a weaker ideal functionality that allows replay attacks.

[1] We did not optimize the tightness of our analysis. The loss of the reduction can be reduced e.g. by using an IND-CPA KEM.

[2] Additionally S_i differs from S_{i-1} in the fact that some outputs of H_{1-d} are distributed the same as encapsulations generated by $\mathsf{Enc}(\mathsf{pk})$.

[3] We make this assumption only for simplicity. If $\mathcal{A}$ has already queried r_n, i.e. $r_n = r_j$ for $j < n$, then if $i < j$, it has been already successfully programmed or if $i > j$, it does not need to be programmed.

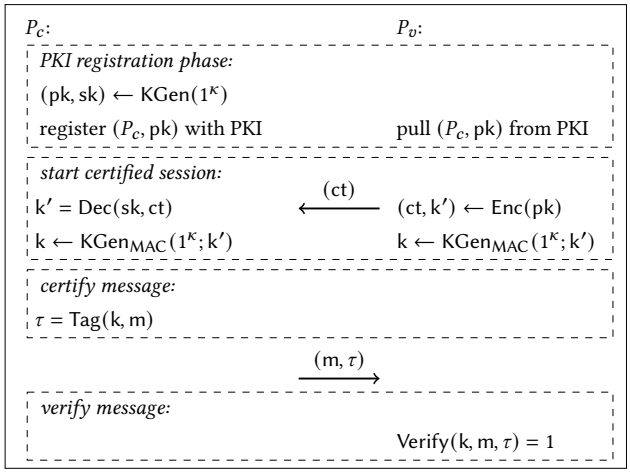

Figure 15: The figure shows a certified session based on a CCA KEM and MAC solely based on PKI.

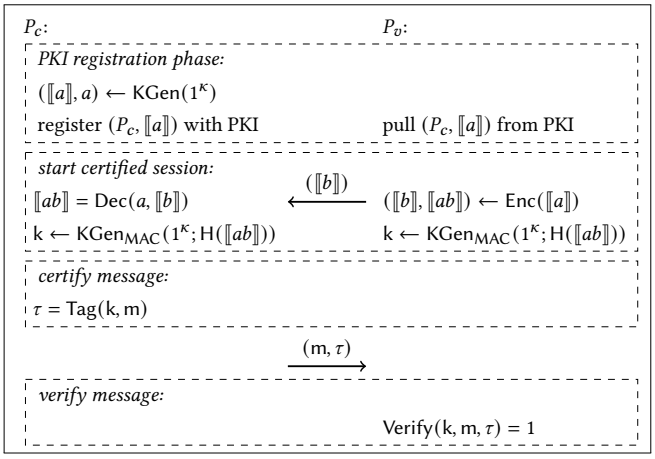

Figure 16: The figure shows a certified session based on a CCA PKE, a hash function H and a MAC solely based on PKI.

Theorem C.1. *Let* Π *be the protocol in Figure 14 and* SIG EU-CMA *secure and correct. Then, for any ppt adversary* $\mathcal{A}$ *in the UM model, there exists a ppt adversary* $\mathcal{A}'$ *in the UM model such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{UM}})_\Pi) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})^{\overline{\mathcal{F}}_{\mathsf{cert}}}) = 1]| = \mathsf{negl},$$

where all algorithms receive input 1^κ.

Proof. Given $\mathcal{A}$, we construct $\mathcal{A}'$. First notice that environment D can only distinguish $\mathcal{A}$ from $\mathcal{A}'$ in two ways. First, when $\mathcal{A}$ creates a valid pair (m, σ) which has not been recorded as $(v, \mathsf{m}, \sigma, 1)$ by $\overline{\mathcal{F}}_{\mathsf{cert}}$. Second, when the correctness of SIG fails.

In case $\mathcal{A} = P_c$, constructing $\mathcal{A}'$ is easy. We let $\mathcal{A}'$ interact with $\mathcal{A}$. $\mathcal{A}'$ knows sk, can verify (m, σ) and force $\overline{\mathcal{F}}_{\mathsf{cert}}$ to record $(\mathsf{pk}, \mathsf{m}, \sigma, 1)$ when (m, σ) is valid. This only fails when the correctness fails.

The non-trivial case is when $\mathcal{A} = P_v$. In this case, $\mathcal{A}$ has no control over pk since it is authenticated using the PKI. We can use the security of the SIG scheme as follows. In the EU-CMA game, we can request arbitrary signatures and win if we can forge a new signature. Our adversary $\mathcal{A}'$ interacts with $\mathcal{A}$ as follows. To certify message, he uses the signature queries in the EU-CMA game to simulate P_c. More specifically, whenever P_c requests a certificate for m from $\overline{\mathcal{F}}_{\mathsf{cert}}$, he makes a signature query for m and sends σ to $\overline{\mathcal{F}}_{\mathsf{cert}}$. Further, he uses (m, σ) as message for $\mathcal{A}$. Whenever P_v makes a query (m, σ). $\mathcal{A}'$ checks whether σ has been a response to a signature query. If not, he makes a verification query. If the verification fails, $\overline{\mathcal{F}}_{\mathsf{cert}}$ answers also with 0 since this query has not been recorded. If the verification succeeds, D might distinguish $\mathcal{A}'$ from $\mathcal{A}$ but $\mathcal{A}'$ breaks the EU-CMA security.

Let SIG be ϵ EU-CMA secure. Then, conditioned on $\mathcal{A} \neq P_c$,

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{UM}})_\Pi) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})^{\overline{\mathcal{F}}_{\mathsf{cert}}}) = 1]| \leq \epsilon.$$

Let SIG be δ correct. Then

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{UM}})_\Pi) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})^{\overline{\mathcal{F}}_{\mathsf{cert}}}) = 1]| \leq \epsilon + \delta.$$

□

C.2 CCA KEM based Session Certification

In Figure 15, we show a secure session certification solely based on PKI, a CCA secure KEM and a CMVA secure MAC without relying on any other certification mechanisms. During the protocol of Figure 6, we also used $\overline{\mathcal{F}}_{\mathsf{cert}}$ to certify the KEM public key in order to prove security. This is not the case in the protocol in Figure 15.

Theorem C.2. *Let* Π *be the protocol in Figure 15,* KEM *be* IND-CCA *secure and correct and* MAC EU-CMVA *secure and correct. Then, for any ppt adversary* $\mathcal{A}$ *in the UM model, there exists a ppt adversary* $\mathcal{A}'$ *in the UM model such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\mathsf{D}(z, (\mathcal{A}_{\mathsf{UM}})_\Pi) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}'_{\mathsf{UM}})^{\overline{\mathcal{F}}_{\mathsf{cs}}}) = 1]| = \mathsf{negl},$$

where all algorithms receive input 1^κ.

Proof. The proof follows the framework of the proof of Theorem 3.3. The only difference is that we use the same public key across different sessions. Notice that in the non-trivial case, I.e. $\mathcal{A} \notin (P_c, P_v)$, the adversary cannot manipulate the public key pk which is directly obtained from the PKI.

During the hybrids, there is a step where we switch the encapsulation to an encapsulation that is uncorrelated to the MAC key. During this step, we will lose access to the secret key. Since our KEM is CCA secure, we will use the decapsulation oracle to decapsulate incoming encapsulations of other sessions. If the encapsulation of another session matches our challenge encapsulation, they will end up being uncorrelated to the MAC key as well while due to correctness, the session must use the same MAC k. Therefore, we can handle such sessions identical to the challenge session.

Once we have switched the encapsulation to be uncorrelated, the rest of the proof is identical to last parts of the proof of Theorem 3.3 in which we exploit the security of the MAC. □

C.3 KEM based Session Certification in AGM

In Figure16, we show a session certification protocol that is based on the CDH KEM and secure in the AGM based on discrete log. To prove this theorem, we use a similar approach as for proving Theorem 4.3. This allows us to use the same KEM for the OT procedure and to encapsulate the MAC key.

Theorem C.3. *Let* Π *be the protocol in Figure 16,* MAC EU-CMVA *secure and correct and the discrete log assumption hold over group* $\mathcal{G}$*. Then, in the random oracle model combined with the AGM model, for any ppt adversary* $\mathcal{A}$ *in the UM model, there exists a ppt adversary* $\mathcal{A}'$ *in the UM model such that for any ppt environment* D *and any polynomial size auxiliary input* z

$$|\Pr[\mathsf{D}(z,(\mathcal{A}_{\mathsf{UM}})_{\Pi})=1]-\Pr[\mathsf{D}(z,(\mathcal{A}'_{\mathsf{UM}})_{\overline{\mathcal{F}_{\mathsf{cs}}}})=1]|=\mathsf{negl},$$

where all algorithms receive input 1^κ*.*

Proof. As in Theorem C.2, the main challenge is how to replace encapsulations that are related to the MAC key with unrelated encapsulations. We use a similar strategy as in Theorem 4.3 for the hybrid of replacing the ciphertexts. Let $\mathcal{A}'_1$ be a hybrid which follows the protocol except that the encapsulation is independent of the MAC key. More specifically, $\mathcal{A}'_1$ will use the extract mechanism of the AGM to try to extract the randomness $b \in \mathbb{Z}_p$ used for the encapsulation. The extract algorithm will return α, β with $\mathsf{ct} = \alpha[\![a]\!] + \beta\mathsf{pk}$. Whenever $\beta \neq 0$, it just uses a random key rather than a random oracle output as MAC key. When $\beta = 0$, it follows the normal protocol by setting $\mathsf{k} = \mathsf{H}(\alpha\mathsf{pk})$.

Our reduction between $\mathcal{A}$ and $\mathcal{A}'_1$ plays the discrete log game in the AGM as follows. First it receives a discrete log instance $[\![a]\!]$. In each session where it needs to generate an encapsulation, he samples a random $\gamma \leftarrow \mathbb{Z}_p$ and sends encapsulation $\gamma[\![a]\!]$ and samples an independent MAC key. Whenever the adversary makes a random oracle query q to H, the reduction checks for any γ whether $\frac{q}{\gamma}[\![1]\!] = [\![a]\!]$, if so it breaks discrete log by returning $\frac{q}{\gamma} = a$. When the adversary sends an encapsulation for which the reduction extracts α, β with $\beta = 0$, the reduction uses $\mathsf{H}(\alpha[\![a]\!])$ to compute the MAC key. When the adversary sends an encapsulation where $\beta \neq 0$, the reduction checks for all previous and all future random oracle the following. Let the query be q and the extraction returns α_q, β_q with $q = \alpha_q[\![1]\!] + \beta_q[\![a]\!]$. It then solves the equation for $\hat{a}$ (using the Tonelli-Shanks algorithm)

$$\beta\hat{a}^2 + (\alpha - \beta_q)\hat{a} - \alpha_q = 0. \tag{1}$$

If it has a non-trivial solution, check for any solution whether $\hat{a}[\![1]\!] = [\![a]\!]$. If so break the discrete log by returning $\hat{a}$.

Due to the perfect correctness of the CDH based KEM, the environment D can only distinguish $\mathcal{A}$ from $\mathcal{A}'_1$ when $\mathcal{A}$ generates an encapsulation $[\![b]\!] = \alpha[\![1]\!] + \beta[\![a]\!]$ with $\beta \neq 0$ and makes a random oracle query for $q = \alpha_q[\![1]\!] + \beta[\![1]\!] = [\![ab]\!]$. In this case, Equation 1 is non-trivial because of $\beta \neq 0$ and has a solution $\hat{a}$ such that the reduction will break the discrete log assumption. Therefore, if discrete log is ϵ hard, then

$$|\Pr[\mathsf{D}(z,(\mathcal{A}_{\mathsf{UM}})_{\Pi})=1]-\Pr[\mathsf{D}(z,(\mathcal{A}'_{1,\mathsf{UM}})_{\Pi})=1]| \leq \epsilon.$$

For the remaining part of the proof, we following the proof of Theorem 3.3 using the security of the MAC for any session for which the MAC key is independent of the encapsulation. In the remaining sessions it is easy for $\mathcal{A}$' to recover the MAC key using the extraction queries and is therefore similar to the trivial case. □

D PROOF OF THEOREM 4.3

The CDH KEM of Section 2.2 has statistical density and perfect correctness. Therefore it is easier to prove security against a malicious sender than in Theorem 4.1.

Claim D.1. *In the programmable random oracle model for any ppt adversary* $\mathcal{A}$*, there exists a ppt adversary* $\mathcal{A}'$ *such that for any ppt distinguisher* D *and any polynomial size auxiliary input* z*,*

$$|\Pr[\mathsf{D}(z,(\mathcal{A},\mathsf{R})_{\Pi})=1]-\Pr[\mathsf{D}(z,(\mathcal{A}',\mathcal{F}_{\mathsf{mNIOT}}))=1]| \leq q_{\mathsf{ssid}}2^{-\kappa+2},$$

where all algorithms receive input 1^κ*,* q_{ssid} *is the amount of protocol subsessions and* R *additionally receives an input* b *for each subsession.*

Proof. We construct our adversary $\mathcal{A}'$ that interacts with the ideal functionality $\mathcal{F}_{\mathsf{mNIOT}}$ and outputs the output of $\mathcal{A}$ (by interacting with him as well) as follows. $\mathcal{A}$' lazily samples all random oracles and just forwards the queries and responses between the oracles and $\mathcal{A}$. $\mathcal{A}$ sends a message $(\mathsf{sid}, \mathsf{pk})$. $\mathcal{A}$' sends the session id sid to $\mathcal{F}_{\mathsf{mNIOT}}$. $\mathcal{F}_{\mathsf{mNIOT}}$ outputs one message $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}_0, \mathsf{k}_1, \mathsf{ssid}')$ for each subsession. $\mathcal{A}$' handles them as follows. If $\mathsf{ssid}' = \emptyset$, he samples two encapsulations together with their encapsulated keys, i.e. $(\mathsf{ct}_0, \mathsf{k}'_0), (\mathsf{ct}_1, \mathsf{k}'_1) \leftarrow \mathsf{Enc}(\mathsf{pk})$. He also samples uniform r_0, r_1 and programs H_0, H_1 such that $ct_0 = r_0 + \mathsf{H}_0(\mathsf{pk}, r_1)$ and $\mathsf{ct}_1 = r_1 + \mathsf{H}_1(\mathsf{pk}, r_0)$. Since r_0, r_1 have high entropy, the programming step will succeed except with at most probability $2 \cdot 2^{-\kappa}$. Because the encapsulations are uniform, this does not change the output distribution of the oracles. $\mathcal{A}$' also programs H_2 such that $k_0 = \mathsf{H}_2(r_0, r_1, \mathsf{k}'_0)$ and $k_1 = \mathsf{H}_2(r_0, r_1, \mathsf{k}'_1)$ which also succeeds except probability at most $2 \cdot 2^{-\kappa}$. Then $\mathcal{A}$' forwards (r_0, r_1) to $\mathcal{A}$. If $\mathsf{ssid}' \neq \emptyset$, $\mathcal{A}$' just resends r_0, r_1 of session ssid' to $\mathcal{A}$. After handling all subsessions, $\mathcal{A}$' outputs the output of $\mathcal{A}$.

Therefore

$$|\Pr[\mathsf{D}(z,(\mathcal{A},\mathsf{R})_{\Pi})=1]-\Pr[\mathsf{D}(z,(\mathcal{A}',\mathcal{F}_{\mathsf{mNIOT}})=1]| \leq q_{\mathsf{ssid}}2^{-\kappa+2}.$$

□

We finish the proof of the theorem by showing that the OT protocol is secure against a malicious receiver. For proving this statement, we need the power of the AGM.

Claim D.2. *Let the discrete log assumption hold over group* $\mathcal{G}$ *except advantage* ϵ*. Then, in the programmable random oracle model combined with the algebraic group model for any ppt adversary* $\mathcal{A}$*, there exists a ppt adversary* $\mathcal{A}'$ *such that for any ppt distinguisher* D *and any polynomial size auxiliary input* z*,*

$$\begin{aligned}&|\Pr[\mathsf{D}(z,(\mathsf{S},\mathcal{A})_{\Pi})=1]-\Pr[\mathsf{D}(z,(\mathcal{F}_{\mathsf{NIOT}},\mathcal{A}'))=1]|\\ &\leq \epsilon + q_{\mathsf{ssid}}2^{-\kappa+1},\end{aligned}$$

where all algorithms receive input 1^κ *and* q_h *is an upper bound on the amount of random oracle queries of* $\mathcal{A}$ *and* q_{ssid} *is a bound on the amount of protocol subsessions.*

Proof. We use again a sequence of hybrids, i.e. a collection of senders $(\mathsf{S}_i)_{i\in[2]}$. S_1 is identical to S except that he uses the extract mechanism on $\mathcal{A}$ and the random oracle (as described in Section A.3) to decapsulate ct_0 and ct_1 instead of his secret key. More concretely, S_1 generates $([\![a]\!], a) \leftarrow \mathsf{KGen}(1^\kappa)$ and sends

$[\![a]\!]$ to $\mathcal{A}$. $\mathcal{A}$ has access to group elements $[\![1]\!]$, $[\![a]\!]$ and all the group elements that are outputs of the random oracles. When $\mathcal{A}$ sends r_0, r_1, S_1 computes ct_0, ct_1 according to protocol and uses the extraction mechanism to learn $\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}$ such that for all $d \in \{0,1\}$, $\mathsf{ct}_d = \alpha_{d,0}[\![1]\!] + \alpha_{d,1}[\![a]\!]$. $\mathcal{A}$' computes $\mathsf{k}'_d = (\alpha_{d,0} + \alpha_{d,1}a)\mathsf{pk}$. He queries H_2 on k'_0 and k'_1 according to protocol and outputs $\mathsf{k}_0, \mathsf{k}_1$. Thus,

$$\Pr[\mathsf{D}(z, (\mathsf{S}, \mathcal{A})_\Pi) = 1] = \Pr[\mathsf{D}(z, (\mathsf{S}_1, \mathcal{A})) = 1].$$

S_2 is identical to S_1 except that he sometimes outputs a uniform string instead of a k with the guarantee that this is the case for at least one of the keys $\mathsf{k}_0, \mathsf{k}_1$ for each subsession. We show that based on the discrete log assumption the two sender are indistinguishable as follows. Given a discrete log instance $[\![a]\!]$, the reduction sends $[\![a]\!]$ as pk to $\mathcal{A}$. It responds to all oracle queries to $\mathsf{H}_0, \mathsf{H}_1$ with $\beta_n[\![a]\!]$ for uniform $\beta_n \in \mathbb{Z}_p \setminus \{0\}$, where n is the query number. This is statistically indistinguishable from responding with $\beta_n[\![1]\!]$ as long as $[\![a]\!]$ is a generator for $\mathcal{G}$.

Whenever $\mathcal{A}$ makes a query q to H_d for $d \in \{0,1\}$, the reduction computes all new candidates $\hat{r}_{0,i,j}, \hat{r}_{1,i,j}$ as follows. Let q be the i-th query. For every query q_j with $j < i$ to H_{1-d}, it computes candidates $\hat{r}_{d,i,j} := q, \hat{r}_{1-d,i,j} := q_j$. We assume that $\mathcal{A}$ always queries his r_0, r_1 to $\mathsf{H}_1, \mathsf{H}_0$. If not, the reduction will make the queries for him.

The reduction extracts $(\alpha_{d,\ell,i,j})_{d\in\{0,1\},i\in\{0,1\}}$ such that

$$\hat{r}_{d,i,j} = \alpha_{d,0,i,j}[\![1]\!] + \alpha_{d,1,i,j}[\![a]\!].$$

The reduction then runs a check procedure with the goal to extract a discrete log solution. Before describing this procedure, we describe how the reduction handles oracle queries to H_2.

The reduction monitors all oracle queries to H_2. For a query q_m with $m \in [q_h]$, the reduction extracts $\gamma_{0,m}, \gamma_{1,m}$ such that $q = \gamma_{0,m}[\![1]\!] + \gamma_{1,m}[\![a]\!]$. It then runs the same check procedure as mentioned earlier. We will describe this procedure later. When this check fails, it returns the output of H_2.

Now, we describe the check procedure. For every $(i, j, m) \in [q_h]^3$ and $d \in \{0,1\}$ for which the αs and γs are defined, the reduction solves the following quadratic equation for $\hat{a}_{i,j,m}$ which can be solved efficiently (using e.g. the Tonelli-Shanks algorithm)

$$(\alpha_{d,1,i,j} + \beta_{d,i,j})\hat{a}^2_{i,j,m} + (\alpha_{d,0,i,j} - \gamma_{1,m})\hat{a}_{i,j,m} - \gamma_{0,m} = 0 \mod p.$$

When it has a non-trivial solution s, the reduction checks whether $s[\![1]\!] = [\![a]\!]$ and if yes it outputs the challenge discrete log $s = a$ to the challenger.

For each subsession, there is at least one encapsulated key k′ such that if $\mathcal{A}$ queries k' to H_2, this check will successfully extract a discrete log solution for the following reason. Recap that the candidate encapsulations are defined as

$$\widehat{\mathsf{ct}}_{d,i,j} = \alpha_{d,0,i,j}[\![1]\!] + (\alpha_{d,1,i,j} + \beta_{d,i,j})[\![a]\!],$$

where $\beta_{d,i,j}[\![a]\!] = \mathsf{H}_d(\mathsf{pk}, \hat{r}_{1-d,i,j})$. Further, the corresponding candidate keys are defined as

$$\hat{\mathsf{k}}_{d,i,j} := \mathsf{H}_2(\hat{r}_{0,i,j}, (\alpha_{d,0,i,j} + a\alpha_{d,1,i,j} + a\beta_{d,i,j})[\![a]\!]).$$

Matching this input with a query to H_2 leads to the quadratic equation from above.

Now we need to argue that for each subsession at least one of the encapsulated keys k'_0, k'_1 of the actual messages r_0, r_1 if queried to H_2 would lead to extracting a. Let $r_d = q_j$ and $r_{1-d} = q_i$ such that $j > i$ for some $d \in \{0,1\}$. Then $\beta_{1-d,i,j}$ is sampled after $(\alpha_{d,c,i,j})_{d,c\in\{0,1\}}$ have been chosen by $\mathcal{A}$. Therefore $\alpha_{1-d,1,i,j} + \beta_{1-d,i,j} \neq 0$ except with a probability of at most $2^{-\kappa+1}$. This ensures that computing k'_{1-d} is non-trivial or from a different perspective, that the quadratic equation has a non-trivial solution. Notice that a solution is guaranteed since a exists and is a solution. Thus if k'_{1-d} is queried to H_2, the reduction can compute a and terminates. S_2 will output a uniform string u instead k_{1-d}. d can be reconstructed by the query order of r_0, r_1. r_d is the query that has been last.

Based on the ϵ-hardness of discrete log,

$$|\Pr[\mathsf{D}(z, (\mathsf{S}_1, \mathcal{A})) = 1] - \Pr[\mathsf{D}(z, (\mathsf{S}_2, \mathcal{A})) = 1]| \leq \epsilon + q_{\mathsf{ssid}}2^{-\kappa+1}.$$

As a last step, we use S_2 to construct $\mathcal{A}$'. $\mathcal{A}$' will use the same strategy as S_2 to reconstruct the subsession choice bits from $\mathcal{A}$. As S_2 does, $\mathcal{A}$' samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$ and sends it to $\mathcal{A}$. Upon receiving sid, ssid, $r_{\mathsf{ssid},0}, r_{\mathsf{ssid},1}$ from $\mathcal{A}$, $\mathcal{A}$' checks whether $r_{\mathsf{ssid},0}, r_{\mathsf{ssid},1}$ are identical to $r_{\widehat{\mathsf{ssid}},0}, r_{\widehat{\mathsf{ssid}},1}$ of a previous subsession. If that is the case, it looks up its message $(\mathsf{sid}, \widehat{\mathsf{ssid}}, \hat{b}, \hat{\mathsf{k}}_{\hat{b}}, \widehat{\mathsf{ssid}'})$ and sends message $(\mathsf{sid}, \mathsf{ssid}, \hat{b}, \hat{\mathsf{k}}_{\hat{b}}, \widehat{\mathsf{ssid}})$ to $\mathcal{F}_{\mathsf{mNIOT}}$. Otherwise, it follows the strategy of S_2 to extract subsession choice bit b and the corresponding key k_b. It then sends message $(\mathsf{sid}, \mathsf{ssid}, b, \mathsf{k}_b, \emptyset)$ to $\mathcal{F}_{\mathsf{mNIOT}}$.

After all subsessions are over, $\mathcal{A}$' outputs the output of $\mathcal{A}$. Further, the CDH KEM is perfectly correct. Therefore,

$$|\Pr[\mathsf{D}(z, (\mathsf{S}_2, \mathcal{A})) = 1] = \Pr[\mathsf{D}(z, (\mathcal{A}', \mathcal{F}_{\mathsf{mNIOT}})) = 1].$$

□

E PROOF OF THEOREM 4.4

The proof is very similar to the proof of Theorem 4.1. We use the same underlying CDH KEM and thus only the part how the random oracles are used differs between the two constructions. We give the full proof for the sake of completeness.

Claim E.1. *In the programmable random oracle model for any ppt adversary $\mathcal{A}$, there exists a ppt adversary $\mathcal{A}$' such that for any ppt distinguisher* D *and any polynomial size auxiliary input z,*

$$|\Pr[\mathsf{D}(z, (\mathcal{A}, \mathsf{R})_\Pi) = 1] - \Pr[\mathsf{D}(z, (\mathcal{A}', \mathcal{F}_{\mathsf{mNIOT}})) = 1]| \leq q_{\mathsf{ssid}}2^{-\kappa+2},$$

where all algorithms receive input 1^κ, q_{ssid} is the amount of protocol subsessions and R *additionally receives an input b for each subsession.*

Proof. We construct our adversary $\mathcal{A}'$ that interacts with the ideal functionality $\mathcal{F}_{\mathsf{mNIOT}}$ and outputs the output of $\mathcal{A}$ (by interacting with him as well) as follows. $\mathcal{A}$' lazily samples all random oracles and just forwards the queries and responses between the oracles and $\mathcal{A}$. $\mathcal{A}$ sends a message $(\mathsf{sid}, \mathsf{pk})$. $\mathcal{A}$' sends the session id sid to $\mathcal{F}_{\mathsf{mNIOT}}$. $\mathcal{F}_{\mathsf{mNIOT}}$ outputs one message $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}_0, \mathsf{k}_1, \mathsf{ssid}')$ for each subsession. $\mathcal{A}$' handles them as follows. If $\mathsf{ssid}' = \emptyset$, he samples two encapsulations together with their encapsulated keys, i.e. $(\mathsf{ct}_0, \mathsf{k}'_0), (\mathsf{ct}_1, \mathsf{k}'_1) \leftarrow \mathsf{Enc}(\mathsf{pk})$. He also samples uniform $s \leftarrow \{0,1\}^\kappa, T \leftarrow \mathcal{G}$ and programs H_1, such that $ct_0 = \mathsf{H}_1(\mathsf{pk}, 0, s + \mathsf{H}_0(\mathsf{pk}, 0, T)) \ominus T$ and $\mathsf{ct}_1 = \mathsf{H}_1(\mathsf{pk}, 1, s + \mathsf{H}_0(\mathsf{pk}, 1, T)) \ominus T$. Since s, T have high entropy, the programming step will succeed except with at most probability $2 \cdot 2^{-\kappa}$. Because the encapsulations are uniform, this does not change the output distribution of the oracles. $\mathcal{A}$' also programs H_2 such that $k_0 = \mathsf{H}_2(s, T, \mathsf{k}'_0)$ and $k_1 = \mathsf{H}_2(s, T, \mathsf{k}'_1)$ which also succeeds except probability at most $2 \cdot 2^{-\kappa}$. Then $\mathcal{A}$' forwards (s, T) to $\mathcal{A}$. If $\mathsf{ssid}' \neq \emptyset$, $\mathcal{A}$' just resends s, T of session

ssid′ to $\mathcal{A}$. After handling all subsessions, $\mathcal{A}$' outputs the output of $\mathcal{A}$.

Therefore

$$|\Pr[\mathsf{D}(z,(\mathcal{A},\mathsf{R})_\Pi)=1]-\Pr[\mathsf{D}(z,(\mathcal{A}',\mathcal{F}_{\mathsf{mNIOT}})=1]| \le q_{\mathsf{ssid}}2^{-\kappa+2}.$$

□

We finish the proof of the theorem by showing that the OT_2 protocol is secure against a malicious receiver. For proving this statement, we again use the power of the AGM.

Claim E.2. *Let the discrete log assumption hold over group $\mathcal{G}$ except advantage ϵ. Then, in the programmable random oracle model combined with the algebraic group model for any ppt adversary $\mathcal{A}$, there exists a ppt adversary $\mathcal{A}$' such that for any ppt distinguisher* D *and any polynomial size auxiliary input z,*

$$|\Pr[\mathsf{D}(z,(\mathsf{S},\mathcal{A})_\Pi)=1]-\Pr[\mathsf{D}(z,(\mathcal{F}_{\mathsf{NIOT}},\mathcal{A}'))=1]| \le \epsilon + q_{\mathsf{ssid}}2^{-\kappa+1},$$

where all algorithms receive input 1^κ and q_h is an upper bound on the amount of random oracle queries of $\mathcal{A}$ and q_{ssid} is a bound on the amount of protocol subsessions.

Proof. We use again a sequence of hybrids, i.e. a collection of senders $(\mathsf{S}_i)_{i\in[2]}$. S_1 is identical to S except that he uses the extract mechanism on $\mathcal{A}$ and the random oracle (as described in Section A.3) to decapsulate ct_0 and ct_1 instead of his secret key. More concretely, S_1 generates $([\![a]\!],a)\leftarrow\mathsf{KGen}(1^\kappa)$ and sends $[\![a]\!]$ to $\mathcal{A}$. $\mathcal{A}$ has access to group elements $[\![1]\!]$, $[\![a]\!]$ and all the group elements that are outputs of the random oracles. When $\mathcal{A}$ sends s,T, S_1 computes ct_0, ct_1 according to protocol and uses the extraction mechanism to learn $\alpha_{0,0},\alpha_{0,1},\alpha_{1,0},\alpha_{1,1}$ such that for all $d\in\{0,1\}$, $\mathsf{ct}_d=\alpha_{d,0}[\![1]\!]+\alpha_{d,1}[\![a]\!]$. $\mathcal{A}$' computes $\mathsf{k}'_d=(\alpha_{d,0}+\alpha_{d,1}a)\mathsf{pk}$. He queries H_2 on k'_0 and k'_1 according to protocol and outputs k_0, k_1. Thus,

$$\Pr[\mathsf{D}(z,(\mathsf{S},\mathcal{A})_\Pi)=1]=\Pr[\mathsf{D}(z,(\mathsf{S}_1,\mathcal{A}))=1].$$

S_2 is identical to S_1 except that he sometimes outputs a uniform string instead of a k with the guarantee that this is the case for at least one of the keys k_0, k_1 for each subsession. We show that based on the discrete log assumption the two sender are indistinguishable as follows. Given a discrete log instance $[\![a]\!]$, the reduction sends $[\![a]\!]$ as pk to $\mathcal{A}$. It responds to all oracle queries to H_1 with $\beta_n[\![a]\!]$ for uniform $\beta_n\in\mathbb{Z}_p\setminus\{0\}$, where n is the query number. This is statistically indistinguishable from responding with $\beta_n[\![1]\!]$ as long as $[\![a]\!]$ is a generator for $\mathcal{G}$.

whenever $\mathcal{A}$ makes a query (pk,d_j,T_j) to H_0 (let j be the query number), the reduction extracts and stores $(\alpha_{j,0},\alpha_{j,1})$ such that

$$T_j=\alpha_{j,0}[\![1]\!]+\alpha_{j,1}[\![a]\!].$$

Further, for any $i,n\in[q]$ with $i<j$, $n<j$ and the ith query is a query to H_1 and the nth a query to H_2, the reduction performs a check. We will explain this check later.

Whenever $\mathcal{A}$ makes a query (pk,d_i,r_i) to H_1 (let i) be the query number), the reduction stores β_i such that

$$\mathsf{H}_1(\mathsf{pk},d_i,r_i)=\beta_i[\![a]\!].$$

Further, for any $j,n\in[q]$ with $j<i$, $n<i$ and the jth query is a query to H_0 and the nth a query to H_2, the reduction performs a check. We will explain this check later.

Whenever $\mathcal{A}$ makes a query (s_n,T_n,k'_n) to H_2 (let n be the query number), the reduction extracts and stores $(\gamma_{n,0},\gamma_{n,1})$ such that

$$\mathsf{k}'_n=\gamma_{n,0}[\![1]\!]+\gamma_{n,1}[\![a]\!].$$

Further, for any $j,i\in[q]$ with $j<n$, $i<n$ and the jth query is a query to H_0 and the ith a query to H_2, the reduction performs a check. We explain this check now.

For any $i,j,n\in[q]$ for which $(\alpha_{j,0},\alpha_{j,1})$, β_i and $(\gamma_{n,0},\gamma_{n,1})$ are defined, the reduction tries to solve the quadratic equation for $\hat{a}_{i,j,n}$ (using the Tonelli-Shanks algorithm)

$$(\beta_n-\alpha_{j,1})\hat{a}^2_{i,j,n}-(\gamma_{j,1}+\alpha_{j,0})\hat{a}_{i,j,n}-\gamma_{j,0}=0 \mod p.$$

If there is a non-trivial solution s, the reduction checks whether $s[\![1]\!]=[\![a]\!]$ and if yes, the reduction outputs the challenge discrete log $s=a$ to the challenger.

We assume that $\mathcal{A}$ always make such queries for his s,T to $\mathsf{H}_1,\mathsf{H}_0$. If not, the reduction will make the queries for him.

For each subsession, there is at least one encapsulated key k′ such that if $\mathcal{A}$ queries k' to H_2, this check will successfully extract a discrete log solution for the following reason. Recap that the candidate encapsulations are defined as

$$\widehat{\mathsf{ct}}_{j,n}=(\beta_n-\alpha_{j,1})[\![a]\!]-\alpha_{j,0}[\![1]\!].$$

Further, the corresponding candidate keys are defined as

$$\hat{\mathsf{k}}_{d,i,j}:=\mathsf{H}_2(s_{i,j},T_j,(a\beta_n-a\alpha_{j,1}-\alpha_{j,0})[\![a]\!]),$$

for a matching $d\in\{0,1\}$ and $s_{i,j}$. Matching this input with a query to H_2 leads to the quadratic equation from above.

Now we need to argue that for each subsession at least one of the encapsulated keys k'_0, k'_1 of the actual messages s,T if queried to H_2 would lead to extracting a. Let $i,i',j\in[q]$ and $d\in\{0,1\}$ such that $T=T_j$, $i<i'$, the ith query to H_1 is for input (pk,d,r_i) and the i'th query to H_1 is for input $(\mathsf{pk},1-d,r_{i'})$ with $s=r_i+\mathsf{H}_0(\mathsf{pk},d,T)$ and $s=r_{i'}+\mathsf{H}_0(\mathsf{pk},d,T)$. First notice that when making the i'th query, there is a unique T_j such that

$$r_i+\mathsf{H}_0(\mathsf{pk},1-d,T_j)=r_{i'}+\mathsf{H}_0(\mathsf{pk},d,T_j)$$

holds. Therefore, in order to make the i' query, $\mathcal{A}$ needs to first query j, i.e. $j<i'$. Thus, β_j is sampled after $(\alpha_{i',0},\alpha_{i',1})$ have been chosen by $\mathcal{A}$. Therefore $\beta_n-\alpha_{i',1}\neq 0$ except with a probability of at most $2^{-\kappa+1}$. This ensures that computing k'_{1-d} is non-trivial or from a different perspective, that the quadratic equation has a non-trivial solution. Notice that a solution is guaranteed since a exists and is a solution. Thus if k'_{1-d} is queried to H_2, the reduction can compute a and terminates. S_2 will output a uniform string u instead k_{1-d}. d can be reconcstructed by observing the query order.

Based on the ϵ-hardness of discrete log,

$$|\Pr[\mathsf{D}(z,(\mathsf{S}_1,\mathcal{A}))=1]-\Pr[\mathsf{D}(z,(\mathsf{S}_2,\mathcal{A}))=1]| \le \epsilon+q_{\mathsf{ssid}}2^{-\kappa+1}.$$

As a last step, we use S_2 to construct $\mathcal{A}$'. $\mathcal{A}$' will use the same strategy as S_2 to reconstruct the subsession choice bits from $\mathcal{A}$. As S_2 does, $\mathcal{A}$' samples $(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KGen}$ and sends it to $\mathcal{A}$. Upon receiving $\mathsf{sid},\mathsf{ssid},s_{\mathsf{ssid}},T_{\mathsf{ssid}}$ from $\mathcal{A}$, $\mathcal{A}$' checks whether $s_{\mathsf{ssid}},T_{\mathsf{ssid}}$ are identical to $s_{\widehat{\mathsf{ssid}}},T_{\widehat{\mathsf{ssid}}}$ of a previous subsession. If that is the case, it looks up its message $(\mathsf{sid},\widehat{\mathsf{ssid}},\hat{b},\hat{\mathsf{k}}_{\hat{b}},\widehat{\mathsf{ssid}'})$ and sends message $(\mathsf{sid},\mathsf{ssid},\hat{b},\hat{\mathsf{k}}_{\hat{b}},\widehat{\mathsf{ssid}})$ to $\mathcal{F}_{\mathsf{mNIOT}}$. Otherwise, it follows the strategy

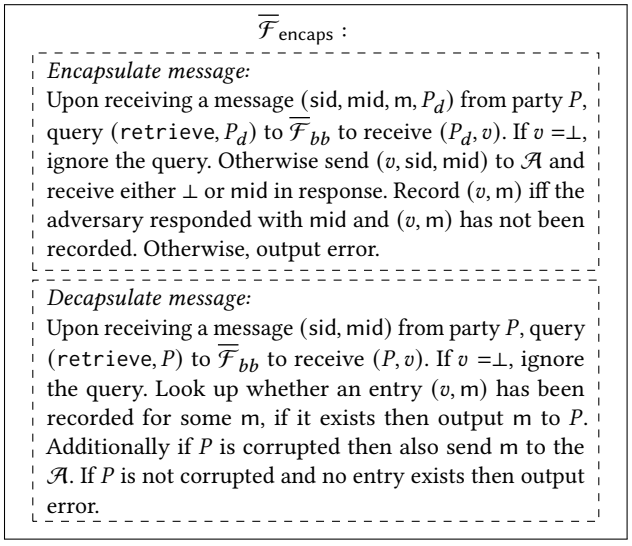

Figure 17: The figure shows the $\overline{\mathcal{F}}_{\text{encaps}}$ functionality.

of S_2 to extract subsession choice bit b and the corresponding key k_b. It then sends message (sid, ssid, b, k_b, $\emptyset$) to $\mathcal{F}_{\text{mNIOT}}$.

After all subsessions are over, $\mathcal{A}$' outputs the output of $\mathcal{A}$. Therefore,

$$|\Pr[\text{D}(z, (\text{S}_2, \mathcal{A})) = 1] = \Pr[\text{D}(z, (\mathcal{A}', \mathcal{F}_{\text{mNIOT}})) = 1].$$

□

F KEMS AND IMPLICIT AUTHENTICATION

As discussed in Section 3.3 in many practical use cases it is sufficient to use a KEM (together with a PKI) for the purposes of providing authentication. However, when using a KEM-only approach we achieve implicit authentication, that is, we are ensured that only the owner of the certified key will be able to receive the sent message. In contrast, an explicitly authenticated protocol ensures to the authenticating party that the authenticated party is present and has engaged in the protocol.

In this section we formalize a global-UC functionality which permits for implicit authentication through the use of global PKI. We again model our PKI through the use of the global bulletin-board functionality $\overline{\mathcal{F}}_{bb}$ as depicted in Figure 3. However, instead of a certification functionality $\overline{\mathcal{F}}_{\text{cert}}$ we instead define an encapsulation functionality $\overline{\mathcal{F}}_{\text{encaps}}$ which models the case of a party encrypting a message towards a certified public key, only the holder the certified (secret) key will be able to decrypt and receive the message. This functionality is show in Figure 17.

We now turn our attention to key confirmation and how it can be used to elevate from implicit authentication to explicit authentication. Reconsider the protocol in Figure 6, if we now simply focus on the KEM functionality (removing use of the MAC) we can prove through similar arguments to Theorem 3.3 that this protocol realizes the $\overline{\mathcal{F}}_{\text{encaps}}$ functionality. More informally the protocol provides implicit authentication. Through the addition of the MAC (used in Figure 6) we can then transform the KEM-only protocol into one that achieves explicit authentication.

More generally, we can extend this relationship between implicit authentication, explicit authentication and key confirmation. To transform any protocol which realizes in the $\overline{\mathcal{F}}_{\text{encaps}}$ functionality to also realize the $\overline{\mathcal{F}}_{\text{cs}}$ functionality one can make use of complier which performs key confirmation, as defined below. In practice this key confirmation step could be the MAC step performed in Figure 6. This therefore provides an alternate route to prove the security of Theorem 3.3.

Definition F.1. *A compiler C is an algorithm that takes as input descriptions of protocols and outputs descriptions of protocols. A* key confirmer *is a compiler C where for any protocol Π which realizes the $\overline{\mathcal{F}}_{\text{encaps}}$ functionality, the protocol $C(\Pi)$ realizes the $\overline{\mathcal{F}}_{\text{cs}}$ functionality.*

One could take a similar approach to formalize implicit authentication for OT-based protocols which would be sufficient for our purposes. Here assuming secure reuse of the public key is possible, then the public key can be certified as part of a PKI to provide these authentication properties. However, as discussed in subsection 3.3, elevating such an OT protocol to explicit authentication through defining an efficient key confirmation step seems to be difficult.