Efficient Covert Two-Party Computation

Stanisław Jarecki

University of California, Irvine
sjarecki@uci.edu

Abstract. Covert computation (of general functions) strengthens the notion of secure computation, so that the computation hides not only everything about the participants' inputs except for what is revealed by the function output, but it also hides the very fact that the computation is taking place, by ensuring that protocol participants are indistinguishable from random beacons, except when the function output explicitly reveals the fact that a computation took place. General covert computation protocols proposed before have non-constant round complexity [16, 4] and their efficiency is orders of magnitude away from known non-covert secure computation protocols. Furthermore, [8] showed that constant-round covert computation of non-trivial functionalities with black-box simulation is impossible in the plain model.

However, the lower-bound of [8] does not disallow constant-round covert computation given some relaxation in the computation model. Indeed, in this work we propose the first constant-round protocol for covert Two-Party Computation (2PC) of general functions, secure against malicious adversaries under concurrent composition, assuming the Common Reference String (CRS) model. Our protocol is a covert variant of a well-known paradigm in standard, i.e. non-covert, secure 2PC, using cut-and-choose technique over O(security parameter) copies of Yao's garbled circuit protocol, and its efficiency is only a constant factor away from non-covert secure 2PC protocols that use cut-and-choose over garbled circuits.

An essential tool in the protocol is a concurrently secure covert simulation-sound Conditional KEM (CKEM) for arithmetic languages in prime-order groups. We show that the Implicit Zero-Knowledge arguments in the CRS model of Benhamouda et al. [2] provide covert CKEM's for all languages needed in our covert 2PC protocol. We also show that in the Random Oracle Model the covert CKEM's of [11] also satisfy concurrent security and simulation-soundness. The ROM-based covert CKEM's of [11] match the cost of known ROM-based NIZK's for the same languages, while the CRS-model CKEM's of [2] are (only) 2-4 times more expensive.

1 Introduction

Covert computation addresses a security concern which is unusual for cryptography, namely how to hide the very fact that a (secure) protocol executes. Such hiding of a protocol instance is possible if the public channels connecting the communicating parties are *steganographic* in the sense that they have intrinsic entropy. A protocol is covert if its messages can be efficiently injected into such

channels in a way that the resulting communication cannot be distinguished from the a priori behavior of these channels. A standard example of such channel is a *random channel*, a.k.a. a *random beacon*, which can be implemented e.g. using protocol nonces, padding bits, time stamps, and various other communication (and cryptographic!) mechanisms which exhibit inherent (pseudo)entropy. Given a random channel, if protocol messages are indistinguishable from random bitstrings, such messages can be injected into the channel, and the protocol counterparty can interpret the information received on the channel as a protocol message. The participants must agree on the time they use such channels to run a protocol, so they know which bits to interpret as protocol messages, but this can be public information because if the protocol is covert then the exchanged messages cannot be distinguished from the a priori behavior of the random channel.[1]

Covert computation was formalized for the two-party setting by Von Ahn, Hopper and Langford in [16], and then generalized to the multi-party setting (and re-formulated) by Chandran et al. in [4], as a protocol that lets the participants securely compute the desired functionality on their joint inputs, with the additional property that each participant cannot distinguish the others from "random beacons", i.e. entities that send out random bitstrings of fixed length instead of prescribed protocol messages, unless the function output determines that it should be revealed. In other words, in covert computation the computed function outputs an additional *reveal* bit: If this bit is zero then each participant remains indistinguishable from a random beacon to the others, but if the reveal bit is one then the participants learn the function output, and in particular learn that a computation took place, i.e. that they were interacting not with random beacons but with counterparties executing the same protocol (whose inputs into this protocol, moreover, made the reveal bit in the function output equal to 1).

Q & A on Covert Computation. *Motivation:* Who would care to compute a function while hiding this very fact from other potential protocol participants (unless the function output reveals it)? One example given by Chandran et al. [4] is a company C that observes worrisome activity on its network: If C could determine that other companies observe similar activity, this information could help all of them fight against a hacking attack, but the very fact of engaging in such protocol reveals that company C observes some worrisome activity, and C might have business reasons to hide this. In another example, a military or intelligence agent A who detects some suspicious information could determine if similar information is detected by other agents, but no one, not even an active participant interacting with A cannot even detect that A engages in this protocol unless they have the information that "matches" A's knowledge. In general, covert computation can be used for any form of authentication whose participants want to

[1] Works on steganographic *communication* [9] imply that random messages can be embedded into any non-uniform random channel with sufficient entropy, hence in particular once we know how to communicate and/or compute covertly assuming uniform random channels then we can also carry this communication/computation over any (non-uniform) steganographic channel with sufficient entropy.

remain undetectable except to counter-parties whose inputs (certificates, permissions, passwords, enviromental observersations, or what have you) matches their authentication policy. For example, if two spies want to authenticate one another in a foreign country, they would surely like to be able to do so in a way that prevents anyone from detecting that this authentication protocol instance is taking place. If the spies authenticated each other using covert computation, the only way their presence can be detected is by an active attacker who is privy to the authentication tokens that match their authentication policy.

Random Channels and Synchronization: But isn't it always possible to detect a presence of some party on a network by just observing whether this party sends out some messages? Moreover, simple counting of the number and size of these messages should suffice to conclude whether or not it follows a given protocol. Indeed, this is why we must assume that protocol participants have access to random channels: A network entity cannot hide the fact that it sends out messages, but if the normal communication this entity emits exhibits some entropy (e.g. in protocol nonces, timing, padding, audio and video signals) this entropy can be used to create a steganographic channel which can be used for covert MPC protocol messages. Another question concerns synchronization: Even if company or agent A in the above examples is connected by a random channel to potential counterparties in a covert MPC protocol, how would she know when to start the protocol? One answer is that the protocol start should be a public convention, e.g. the first message after time 12:00 on each day could contain a protocol instance. Party P willing to participate would always interpret such messages as protocol messages, and if their sender was not engaging in the protocol, or it engaged but the reveal bit in the computation output is 0, party P would reject in such instance.

Covert MPC vs. Steganography: Covert MPC does not trivially follow by using steganography [9] to establish covert communication channels between potential protocol participants and running standard MPC over them. First, covert channels require prior key distribution which is not always possible, e.g. in the general authentication application above. Second, even if potential participants did have pre-shared keys, they still might want to hide whether or not they actually engage in this protocol.

Covert MPC vs. Secure MPC: Secure computation is considered to be a blueprint for every security task: Whatever security property we want some network interaction to achieve, we can abstract it as a secure computation of an idealized functionality, and we can achieve it by MPC for this functionality. However, secure computation does leak one additional "bit" of information, namely it is not designed to hide whether or not some entity engages in the protocol, which in many applications (see examples above) is a very essential information. To give an extreme example: If CIA is the only organization whose agents follow some secure authentication protocol then a man-in-the-middle attacker will be trivially able to identify (potential) CIA agents. A *covert computation* strenghtens secure computation to hide this one remaining bit, and allows protocol participation to be undetectable even to active protocol participants except

(and this "escape clause" seems unavoidable) if the function output itself determines that the outputs, and hence also the fact of protocol participation, should be revealed to the participants. What we show is that this strenghtening of secure computation to covertness can be achieved in some sense "for free", i.e. at costs which are comparable to those born by known standard, i.e. non-covert, secure computation protocols. Moreover, in the process we show general tools for covert enforcement of honest protocol execution which can be re-used (and further improved) in efficient covert protocols for specific functions of interest.

Previous Works on Covert Computation. Von Ahn et al. [16] proposed the first covert two-party computation (2PC) protocol. Their protocol performed $O(\tau)$ repetitions, for τ a security parameter, of Yao's garbled circuit evaluation (with the circuit extended to compute an additional hash function), but this protocol guaranteed only secrecy against malicious participants, and not output correctness. The covert multi-party computation protocol of [4] realized a covert computation functionality against malicious participants (and in particular guaranteed output correctness), but it also used $O(\tau)$ rounds and its efficiency was several orders of magnitude away from known non-covert MPC protocols: Each party was covertly proving that it followed a GMW MPC protocol on committed input by casting it as an instance of a Hamiltonian Cycle problem, and that proof internally used Yao's garbled circuits for checking correctness of committed values. Moreover, Goyal and Jain subsequently showed that [8] the non-constant round protocol is necessary for achieving computation (with black-box simulation) against malicious adversaries, at least in the standard MPC model, i.e., without access to trusted parameters or public keys.

On the plus side, some constant-round covert protocols secure against malicious adversaries are known as well: Jarecki [10] showed a covert Authenticated Key Exchange (AKE) with $O(1)$ rounds and public key operations, but this protocol satisfied a game-based AKE definition, and in particular it was not a covert secure computation of any function. Assuming a Random Oracle Model (ROM), Cho et al. [5] exhibited practical constant-round covert computation protocols, secure against active adversaries, for two non-trivial functionalities, namely for string equality and set intersection. (Moroever, Cho et al. strengthen the definition of covert computation of [4] to include concurrent self-composition, and we adopt this stronger notion of covert computation in this work.) However, in addition to relying on ROM, their constructions are custom-made for functionalities dealing with equality or set-membership checking, and it is not clear how they can be extended to computation of general functions.

Our Result: Efficient Covert Concurrent 2PC. This leaves a natural open question whether general two-party functions can be computed covertly by a constant-round protocol, or even, better, by a protocol whose assumptions, the security guarantees, and efficiency, are all comparable to those of the currently known constant-round standard, i.e. non-covert, secure 2PC protocols. We answer all these questions affirmatively with a construction of a constant-round protocol for covert 2PC of general functions, secure against malicious adversaries. Our protocol follows the well-known paradigm for standard, i.e. non-covert, se-

cure 2PC, using the cut-and-choose technique over $O(\tau)$ copies of Yao's garbled circuit protocol, and its efficiency is asymptotically the same as non-covert secure 2PC protocols based on cut-and-choose and Yao's garbled circuits, except that we do not handle OT extension. Concretely, for a function on n-bit inputs computable by a Boolean circuit with c gates, the proposed protocol requires 9 rounds, $O(n\tau)$ exponentiations and $O(c\tau)$ symmetric cipher operations.[2] The protocol works in the Common Reference String (CRS) model, it maintains covertness under concurrent composition, and it is secure under the Decisional Diffie-Hellman (DDH) assumption in the standard model (i.e. without ROM).

Enabling Tool: Covert Zero-Knowledge. Assuming random channels, covert *communication* is essentially as easy as secure communication: Since block ciphers are assumed to be pseudorandom functions many standard encryption (or even authenticated encryption) modes have ciphertexts which in addition to protecting the plaintext are also indistinguishable from random bitstrings. Several known public-key encryption schemes, e.g. Cramer-Shoup encryption [6], also have ciphertexts that are indistinguishable from a tuple of random group elements (assuming DDH), and random group elements, e.g. in a prime-order subgroup of modular residues, are easy to encode as random bitstrings. Assuming honest-but-curious participants, Von Ahn et al. [16] showed that general covert *computation* is also not more difficult than general secure computation: Given block ciphers whose outputs are indstinguishable from random strings, Yao's garbled circuit construction can be adjusted so that a garbled circuit for c-gates looks like $4c$ random ciphertexts even to the evaluator (except for whatever is revealed by the output, but that can be set to a random string if the "reveal bit" in the output evaluates to 0), and because ElGamal encryption is covert under DDH, an honest-but-curious secure OT based on ElGamal encryption, like the Naor-Pinkas OT [14], also remains covert.

However, it is difficult to achieve covert 2PC/MPC protocols secure against *malicious* adversaries, and this is because of the lack of efficient covert counterparts to standard mechanisms for enforcing honest behavior in protocols. For example, the two chief tools used to enforce honest behavior in Yao's garbled circuit protocol are (1) Zero-Knowledge (ZK) proofs, e.g. to show that the sender entered consistent inputs into the Oblivious Transfer (OT) and into multiple garbled circuit instances, and (2) opening a committment to show that the committed value is correctly formed, which is the basis of the general "cut-and-choose" method for enforcing honest behaviour. Either of these tools would violate covertness because both are publicly verifiable, and in particular, the first party who sends a ZK proof or an opening of a commitment becomes distinguishable from random noise by the counterparty.

The enabling tool we use to enforce honest behavior in protocol are efficient realizations of a covert *Conditional Key Encapsulation Mechanism* (CKEM) for a

[2] The cryptographic security parameter τ can be replaced by a statistical security parameter k in both expressions, but the result would meet a reduced security goal, where with probability 2^{-k} the adversary can make an honest counterparty compute some function of its choice on their joint inputs.

wide class of discrete-log-based languages, including statements that the Cramer-Shoup encryption [6] ciphertexts or a response in the Oblivious Transfer (OT) of Aiello et al. [1], are computed correctly, or that an encrypted value is a bit, or that a commitment decommits to a given plaintext (with the decommitment as the prover's witness). A CKEM can be thought of a language-based envelope, a KEM-version of Conditional Oblivious Transfer [7], or an interactive counterpart of Smooth Projective Hash Functions (SPHF): A CKEM for language L is a protocol which allows a sender S with input x to transmit a random key K to receiver R with input (x, w) if and only if w is a witness for x in L. A *covert* CKEM additionally assures that an interaction with either S or R is indistinguishable from an interaction with a random beacon. In particular, even knowing the witness w for x a malicious receiver cannot distinguish $\mathsf{S}(x)$ from a random beacon: It computes a key K which is the same as the key computed by the sender, but since the key is random, it still does not know if the sender was a real party or a source of randomness. A covert CKEM can thus provide a counterpart to a zero-knowledge proofs: Instead of asking party A in a 2PC protocol to explicitly prove that it creates its messages correctly, which makes A's presence publicly verifiable, parties B and A run a covert CKEM for the same language as resp. S and R, and then use key K to (covertly) encrypt subsequent protocol messages: If A's messages were not formed correctly, A will not be able to derive B's key K, which will make B's subsequent messages indistinguishable from random in A's view.

A Covert CKEM for language L was introduced as "Zero-Knowledge Send" by Chandran et al. [4] and achieved for general languages, but their construction reduced L to an NP-complete problem (Hamiltonian Cycle) and used garbled circuit evaluation at each step of a ZK proof for this problem. By contrast, we are looking for covert CKEM's for a class of languages, which we call *Linear Map Image* languages, which are discrete-log-based languages that have practical (HV)ZK proofs (and efficient SPHF systems), and we want the CKEM costs to be in a similar ballpark as the cost of these proofs. A natural starting point for a CKEM for a discrete-log-based language are well-known efficient SPHF schemes for such languages: If B uses an SPHF system on x and defines key K as the SPHF hash value then A can derive the same key K using its witness w for x in L. However, SPHF by itself cannot replace a zero-knowledge proof in a larger protocol, because it does not offer a way for the simulator playing the role of A and simulating A's message x, e.g. with random noise, to recover B's key K and continue the simulation on A's behalf in subsequent protocol rounds. Zero-knowledge works because the simulator can send $x \notin \mathsf{L}$ on A's behalf and then simulate A's ZK proof as if x was in L. By contrast, if $x \notin \mathsf{L}$ then an SPHF on x will hide B's key K in an information-theoretic sense.

This insufficiency of SPHF's to replace zero-knowledge proofs in MPC protocols was recognized by Benhamouda et al. [2] who added two essential properties to SPHF's for discrete-log-based languages: First, they added (concurrent) zero-knowledge, i.e. the ability for the simulator in position of the global trapdoor in the CRS to derive S's key K even on the wrong statement $x \notin \mathsf{L}$. Secondly,

they added simulation-soundness, i.e. the assurance that a cheating receiver cannot recover S's key K for a protocol instance executing on a wrong statement $x \notin \mathsf{L}$ even if the adversary concurrently engages with the simulator who recovers keys corresponding to multiple protocol instances running on any *other* wrong statements $x' \notin \mathsf{L}$. These two properties, simulation-soundness and (concurrent) zero-knowledge are needed of zero-knowledge in the standard compiler from (concurrent) 2PC secure against honest-but-curious adversaries to (concurrent) 2PC secure malicious parties. Benhamouda et al. called this class of protocols (concurrent) simulation-sound *Implicit Zero-Knowledge* (IZK) Arguments, and they showed an efficient construction for such IZK's for a wide class of discrete-log-based languages (including the those listed above). The notion of IZK defined by [2] does not ask for the iZK protocol to be covert, because the goal of the Implicit ZK Arguments of [2] was reduction of round complexity in the honest-but-curious to malicious-security protocol compilation: The IZK schemes they show take only 2 rounds in the CRS model under DDH, compared to 3 rounds for zero-knowledge proofs of comparable efficiency for the same language class. Here we extend the IZK notion of [2] to include covertness (and we call the resulting notion *Covert CKEM*), and we observe that the simulation-sound IZKs of [2] for several discrete-log-based languages of our interest already satisfy this stronger notion without any changes (except for the trivial change of encoding (pseudo)random group elements as (pseudo)random bitstrings).

In addition we show that assuming ROM the two-round CKEM's of [11] for any language with a special sigma-protocol (which includes all Linear Map Image languages), which were shown to satisfy a weaker notion of covertness in [11], also satisfy the covert simulation-sound and zero-knowledge CKEM notion. This ROM-based covert CKEM realization is relevant for practical purposes because it constructs covert CKEM's with the same efficiency as NIZK's for these lanuages which are used in standard, i.e. *non-covert*, maliciously-secure 2PC protocols based on cut-and-choose and Yao's garbled circuits.

Organization. In Section 3 we define concurrent covert 2PC for arbitrary functions. In Section 4 we list some covert protocol building blocks. In Section 5 we define covert CKEM. In Section 6 we show covert CKEM's for languages we use in the covert 2PC protocol. Finally in Section 7 we describe the concurrent covert 2PC protocol for arbitrary functions, and sketch the proof of its security.

Revision Notes. In a previous version of this paper we used too simplistic model of CKEM covertness to deal with the standard-model CKEM's of Benhamouda et al. [2]: Specifically, the sender in these CKEM's can be distinguished from a random beacon given the CKEM simulation trapdoor. In this version we include a relaxation of sender covertness to simulation-covertness, i.e. that sender is indistinguishable from a random beacon given an oracle access to a simulator on other CKEM instances. This notion suffices for CKEM applications in MPC protocols like our covert 2PC protocol of Section 7, and it holds for the CKEM's of [2]. In the previous version we also did not point out the in ROM one can use more efficient CKEM's of [11] instead of the standard-model CKEM's of [2]. In

the next revision we plan to (1) give a more intuitive overview of the covert 2PC protocol of Section 7, and (2) edit the proof sketches for this protocol.

2 Preliminaries

Notation. If a, b are bitstrings then $|a|$ is the length of a, $a|b$ is the concatenation of stings a and b, and $a[i]$ is the i-th bit of a. If n is an integer then $[n] = \{1, ..., n\}$. We write $y \leftarrow \mathsf{P}(x)$ when y is an output of a (randomized) procedure P on input x, and $y \leftarrow \mathsf{S}$ when y is sampled from uniform distribution over set S. We write $y \in \mathsf{P}(x)$ if there is randomness r s.t. $\mathsf{P}(x; r)$ outputs y. We say $(a, b) \leftarrow [A(x), B(y)]$ if a, b are the local outputs of algorithms resp. A, B interacting on local inputs resp. x, y. If L is a language in NP then $\mathcal{R}[\mathsf{L}]$ is a relation s.t. $(x, w) \in \mathcal{R}[\mathsf{L}]$ if w is an efficiently verifiable witness for $x \in \mathsf{L}$.

We call two-party protocol (A, B) *regular* if the number of rounds and length of all messages is a function of the security parameter, and not the local inputs of either party. If P is an interactive algorithm in a regular two-party protocol then $\mathsf{P}^{\$(\tau)}$ denotes a random beacon corresponding to P, which sends random bitstrings of the same length as P's messages in every protocol round. If P is an interactive algorithm then $P_{\&\mathsf{Out}}(x)$ is a wrapper which runs $P(x)$ and includes P's final local output in its last message. For any algorithm Setup and oracles P_0, P_1 we say that $\{\mathcal{A}^{P_0(x_0)}(z)\} \approx \{\mathcal{A}^{P_1(x_1)}(z)\}$ *for* $(x_0, x_1, z) \leftarrow \mathsf{Setup}(1^\tau)$ if for every efficient $\mathcal{A}$ quantity $|p^0_{\mathcal{A}} - p^1_{\mathcal{A}}|$ is negligible where $p^b_{\mathcal{A}} = \Pr[1 \leftarrow \mathcal{A}(z)^{P_b(x_b)} \,|\, (x_0, x_1, z) \leftarrow \mathsf{Setup}(1^\tau)]$, where the probability goes over the coins of Setup, $\mathcal{A}$, and P_b.

Covert Encodings. In all protocols we will use all communicated values are either bitstrings or elements of a prime-order group G. In the latter case what is sent on the wire is not a group element $a \in G$ itself, but its covert encoding $\mathsf{EC}(a)$. A covert encoding is a randomized function $\mathsf{EC} : G \to \{0,1\}^{p(\tau)}$ defined for some polynomial p, s.t. a random variable $\{\mathsf{EC}(a; \mathsf{r})\}$, induced by a random choice of r and a sampled uniformly at random in G, is statistically close to a random string of length $p(\tau)$. Moreover, there must exist a corresponding decoding procedure DC s.t. $\mathsf{DC}(\mathsf{EC}(a; \mathsf{r})) = a$ for all r and all $a \in G$. For example, if G is a subgroup of prime p order of a group of residues modulo prime p' s.t. $p' = pr + 1$ for $gcd(p, r) = 1$, then $\mathsf{EC}(a)$ can pick $b \leftarrow \mathbb{Z}_{p'}$ and $i \leftarrow [2^\tau]$ and output $v = (a \cdot (b)^p) \bmod p' + i \cdot p'$, where $\cdot$ and $+$ are operations on integers, while $\mathsf{DC}(v)$ outputs $w^s \bmod p'$ for $w = v^r \bmod p'$ and $s = r^{-1} \bmod p$.

3 Concurrent Covert Two-Party Computation

We provide the definition of concurrent covert computation of two-party functions, which is a close variant of the definition which appeared recently in [5]. Intuitively, the differences between the covert computation of a two-party functionality F and the secure computation for F is that (1) F's inputs and outputs are extended to include a special sign $\perp$ designating non-participation; (2) F is

restricted to output a non-participation symbol $\perp$ to each party if the input of either party is $\perp$; and (3) the real-world protocol of either party on the non-participation input $\perp$ is fixed as a "random beacon", i.e. a protocol which sends out random bitstrings of fixed length independently of the messages it receives.

The definition of concurrent covert computation of [5], which we recall (and refine) below, follows the definition of stand-alone (i.e. "single-shot") covert computation given by Chandran et al. [4], here restricted to the two-party case. The definition casts this notion in the framework of universal composability (UC) by Canetti [3], but the composability guarantee it implies is restricted to concurrent self-composition because it guarantees only self-composability of covert computation for *functions*, and not for general reactive functionalities as in the case of UC definition [3]. The reason for this restriction is two-fold: First, concurrent covert computation for arbitrary efficiently computable functions already provides a significant upgrade over the "single-shot" covert computation notion of [4], and achieving it efficiently presents sufficient technical challenges that justify focusing on this restricted notion. Secondly, composing functionally distinct covert protocols poses conceptual challenges: Consider a protocol Π implemented by a protocol Π_1 which runs Π_2 as a subroutine, and note that the outputs of subroutine Π_2 can reveal the participation of an honest party in Π before Π completes. Here we focus on concurrent composition of covert computation of two-party function, and leave development of a framework for fully composable covert computation for future work.

On input $(\mathsf{Input1}, \mathsf{sid}, B, x)$ **from party** A**:**
Record $(\mathsf{Input1}, \mathsf{sid}, A, B, x)$ and send $(\mathsf{Input1}, \mathsf{sid}, A, B)$ to $\mathcal{A}^*$.

On input $(\mathsf{Input2}, \mathsf{sid}, A, y)$ **from party** B**:**
Record $(\mathsf{Input2}, \mathsf{sid}, A, B, y)$ and send $(\mathsf{Input2}, \mathsf{sid}, A, B)$ to $\mathcal{A}^*$.

Given records $(\mathsf{Input1}, \mathsf{sid}, A, B, x)$ and $(\mathsf{Input2}, \mathsf{sid}, A, B, y)$ compute

$$(z, v) \leftarrow \begin{cases} (\perp, \perp) & \text{if} \quad x = \perp \ \vee \ y = \perp \ \vee \ g(x, y) = 0 \\ f(x, y) & \text{otherwise} \end{cases}$$

and record $(\mathsf{Output}, \mathsf{sid}, A, z)$ and $(\mathsf{Output}, \mathsf{sid}, B, v)$.
If A is corrupt, send $(\mathsf{Output}, \mathsf{sid}, z)$ to A.
If B is corrupt, send $(\mathsf{Output}, \mathsf{sid}, v)$ to B.

On input $(\mathsf{Output}, \mathsf{sid}, P, \mathsf{release?})$ **from** $\mathcal{A}^*$**:**
Retrieve record $(\mathsf{Output}, \mathsf{sid}, P, w)$ (ignore if record does not exist).
If $\mathsf{release?} = \mathsf{T}$ then send $(\mathsf{Output}, \mathsf{sid}, w)$ to P.
If $\mathsf{release?} = \mathsf{F}$ then send $(\mathsf{Output}, \mathsf{sid}, \perp)$ to P.

Fig. 1. Covert 2-Party Function Computation Functionality $\mathsf{F}_{\mathsf{C}(f|g)}$

Ideal and Real Models. The definition of the *ideal model* is the UC analogue of the ideal model of Chandran et al. [4], except that composability guarantees are restricted to self-composition. Covert computation is defined by functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ shown in Figure 1, where f, g are functions defined on pairs of bitstrings. We note that f and g can be randomized functions, in which case functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ picks the randomness which is appended to input (x, y) before computing g and f. The ideal process involves functionality $\mathsf{F}_{\mathsf{C}(f|g)}$, an ideal process adversary $\mathcal{A}^*$, an environment $\mathcal{Z}$ with some auxiliary input z, and a set of dummy parties, any number of which can be (statically) corrupted. Each party can specify its input to some instance of $\mathsf{F}_{\mathsf{C}(f|g)}$, which is either a bitstring or a special symbol $\perp$ indicating that there is no party which will participate in a given role, e.g. a requester or responder in this protocol instance. The *real model* is exactly as in the standard UC security model, except that the protocol of each real-world uncorrupted party which runs on input $\perp$ is a-priori specified as a random beacon protocol, i.e. such party sends out random bitstrings of lengths appropriate for a given protocol round.

Let $\mathsf{Ideal}_{\mathsf{F},\mathcal{A}^*,\mathcal{Z}}(\tau, z, r)$ denote the output of environment $\mathcal{Z}$ after interacting in the ideal world with adversary $\mathcal{A}^*$ and functionality $\mathsf{F} = \mathsf{F}_{\mathsf{C}(f|g)}$, on security parameter τ, auxiliary input z, and random input $\mathsf{r} = (\mathsf{r}_{\mathcal{Z}}, \mathsf{r}_{\mathcal{A}^*}, \mathsf{r}_{\mathsf{F}})$, as described above. Let $\mathsf{Ideal}_{\mathsf{F},\mathcal{A}^*,\mathcal{Z}}(\tau; z)$ be the random variable $\mathsf{Ideal}_{\mathsf{F},\mathcal{A}^*,\mathcal{Z}}(\tau, z, \mathsf{r})$ when r is uniformly chosen. We denote the distribution ensemble of variable $\mathsf{Ideal}_{\mathsf{F},\mathcal{A}^*,\mathcal{Z}}(\tau; z)$ by $\{\mathsf{Ideal}_{\mathsf{F},\mathcal{A}^*,\mathcal{Z}}(\tau, z)\}_{\tau\in\mathbb{N};z\in\{0,1\}^*}$. In the corresponding way we define $\mathsf{Real}_{\Pi,\mathsf{Adv},\mathcal{Z}}(\tau, z, \mathsf{r})$ as the output of $\mathcal{Z}$ after interacting with a real-world adversary Adv and parties running protocol Π on security parameter τ, input z, and random tapes $\mathsf{r} = (\mathsf{r}_{\mathcal{Z}}, \mathsf{r}_{\mathsf{Adv}}, \mathsf{r}_A, \mathsf{r}_B)$. In parallel to the ideal model, we define the corresponding distribution ensemble $\{\mathsf{Real}_{\Pi,\mathsf{Adv},\mathsf{F}}(\tau, z)\}_{\tau\in\mathbb{N};z\in\{0,1\}^*}$.

Definition 1. *Protocol Π realizes the concurrent two-party covert computation functionality $\mathsf{F} = \mathsf{F}_{\mathsf{C}(f|g)}$ if for any efficient adversary Adv there exists an efficient ideal-world adversary $\mathcal{A}^*$ such that for any efficient environment $\mathcal{Z}$,*

$$\{\mathsf{Ideal}_{\mathsf{F},\mathcal{A}^*,\mathcal{Z}}(\tau, z)\}_{\tau\in\mathbb{N};z\in\{0,1\}^*} \overset{c}{\approx} \{\mathsf{Real}_{\Pi,\mathsf{Adv},\mathsf{F}}(\tau, z)\}_{\tau\in\mathbb{N};z\in\{0,1\}^*}$$

Notes on Functionality $\mathsf{F}_{\mathsf{C}(f|g)}$. Functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ in Figure 1 is realizable only assuming secure channels. Without secure channels the adversary could hijack a protocol session an honest player wants to execute with some intended counterparty. However, the secure channel assumption does not substantially change the complexity of the protocol problem because the intended counterparty can itself be corrupted and follow an adversarial protocol. The second point worth mentioning is that functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ always delivers the output first to a corrupted party, whether it is party A or B, and if this output is not a non-participation symbol $\perp$ then in both cases the corrupted party can decide if the correct computation output should also be delivered to its (honest) counterparty or the honest counterparty's output will be modified to $\perp$. (Note that if an output of a corrupt party, say A, is $\perp$ then B's output is also $\perp$, hence it does not matter in this case whether the adversary sends $(\mathsf{Output}, \mathsf{T}, \mathsf{sid})$ or

(Output, F, sid) to $\mathsf{F}_{\mathsf{C}(f|g)}$.) It is necessarily the case that $O(1)$-round protocol without a trusted party is *unfair* in the sense that the party which speaks last gets its output but can prevent the delivery of an output to its counterparty. However, functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ affords this unfair advantage to both the corrupt requester and the corrupt responder. Indeed, a concrete protocol Π_{COMP} presented in Section 7 which realizes this functionality allows the corrupt party A to learn its output z and stop B from learning anything about its output v (simply by aborting before sending its last message to B). However, this protocol also allows the corrupt party B to prevent party A from being able to decide if its output z (learned in step A2 in Figure 4) is an output of $f(x, y)$ or a random value induced from an interaction with a random beacon: Only B's final message can confirm which is the case for A, but a corrupt B can send this message incorrectly, in which case an honest A will dismiss the tentative value z it computed and output $\perp$ instead.

4 Protocol Building Blocks for Covert 2PC

SPHF. A Smooth Projective Hash Function (SPHF) for a language family L parametrized by parameter π, is a tuple $(\mathsf{PG}, \mathsf{KG}, \mathsf{Hash}, \mathsf{PHash})$ s.t. $\mathsf{PG}(1^\tau)$ generates public parameters π and a trapdoor td which allows for efficient testing of membership in $\mathsf{L}(\pi)$, $\mathsf{KG}(\pi)$ generates key hk together with a *key projection* hp, $\mathsf{Hash}(\pi, x, \mathsf{hk})$ and $\mathsf{PHash}(\pi, x, w, \mathsf{hp})$ generate hash values denoted respectively H and projH s.t. for all τ, all (π, td) generated by $\mathsf{PG}(1^\tau)$, all $(\mathsf{hk}, \mathsf{hp})$ generated by $\mathsf{KG}(\pi)$, and all $(x, w) \in \mathcal{R}[\mathsf{L}(\pi)]$, we have that $\mathsf{Hash}(\pi, x, \mathsf{hk}) = \mathsf{PHash}(\pi, x, w, \mathsf{hp})$. (We omit parameter π in the calls to Hash and PHash if it is implicit.) This defines so-called KV-SPHF, named for Katz-Vaikuntanathan [12], where the (key,projection) pair is generated given only the public parameters, and smoothness holds even for wrong statements chosen as a function of the key projection. This is in contrast to the GL-SPHF, named for Gennaro-Lindell [15], where key generation can depend on the statement x.

CCA-Covert Public Key Encryption. Covertness of a public key encryption scheme in a Chosen-Ciphertext Attack, or *CCA covertness* for short, is a generalization of the standard notion of CCA security, where instead of requiring that ciphertexts of two challenge messages are indistinguishable from each other, we require that a ciphertext on any (single) challenge message is indistinguishable from a random bitstring, even in the presence of a decryption oracle. For technical reasons it suffices if interaction with the real PKE scheme is indistinguishable from an interaction with a simulator who not only replaces a challenge ciphertext with a random string but also might follow an alternative key generation and decryption strategy.

Formally, we call a (labeled) PKE scheme $(\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ *CCA covert* if there exist polynomial n s.t. for any efficient algorithm $\mathcal{A}$, quantity $\mathsf{Adv}_{\mathcal{A}}(\tau) = |p^0_{\mathcal{A}}(\tau) - p^1_{\mathcal{A}}(\tau)|$ is negligible, where $p^b_{\mathcal{A}}(\tau)$ is the probability that $b' = 1$ in the following game: Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(1^\tau)$, and let $\mathcal{A}^{\mathsf{D}(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$ output an encryption

challenge (m^*, ℓ^*). If $b = 1$ then set $\mathsf{ct}^* \leftarrow \mathsf{E}(\mathsf{pk}, m^*, \ell^*)$, and if $b = 0$ then pick ct^* as a random string of length $n(\tau)$. In either case set $b' \leftarrow \mathcal{A}^{\mathsf{D}(\mathsf{sk},\cdot,\cdot)}$, while oracle $\mathsf{D}(\mathsf{sk}, \cdot, \cdot)$ returns $\mathsf{D}(\mathsf{sk}, \mathsf{ct}, \ell)$ on any ciphertext,label pair s.t. $(\mathsf{ct}, \ell) \neq (\mathsf{ct}^*, \ell^*)$.

Notice that by transitivity of indistinguishability if PKE is CCA-covert then it is also CCA-secure. The other direction does not hold in general, but many known CCA-secure PKE's are nevertheless also CCA-covert. One example is RSA OAEP, but another is Cramer-Shoup PKE [6], and this is the one we will use here because its arithmetic structure allows for efficient OT extension and efficient covert CKEM's on associated languages (e.g. that a ciphertext encrypts a given plaintext). Recall that in Cramer-Shoup PKE $\mathsf{Kg}(1^\tau)$ chooses generator g of group G with prime order p of an appropriate length, sets a collision-resistant hash function H, picks $(x_1, x_2, y_1, y_2, z) \leftarrow (\mathbb{Z}_p^*)^5$, picks $(g_1, g_2) \leftarrow (G \backslash 1)^2$, sets $(c, d, h) \leftarrow (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, g_1^z)$, and outputs $\mathsf{sk} = ((g, G, p, \mathsf{H}), x_1, x_2, y_1, y_2, z)$ and $\mathsf{pk} = ((g, G, p, \mathsf{H}), g_1, g_2, c, d, h)$. Encryption $\mathsf{E}_{\mathsf{pk}}(\mathsf{m}, \ell)$, assuming the message space is G, picks $r \leftarrow \mathbb{Z}_p$, sets $(u_1, u_2, e) \leftarrow (g_1^r, g_2^r, \mathsf{m} \cdot h^r)$, $\xi \leftarrow \mathsf{H}(\ell, u_1, u_2, e)$, $v \leftarrow (cd^\xi)^r$, and outputs $\mathsf{ct} = (u_1, u_2, e, v)$. Decryption $\mathsf{D}_{\mathsf{sk}}((u_1, u_2, e, v), \ell)$ recomputes ξ, outputs $\perp$ if $v \neq u_1^{x_1 + \xi y_1} u_2^{x_2 + \xi y_2}$, and outputs $\mathsf{m} = e \cdot u_1^z$ otherwise. In Appendix A we show that the proof of CCA security of this PKE under the DDH assumption [6] can be extended to imply its CCA covertness. For notational convenience we will assume that the key generation Kg picks the group setting (g, G, p) as a *deterministic* function of security parameter τ. The restriction of the message space to group G is taken for convenience, since this is how we use it in the covert 2PC protocol, but one can easily extend it to general message spaces using covert symmetric encryption.

Covert Non-Malleable Commitments. It is well-known that CCA-secure PKE implements non-malleable commitment. However, to stress that sometimes no one (including the simulator) needs to decrypt, we define commitment $\mathsf{Com}_{\mathsf{pk}}(m)$ as a syntactic sugar for $\mathsf{E}_{\mathsf{pk}}(H(m))$ where H is a collision-resistant hash onto G, but we will not define a covert commitment notion, relying instead directly on the fact that $\mathsf{Com}_{\mathsf{pk}}(m)$ stands for $\mathsf{E}_{\mathsf{pk}}(H(m))$.

Covert Oblivious Transfer. Von Ahn et al. [16] used a covert version of Naor-Pinkas OT [14] for their covert 2PC secure against honest-but-curious adversaries. Here we will use a covert version of the OT of Aiello et al. [1] instead because it is compatible with CCA-covert Cramer-Shoup encryption and covert CKEM's of Section 6. Let E be the Cramer-Shoup encryption and let $\mathsf{pk} = ((g, G, p, \mathsf{H}), g_1, g_2, c, d, h)$. Define a 2-message OT scheme $(\mathsf{E}, \mathsf{OTrsp}, \mathsf{OTfin})$ on Rec's input b and Snd's input $m_0, m_1 \in G$ as follows:
(1) Rec picks $\mathsf{r} \leftarrow \mathbb{Z}_p$ and sends $\mathsf{ct} = \mathsf{E}_{\mathsf{pk}}(g^b; \mathsf{r})$ to Snd;
(2) Snd picks $\mathsf{r}' = (\alpha_0, \beta_0, \alpha_1, \beta_1) \leftarrow \mathbb{Z}_p^4$ and sends back to Rec message $\mathsf{otr} = (s_0, t_0, s_1, t_1)$ computed using $\mathsf{OTrsp}_{\mathsf{pk}}(\mathsf{ct}, m_0, m_1; \mathsf{r}')$ $\mathsf{ct} = (u_1, u_2, e, v)$, and , which sets $s_i = g_1^{\alpha_i} h^{\beta_i}$ and $t_i = {u_1}^{\alpha_i} (e/g^i)^{\beta_i} m_i$ for $i = 0, 1$;
(3) Rec runs $\mathsf{OTfin}_{\mathsf{pk}}(b, \mathsf{r}, \mathsf{otr})$ which outputs $m = t_b \cdot (s_b)^{-\mathsf{r}}$.

Covertness of E implies that Rec's message is indistinguishable from random even on access to the decryption oracle. It is also easily seen that Snd's message

is indisguishable from random if payloads (m_0, m_1) are random in G^2. Otherwise payload m recovered by Rec suffices to distinguish OTrsp and $\mathsf{OTrsp}^{\$(\tau)}$.

Covert Garbled Circuit. Von Ahn et al. [16] shows a covert version of Yao's garbling procedure GCgen_f for any $f : \{0,1\}^n \to \{0,1\}^m$. The GCgen_f procedure outputs (1) a vector of input wire keys $\mathsf{ks} = \{k^{w,b}\}_{w \in W, b \in \{0,1\}}$ where W is the set of the n input wires, and (2) a vector gc of $4|C|$ covert symmetric encryption ciphertexts, where $|C|$ is the number of gates in a Boolean circuit for f. If $S = \{(m^{w,0}, m^{w,1})\}_{w \in [n], b \in \{0,1\}}$ and $x \in \{0,1\}^n$ then let $S[:x]$ denote $\{m^{w,x_w}\}_{w \in [n]}$, i.e. a *selection* of bistrings from the n pairs in S according to the n bits of x. Let $m' = 4|C|\tau + n\tau$. The notion of garbled circuit covertness which [16] define, and which their Yao's garbling variant satisfies, is that for any function f, any distribution D over its inputs, and any efficient algorithm $\mathcal{A}$, there is an efficient algorithm $\mathcal{A}^*$ s.t. $|\mathsf{Adv}_{\mathcal{A}} - \mathsf{Adv}_{\mathcal{A}^*}|$ is negligible, where:

$$\mathsf{Adv}_{\mathcal{A}} = |\Pr[1 \leftarrow A(\{\mathsf{gc}, \mathsf{ks}[:x]\})]_{x \leftarrow D, (\mathsf{gc},\mathsf{ks}) \leftarrow \mathsf{GCgen}_f} - \Pr[1 \leftarrow \mathcal{A}(r)]_{r \leftarrow \{0,1\}^{m'}}|$$
$$\mathsf{Adv}_{\mathcal{A}^*} = |Pr[1 \leftarrow \mathcal{A}^*(f(x))]_{x \leftarrow D} - \Pr[1 \leftarrow \mathcal{A}^*(r)]_{r \leftarrow \{0,1\}^m}|$$

In other words, for any function f and distribution D over its inputs, the garbled circuit for f together with the wire keys corresponding to x sampled from D are (in)distinguishable from a random string *to the same degree* as function outputs $f(x)$ for $x \leftarrow D$. In particular, if f, D are such that $\{f(x)\}_{x \leftarrow D}$ is indistinguishable from random, then so is $\{\mathsf{gc}, \mathsf{ks}[:x]\}_{x \leftarrow D, (\mathsf{gc},\mathsf{ks}) \leftarrow \mathsf{GCgen}_f}$.

5 Covert Simulation-Sound CKEM

A *Conditional Key Encapsulation Mechanism* (CKEM) [10] is a generalization of SPHF to interactive protocols. A CKEM scheme for language L is a protocol between two parties, a sender S and a receiver R, on S's input a statement x_S and R's input a (statement,witness) pair (x_R, w_R). The outputs of S and R are respectively K_S and K_R s.t. K_S is a random string of τ bits, and $\mathsf{K}_R = \mathsf{K}_S$ if and only if $x_S = x_R$ and $(x_R, w_R) \in \mathcal{R}[\mathsf{L}]$. A CKEM scheme is an encryption counterpart of a zero-knowledge proof, where rather than having the prover/receiver use its witness w_R to prove to the verifier/sender that $x_S \in \mathsf{L}$, here the prover establishes a session key K with the verifier if and only if w_R is a witness for x_S in L. Because of this relation to zero-knowledge proofs, we will follow [2] and use proof-system terminology to define security properties of a CKEM scheme. In particular, we will refer to the CKEM security property that if $x \notin \mathsf{L}$ then no efficient algorithm can compute K output by $\mathsf{S}(x)$ as the *soundness* property.

A notion of CKEM was extended by Benhamouda et al. [2], who called it *Implicit Zero-Knowledge*, to a *trapdoor CKEM*. Namely, consider a CRS generation procedure which together with parameters π generates a trapdoor td that allows an efficient simulator algorithm to compute the session key K_S output by a sender $\mathsf{S}(x)$ for any x, including $x \notin \mathsf{L}$. The ability to construct such simulator makes CKEM into a more versatile protocol building block. For example, trapdoor CKEM implies a zero-knowledge proof for the same language, if R simply

returns the key K_R to S who accepts iff $\mathsf{K}_R = \mathsf{K}_S$. Indeed, following [2], we refer to the property that the simulator computes the same key as the honest receiver would in the case $x \in \mathsf{L}$ as the *zero-knowledge* of CKEM.

As in the case of zero-knowledge proofs, if multiple parties perform CKEM instances concurrently then we need to strengthen CKEM security properties to *simulation-soundness*, so that all instances executed by the corrupt players remain sound even in the presence of a simulator $\mathcal{S}$ who uses its trapdoor to simulate the instances performed on behalf of the honest players. Simulation-soundness is closely related to non-malleability: If $\mathcal{S}$ simulates a CKEM instance π on $x \notin \mathsf{L}$ then an efficient adversary must be unable to use protocol instance π executed by $\mathcal{S}$ to successfully complete another instance π' of CKEM executed by a corrupt party for any $x' \notin \mathsf{L}$.

We define the above security properties formally, building on the definitions given by Benhamouda et al. [2], and strengthening them to assure CKEM *covertness*. Namely, for *covert zero-knowledge* we require not only that an interaction with an honest receiver is indistinguishable from the interaction with a simulator but also that both the receiver or the simulator are indistinguishable from a random beacon. Similarly, for *covert (simulation) soundness* we require not only that the sender's session key K is indistinguishable from a random string if the protocol proceeds on $x \notin \mathsf{L}$ (even in the presence of a multiple simulated protocol instances), but that all messages the sender sends *and* the session key it outputs, are together indistinguishable from a random beacon.

Since in the CKEM applications we need both protocol parties to be covert, we add a requirement to covert zero-knowledge that an interaction with the sender is also indistinguishable from an interaction with a random beacon. However, we note that since we want protocol parties to be covert on all statements x, we can only require indistinguishability from a random beacon of the protocol interaction, i.e. excluding the key (or rejection) which forms a local output of the interacting party. (Otherwise an adversary running the counterparty's protocol on the same x and/or its witness w, can distinguish this party from a random beacon by computing the same output key.)

To distinguish between different CKEM sessions the CKEM syntax must also be amended by *labels*, which play similar role as labels in CCA encryption. Formally, a CKEM scheme for language family L is a tuple of algorithms $(\mathsf{PG}, \mathsf{TPG}, \mathsf{Snd}, \mathsf{Rec}, \mathsf{TRec})$ s.t. parameter generation $\mathsf{PG}(1^\tau)$ generates CRS parameter π, trapdoor parameter generation $\mathsf{TPG}(1^\tau)$ generates π together with the simulation trapdoor td, and sender Snd, receiver Rec, and trapdoor receiver TRec are interactive algorithms which run on local inputs respectively (π, x, ℓ), (π, x, ℓ, w), and $(\pi, x, \ell, \mathsf{td})$, and each of them outputs a session key K as its local output. CKEM correctness requires that for all ℓ:

$$\forall (x, w) \in \mathcal{R}[\mathsf{L}],\ [\mathsf{K}_S, \mathsf{K}_R] \leftarrow [\mathsf{Snd}(\pi, x, \ell), \mathsf{Rec}(\pi, x, \ell, w)] \Rightarrow \mathsf{K}_S = \mathsf{K}_R \quad (1)$$

$$\forall x,\ [\mathsf{K}_S, \mathsf{K}_R] \leftarrow [\mathsf{Snd}(\pi, x, \ell), \mathsf{TRec}(\pi, x, \ell, \mathsf{td})] \Rightarrow \mathsf{K}_S = \mathsf{K}_R \quad (2)$$

where (1) holds for all π generated by $\mathsf{PG}(1^\tau)$ and (2) for all (π, td) generated by $\mathsf{TPG}(1^\tau)$. Crucially, property (2) holds for all x, and not just for x in L.

Covert Zero-Knowledge. We say that a CKEM for language (family) L is *covert zero-knowledge* if the following properties hold:

1. *Setup Indistinguishability:* Parameters π generated by $\mathsf{PG}(1^\tau)$ and $\mathsf{TPG}(1^\tau)$ are computationaly indistinguishable.
2. *Zero Knowledge:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\mathsf{Rec}_{\&\mathsf{Out}}(\pi,x,\ell,w)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{TRec}_{\&\mathsf{Out}}(\pi,x,\ell,\mathsf{td})}(\mathsf{st})\}$$

   for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and $(\mathsf{st}, x, w, \ell) \leftarrow \mathcal{A}_1(\pi, \mathsf{td})$ s.t. $(x, w) \in \mathcal{R}[\mathsf{L}]$.[3]
3. *Receiver Covertness:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\mathsf{Rec}(\pi,x,\ell,w)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{Rec}^{\$(\tau)}}(\mathsf{st})\}$$

   for $\pi \leftarrow \mathsf{PG}(1^\tau)$ and $(\mathsf{st}, x, w, \ell) \leftarrow \mathcal{A}_1(\pi, \mathsf{td})$ s.t. $(x, w) \in \mathcal{R}[\mathsf{L}]$.
4. *Trapdoor-Receiver Covertness:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\mathsf{TRec}(\pi,x,\ell,\mathsf{td})}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{TRec}^{\$(\tau)}}(\mathsf{st})\}$$

   for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and $(\mathsf{st}, x, \ell) \leftarrow \mathcal{A}_1(\pi, \mathsf{td})$.
5. *Sender Covertness:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\mathsf{Snd}(\pi,x,\ell)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{Snd}^{\$(\tau)}}(\mathsf{st})\}$$

   for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and $(\mathsf{st}, x, \ell) \leftarrow \mathcal{A}_1(\pi, \mathsf{td})$.

Discussion: Zero-knowledge means that an interaction with Rec on any $x \in \mathsf{L}$ followed by seeing Rec's local output K_R, can be simulated by TRec without knowledge of the witness for x. Receiver and Trapdoor-Receiver covertness mean that in addition, the adversary $\mathcal{A}$ who interacts with resp. Rec and TRec but does not see their local outputs cannot tell them from random beacons. In the case of TRec we ask for this to hold for any x and not only for $x \in \mathsf{L}$ because a simulator of a higher-level protocol will typically create incorrect statements and then it will need to simulate the Receiver algorithm on them. Note that we cannot include the output K_R of either Rec or TRec in $\mathcal{A}$'s view in the covertness game because $\mathcal{A}$ can compute it by running $\mathsf{Snd}(x)$. Sender covertness means that an intercation with the Snd is indistinguishabile from an interaction with a random beacon for any x. Here too we cannot include Snd's local output K_S in $\mathcal{A}$'s view because if $(x, w) \in \mathcal{R}[\mathsf{L}]$ then $\mathcal{A}$ who holds w can compute it running $\mathsf{Rec}(x, w)$. Note that $\mathcal{A}$'s view includes both the public parameters π and the simulator's trapdoor td, which implies that all the properties will hold in the presence of multiple CKEM instances simulated by TRec using td.

Covert Soundness and Simulation-Soundness. A CKEM is covert sound if interaction with Snd on $x \notin \mathsf{L}$ followed by Snd's local output K_S is indistinguishable from interaction with a random beacon. Recall that soundness requires

[3] If $\mathcal{A}_1$ outputs $(x, w) \notin \mathcal{R}[\mathsf{L}]$ we override $\mathcal{A}_2$'s output by an arbitrary constant.

pseudorandomness of only Snd's output K_S on $x \notin \mathsf{L}$ while here we require it also of all network messages Snd sends. Covert simulation-soundness requires that this holds even if the adversary has access to the Trapdoor-Receiver for any (x', ℓ') which differs from the pair (x, ℓ) that defines the soundness challenge. To that end we use notation $P_{\mathsf{Block}(x)}$ for a wrapper over (interactive) algorithm P which outputs $\perp$ on input $x' = x$ and runs $P(x')$ for $x' \neq x$:

CKEM is *Covert Sound* if for every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\mathsf{Snd}_{\&\mathsf{Out}}(\pi,x,\ell)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{Snd}^{\$(\tau)}_{\&\mathsf{Out}}}(\mathsf{st})\}$$

for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and $(\mathsf{st}, x, \ell) \leftarrow \mathcal{A}_1(\pi)$ s.t. $x \notin \mathsf{L}$.

CKEM is *Covert Simulation-Sound* if for every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\mathsf{Snd}_{\&\mathsf{Out}}(\pi,x,\ell),\mathsf{TRec}_{\mathsf{Block}(x,\ell)}(\mathsf{td},\cdot)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{Snd}^{\$(\tau)}_{\&\mathsf{Out}},\mathsf{TRec}_{\mathsf{Block}(x,\ell)}(\mathsf{td},\cdot)}(\mathsf{st})\}$$

for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and $(\mathsf{st}, x, \ell) \leftarrow \mathcal{A}_1^{\mathsf{TRec}(\mathsf{td},\cdot)}(\pi)$ s.t. $x \notin \mathsf{L}$ and $\mathsf{TRec}(\mathsf{td}, \cdot)$ was not queried on (x, ℓ).

Covert ZK Relaxation. The standard-model CKEM systems we show in Section 6.2 maintain a version of covertness which is relaxed in two ways. First, receiver and trapdoor-receiver covertness hold only for language statements parameterized by a parameter randomly generated by a trusted third party. (For all languages listed in Section 6.1 these parameters consist of a public key of the Cramer-Shoup encryption scheme.) If the simulator of the higher-level protocol that utilizes this CKEM system uses the trapdoor embedded in these parameters, i.e. the decryption key corresponding to this Cramer-Shoup public key, we must assure covertness in the presence of such oracle. Looking ahead, the simulator in the covert 2PC protocol of Section 7 uses the Cramer-Shoup decryption key to decrypt the ciphertexts sent by the corrupt parties, and we need the CKEM where the honest party A is a receiver to be receiver-covert even in the presence of a simulator which uses the trapdoor to decrypt the ciphertexts sent by the corrupt party B, and vice versa. If the higher-level protocol separates the labels used in A's ciphertexts and CKEM instances where A acts as a prover/receiver from the corresponding labels used by B, then the relaxation of covert zero-knowledge given below will suffice.

Let $\mathsf{TPG}', \mathsf{O}$ be a pair of algorithms s.t. $\mathsf{TPG}'(1^\tau)$ generates (π', td') and $\mathsf{O}(\mathsf{td}', \cdot)$ is an oracle. Let $\theta(x)$ be a predicate, and let $\mathsf{O}_{\mathsf{Block}(\theta(x,\cdot))}(\mathsf{td}', \cdot)$ be a wrapper which on input x' outputs $\perp$ if $\theta(x, x') = 1$ and otherwise outputs $\mathsf{O}(\mathsf{td}', x')$. Consider language L parametrized by π'. We say that CKEM for L is *Covert Zero-Knowledge Relative To* $(\mathsf{PG}', \mathsf{O}, \theta)$ if it satisfies *Setup Indistinguishability*, *Zero-Knowledge*, and *Sender Covertness* for $\mathsf{L}(\pi')$ for any π' output by $\mathsf{TPG}'(1^\tau)$, and it satisfies:

3'. *Receiver Covertness Relative to* $(\mathsf{TPG}', \mathsf{O}, \theta)$*:* For every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\mathsf{Rec}(\pi,x,\ell,w),\mathsf{O}_{\mathsf{Block}(\theta(x,\cdot))}(\mathsf{td}',\cdot)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{Rec}^{\$(\tau)},\mathsf{O}_{\mathsf{Block}(\theta(x,\cdot))}(\mathsf{td}',\cdot)}(\mathsf{st})\}$$

for $(\pi, \mathsf{td}) \leftarrow \mathsf{PG}(1^\tau)$, $(\pi', \mathsf{td}') \leftarrow \mathsf{PG}'(1^\tau)$, $(\mathsf{st}, x, w, \ell) \leftarrow \mathcal{A}_1^{\mathsf{O}(\mathsf{td}', \cdot)}(\pi, \mathsf{td}, \pi')$ s.t. $(x, w) \in \mathcal{R}[\mathsf{L}(\pi')]$.

4'. *Trapdoor-Receiver Covertness Relative to* $(\mathsf{TPG}', \mathsf{O}, \theta)$*:* For every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\mathsf{TRec}(\pi, x, \ell, \mathsf{td}), \mathsf{O}_{\mathsf{Block}(\theta(x, \cdot))}(\mathsf{td}', \cdot)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{TRec}^{\$(\tau)}, \mathsf{O}_{\mathsf{Block}(\theta(x, \cdot))}(\mathsf{td}', \cdot)}(\mathsf{st})\}$$

for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$, $(\pi', \mathsf{td}') \leftarrow \mathsf{PG}'(1^\tau)$, $(\mathsf{st}, x, \ell) \leftarrow \mathcal{A}_1^{\mathsf{O}(\mathsf{td}', \cdot)}(\pi, \mathsf{td}, \pi')$.

Secondly, the standard-model CKEM systems we show in Section 6.2 do not assure sender covertness against an adversary who is given the simulation trapdoor td. However, sender covertness does hold for adversary who gets parameters π generated by TPG and has access to the Trapdoor-Receiver simulation oracle for any (x', ℓ') which differs from (x, ℓ) that defines the sender covertness challenge. We say that CKEM for L is *Sender Simulation-Covert* if it satisfies:

5'. *Sender Simulation-Covertness:* For every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\mathsf{Snd}(\pi, x, \ell), \mathsf{TRec}_{\mathsf{Block}(x, \ell)}(\mathsf{td}, \cdot)}(\mathsf{st})\} \approx \{\mathcal{A}_2^{\mathsf{Snd}^{\$(\tau)}, \mathsf{TRec}_{\mathsf{Block}(x, \ell)}(\mathsf{td}, \cdot)}(\mathsf{st})\}$$

for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and $(\mathsf{st}, x, \ell) \leftarrow \mathcal{A}_1^{\mathsf{TRec}(\mathsf{td}, \cdot)}(\pi)$ s.t. $\mathsf{TRec}(\mathsf{td}, \cdot)$ was not queried on (x, ℓ).

We say that a CKEM for L is *Covert Zero-Knowledge (relative to* $(\mathsf{PG}', \mathsf{O})$*) with Sender Simulation-Covertness* if CKEM satisfies all the properties required by Covert Zero-Knowledge (relative to $(\mathsf{PG}', \mathsf{O})$), except with the *Sender Covertness* property (5) replaced by the *Sender Simulation-Covertness* property (5').

Notice that sender simulation-covertness together with standard, i.e. non-covert, simulation-soundness, imply covert simulation-soundness of a CKEM.

Lemma 1. *If a CKEM scheme for some language is simulation sound and sender simulation-covert, then it is also covert simulation-sound.*

Proof. Consider the simulation-soundness game where adversary $\mathcal{A}$ on input π for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ interacts with $\mathsf{TRec}(\mathsf{td}, \cdot)$, generates (x, ℓ) s.t. $x \notin \mathsf{L}$, and interacts with oracles $\mathsf{Snd}_{\&\mathsf{Out}}(\pi, x, \ell)$ and $\mathsf{TRec}_{\mathsf{Block}(x, \ell)}(\mathsf{td}, \cdot)$. The standard (i.e. non-covert) simulation soundness of this CKEM implies that this game is indistinguishable from a modification in which key K_S output by $\mathsf{Snd}_{\&\mathsf{Out}}(\pi, x, \ell)$ is chosen at random. Once K_S is independently random, sender simulation-covertness, which hods for all x, implies that this game is indstinguishable from a modifcation where the *messages* sent by Snd are replaced with a random beacon. Since these two moves replace oracle $\mathsf{Snd}_{\&\mathsf{Out}}(\pi, x, \ell)$ with $\mathsf{Snd}_{\&\mathsf{Out}}^{\$(\tau)}$, it follows that the CKEM is covert simulation-sound.

6 Covert CKEM's for *Linear Map Image* Languages

The Covert 2PC protocol in Section 7 relies on covert CKEM's for what we call *Linear Map Image* languages. A linear map image language $\mathsf{LMI}_{n,m}$ for group G of prime order p is defined as the language of pairs $(C, M) \in G^n \times G^{n\times m}$ s.t. there exists a vector $w \in \mathbb{Z}_p^m$ s.t. $C = w \cdot M$, where the vector dot product denotes component-wise exponentiation, e.g. $[w_1, ..., w_m] \cdot [g_{i1}, ..., g_{im}] = \prod_{j=1}^{m} (g_{ij})^{w_j}$. In other words, $(C, M) \in \mathsf{LMI}_{n,m}$ if C is in the image of a linear map $f_M : \mathbb{Z}_p^m \to G^n$ defined as $f_M(w) = w \cdot M$, i.e. if C is in the subspace of G^n spanned by the rows of M, which we denote $\mathsf{span}(M)$. We extend this notion to a *class* of Linear Map Image languages, denoted LMI, consisting of all languages L for which there are some integers n, m and a pair of efficiently computable functions $\phi : U_x \to (G \times G^{n\times m})$ and $\gamma : U_w \to \mathbb{Z}_p^m$ where U_x, U_w are the implicit universes respectively of statements in L and of witnesses for these statements, s.t. for all $(x, w) \in U_x \times U_w$, w is a witness for $x \in \mathsf{L}$ if and only if $\gamma(w)$ is a witness for $\phi(x) \in \mathsf{LMI}_{n,m}$. We will sometimes abuse notation by treating set $\{\phi(x)\}_{x\in\mathsf{L}}$, i.e. L mapped onto (a subset of) $\mathsf{LMI}_{n,m}$, replaceably with L itself.

It is well known that Linear Map Image languages have efficient SPHF's: KG picks the hashing key $\mathsf{hk} = (\mathsf{hk}_1, ..., \mathsf{hk}_n)$ at random in $\mathbb{Z}_p^n$, and computes the key projection $\mathsf{hp} = (\mathsf{hp}_1, ..., \mathsf{hp}_m)$ as $\mathsf{hp} = M \cdot \mathsf{hk}$, i.e. $\mathsf{hp}_i = \prod_{j=1}^{n} (g_{ij})^{\mathsf{hk}_j}$. $\mathsf{Hash}((C, M), \mathsf{hk})$ outputs $\mathsf{H} = C \cdot \mathsf{hk} = \prod_{j=1}^{n} (C_j)^{\mathsf{hk}_j}$, while $\mathsf{PHash}((C, M), w, \mathsf{hp})$ outputs $\mathsf{projH} = w \cdot \mathsf{hp} = \prod_{i=1}^{m} (\mathsf{hp}_i)^{w_i}$. Correctness holds because if $C = w \cdot M$ then $\mathsf{H} = C \cdot \mathsf{hk} = (w \cdot M) \cdot \mathsf{hk} = w \cdot (M \cdot \mathsf{hk}) = w \cdot \mathsf{hp} = \mathsf{projH}$, and smoothness holds because if $C \notin \mathsf{span}(M)$ then $\mathsf{H} = C \cdot \mathsf{hk}$ is independent from $\mathsf{hp} = M \cdot \mathsf{hk}$.

We will show two types of covert CKEM's for Linear Map Image languages. In Section 6.2 we show that the zero-knowledge and simulation-sound CKEM proposed by Benhamouda et al. [2] for any language L in LMI, satisfies sender simulation-covertness and covert simulation soundness if M is full row rank for all x in the implicit universe U_x of statements in L, which is the case for all LMI languages used in our covert 2PC protocol – we list these languages in Section 6.1. We also show that this CKEM is receiver and trapdoor-receiver covert *relative to the Cramer-Shoup decryption oracle* for all these languages. Secondly, in Section 6.3 we show that covert CKEM proposed proposed in [11] for any language with a sigma-protocol, i.e. 3-round public coin honest-verifier ZK proof system with some additional properties, is also covert zero-knowledge and simulation-sound in the Random Oracle Model (ROM). Since every Linear Map Image language has such sigma-protocol, this implies a covert zero-knowledge and simulation-sound CKEM for all such languages in ROM.

Note on CKEM Efficiency and Generality. Both CKEM systems are two-round protocols, but the standard model CKEM of Section 6.2 uses at least twice more computation and bandwidth than the random oracle model CKEM of Section 6.3. For language L in $\mathsf{LMI}_{n,m}$, the first CKEM takes $2m{+}14$ multi-exponentiations with (up to) $2n{+}10$ bases for S, $2n{+}12$ multi-exp's with (up to) $2m{+}12$ bases for R, and its bandwidth is (a covert encoding of) $2n{+}2m{+}22$ group elements, while the second CKEM takes n multi-exp's in (up to) $m{+}1$

bases for both S and R, and its bandwidth is (a covert encoding of) m elements in $\mathbb{Z}_p$ and 3 group elements. Also, we show the standard model CKEM of Section 6.2 as covert ZK and SS (relative to the Cramer-Shoup decryption oracle, with sender simulation covertness) only for Linear Map Image languages listed in Section 6.1, while the CKEM of Section 6.3 is covert ZK and SS for any Linear Map Image language, albeit only in ROM.

6.1 Linear Map Image Languages for Covert 2PC

We list the LMI languages used in the covert 2PC protocol of Section 7 for which we need covert zero-knowledge and simulation-sound CKEM's. We defer to Appendix B for the full explanation why each of these languages is in class LMI, and for the specifications of how language instances are mapped to (C, M) instances of $\mathsf{LMI}_{n,m}$ for some n, m. Let $(\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ be the CCA-covert Cramer-Shoup PKE. All languages below are implicitely parametrized by the public key pk output by $\mathsf{Kg}(1^\tau)$, which also specifies the prime-order group setting (g, G, p). (Formally, pk is part of a statement in each of the languages below.)

Language Le of correct (ciphertext,label,plaintext) tuples for plaintext $\mathsf{m} \in G$:

$$\mathsf{Le}(\mathsf{pk}) = \{(\mathsf{ct}, \mathsf{m}, \ell) \text{ s.t. } \mathsf{ct} \in \mathsf{E}_{\mathsf{pk}}(\mathsf{m}, \ell)\}$$

(see Appendix B for a 1-to-1 mapping from Le to $\mathsf{LMI}_{4,1}$.)

Language Lbit of "shifted" encryptions of a bit:

$$\mathsf{Lbit}(\mathsf{pk}) = \{(\mathsf{ct}, \ell) \text{ s.t. } \exists b\ (\mathsf{ct}, g^b, \ell) \in \mathsf{Le}(\mathsf{pk}) \wedge b \in \{0, 1\}\}$$

(see Appendix B for a 1-to-1 mapping from $\mathsf{Lbit}(\mathsf{pk})$ to $\mathsf{LMI}_{6,3}$.)

Language Ldis is used for proving that a key corresponding to some *sender's wire* in Yao's garbled circuit is consistent with the two key values the sender committed in $\mathsf{ck}_0, \mathsf{ck}_1$ and with the bit the sender committed in ct. To cast this language as a (simple) LMI language we use the "shifted" version of Cramer-Shoup encryption in these statements, i.e. we encrypt $g^{\mathsf{m}} \in G$ instead of $\mathsf{m} \in \mathbb{Z}_p$. In other words, $\mathsf{Ldis}(\mathsf{pk})$ consists of tuples $(\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1)$ s.t. either (ct encrypts g^0 and ck_0 encrypts g^{m}) or (ct encrypts g^1 and ck_1 encrypts g^{m}):

$$\mathsf{Ldis}(\mathsf{pk}) = \{(\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell, \ell_0, \ell_1) \text{ s.t. } \exists b\ (\mathsf{ct}, g^b, \ell), (\mathsf{ck}_b, g^{\mathsf{m}}, \ell_b) \in \mathsf{Le}(\mathsf{pk})\}$$

(see Appendix B for a 1-to-1 mapping from Ldis to $\mathsf{LMI}_{19,11}$.)

Language Ldis' is a simplification of Ldis which omits checking the constraint imposed by ciphertext ct.

Language Lotr is used for proving correctness of a response in an Oblivious Transfer of Aiello et al. [1], formed using procedure OTrsp (see Section 4), which the sender uses in Yao's protocol to send keys corresponding to *receiver's wires*:

$$\begin{aligned}\mathsf{Lotr}(\mathsf{pk}) \;=\; \{\; &(\mathsf{otr}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell_0, \ell_1) \quad \text{s.t.} \quad \exists k_0, k_1, \mathsf{r} \\ &(\mathsf{ck}_0, k_0, \ell_0) \in \mathsf{Le}(\mathsf{pk}) \wedge (\mathsf{ck}_1, k_1, \ell_1) \in \mathsf{Le}(\mathsf{pk}) \wedge \mathsf{otr} = \mathsf{OTrsp}_{\mathsf{pk}}(\mathsf{ct}, k_0, k_1; \mathsf{r})\;\}\end{aligned}$$

(see Appendix B for a 1-to-1 mapping from Lotr to $\mathsf{LMI}_{10,6}$.)

Finally, observe that LMI is closed under conjunctions: (C_1, M_1) is in LMI_{n_1,m_1} and (C_2, M_2) is in LMI_{n_2,m_2} if and only if (C, M) is in $\mathsf{LMI}_{n_1+n_2,m_1+m_2}$ for $C = (C_1, C_2)$ and M formed from by placing M_1 in the upper-left corner, M_2 in the lower-right corner, and all-one matrices in the remaining quadrants.

6.2 Covert CKEM for a subclass of LMI Languages

Benhamouda et al. [2] showed a simulation-sound and zero-knowledge CKEM for any linear map image language L. We will argue that this CKEM is also covert for all the linear map image languages listed in Section 6.1.

We start with a slightly simplified version of the CKEM construction of [2] where points u'', e'' are globally set in π rather than determined per each (x, ℓ) pair. Let DH be the set of Diffie-Hellman tuples in G and let $\mathsf{DH}(g', h')$ be the set of (u', e') pairs s.t. $(g', h', u', e') \in \mathsf{DH}$. The (simplified) parameter generation $\mathsf{PG}(1^\tau)$ defines a prime-order group (g, G, p), samples $\pi = (g', h', u', e')$ uniformly in G^4, and (u'', e'') uniformly in $\mathsf{DH}(g'', h'')$, while the trapdoor generation $\mathsf{TPG}(1^\tau)$ samples (g', h') in G^2 and both (u', e') and (u'', e'') in $\mathsf{DH}(g', h')$, setting trapdoor td s.t. $(g', h')^{\mathsf{td}} = (u', e')$. Given statement $(C, M) \in G^n \times G^{n\times m}$, the construction expands matrix M in $G^{n\times m}$ into matrix M' in $G^{(n+5)\times(m+6)}$ and then "doubles" M' into $\overline{M}$ in $G^{(2n+10)\times(2m+12)}$, as shown in equation (3).

The point of this expansion is to modify the original LMI language of pairs (C, M) s.t. $C \in \mathsf{span}(M)$ into the language of triples (C, M, π) s.t. *either* $M \in \mathsf{span}(C)$ *or* elements of π have some (trapdoor) property. Moreover, matrix M' shows that this modification can be cast as another instance of LMI: Consider $C' = (g'^{-1}, 1, ..., 1) \in G^{n+5}$ and observe that $C' \in \mathsf{span}(M')$ if and only if

$$\text{(c1)}\ \ C \in \mathsf{span}(M) \quad \text{or} \quad \text{(c2)}\ \ (u', e') \in \mathsf{DH}(g', h') \quad \text{or} \quad \text{(c3)}\ \ (u'', e'') \notin \mathsf{DH}(g', h')$$

Note that if condition (c1) fails then rows $r_1, ..., r_m, e_1$ cannot be used in the linear combination λ' s.t. $C' = \lambda' \cdot M'$; If condition (c2) fails then rows e_2, e_3 must drop out too; And if condition (c3) fails then rows e_4, e_5, e_6 must also drop out. These additional conditions (c2) and (c3) allow PG to set π so that to guarantee soundness, while TPG can set π to inject the trapdoors needed to argue zero-knowledge and simulation-soundness, as we explain below.

$$\mathbf{M}' = \left[\begin{array}{c|cc|c|cc}
\multicolumn{3}{c|}{\mathbf{1}} & \multicolumn{1}{c|}{\mathbf{M}} & \multicolumn{2}{c}{\mathbf{1}} & (r_1), ..., (r_m) \\
\hline
g' & 1 & 1 & \mathbf{C} & 1 & 1 & (e_1) \\
\hline
1 & g' & h' & 1 \cdots 1 & 1 & 1 & (e_2) \\
\hline
g' & u' & e' & 1 \cdots 1 & 1 & 1 & (e_3) \\
\hline
1 & 1 & 1 & 1 \cdots 1 & g' & h' & (e_4) \\
\hline
1 & 1 & 1 & 1 \cdots 1 & u'' & e'' & (e_5) \\
\hline
g' & 1 & 1 & 1 \cdots 1 & g' & 1 & (e_6)
\end{array}\right] \qquad \overline{\mathbf{M}} = \left[\begin{array}{c|c} \mathbf{M}'' & \mathbf{1} \\ \hline \mathbf{1} & \mathbf{M}'' \end{array}\right] \qquad (3)$$

Note that if $C = w \cdot M$ then $C' = \lambda_w \cdot M'$ for $\lambda_w = (w, -1, 0, 0, 0, 0, 0)$ and if $(u', e') \in \mathsf{DH}(g', h')$ and $\mathsf{td} = \mathsf{DL}(g', u')$ then $C' = \lambda_{\mathsf{td}} \cdot M'$ for $\lambda_{\mathsf{td}} = (0, ..., 0, \mathsf{td}, -1, 0, 0, 0)$. Figure 2 shows the CKEM of [2] which specifies protocol Rec for R and protocol Snd for S, on respective inputs $\overline{M}$, C', λ_w derived as above. The TRec protocol is exactly like Rec except it uses λ_{td} instead of λ_w.

On inputs π and $(C, M) = \phi(x)$, and on R's input λ_w derived from w s.t. $C = w \cdot M$:

R: Pick $\mathsf{tk} \leftarrow \mathbb{Z}_p^{2m+12}$ and send $\mathsf{tp} = (\overline{M}^T) \cdot \mathsf{tk}$ to S;

S: Pick $\mathsf{hk} \leftarrow \mathbb{Z}_p^{2n+10}$ and $\psi \leftarrow \mathbb{Z}_p$, set $\overline{C} = (C', C' \cdot \psi)$, $\mathsf{hp} = \overline{M} \cdot \mathsf{hk}$, $\mathsf{H} = \overline{C} \cdot \mathsf{hk}$, and $\mathsf{tprojH} = \mathsf{hk}^T \cdot \mathsf{tp}$. Send (ψ, hp) to R and output $\mathsf{K}_S = \mathsf{H} \cdot \mathsf{tprojH}$;

R: Set $\bar{\lambda}_w = (\lambda_w, \lambda_w \cdot \psi)$, $\mathsf{projH} = \bar{\lambda}_w \cdot \mathsf{hp}$, and $\mathsf{tH} = \mathsf{hp}^T \cdot \mathsf{tk}$. Output $\mathsf{K}_R = \mathsf{projH} \cdot \mathsf{tH}$.

Fig. 2. CKEM for Linear Map Image Language due to Benhamouda et al. [2]

Zero-Knowledge and Simulation-Soundness. We recall the Zero-Knowledge and Simulation-Soundness arguments for this CKEM given by [2], since the covertness arguments we supply next will rely on them.

Correctness: The CKEM of Figure 2 is correct because it composes two SPHF's for two LMI languages: In the first SPHF, S generates $(\mathsf{hk}, \mathsf{hp})$ to verify that R holds λ s.t. $\overline{C} = \bar{\lambda} \cdot \overline{M}$. In the second SPHF, R generates $(\mathsf{tk}, \mathsf{tp})$ to verify whether S forms hp correctly, i.e. if $\mathsf{hp} = \overline{M} \cdot \mathsf{hk} = \mathsf{hk}^T \cdot \overline{M}^T$ for some hk.

Zero-Knowledge: The key observation is that if $C = w \cdot M$ and $(u', e') \in \mathsf{DH}(g', h')$ (as set by TPG) then $\overline{C}$ is equal to both $\bar{\lambda}_w \cdot \overline{M}$ and $\bar{\lambda}_{\mathsf{td}} \cdot \overline{M}$. Therefore if $\mathsf{hp} = \overline{M} \cdot \mathsf{hk}$ for some hk then $\bar{\lambda}_w \cdot \mathsf{hp} = \bar{\lambda}_{\mathsf{td}} \cdot \mathsf{hp}$. Otherwise, i.e. if $\mathsf{hp}^T \notin \mathsf{span}(\overline{M})^T$, then $\mathsf{tH} = \mathsf{hp}^T \cdot \mathsf{tk}$ (and hence also K_R) is independent of $\mathsf{hp} = (\overline{M})^T \cdot \mathsf{tk}$ in S's view. Note that this argument essentially shows that interaction with TRec (or Rec) is (perfectly) witness-hiding, i.e. that interaction with oracles $\mathsf{TRec}(\lambda, \cdot)$ and $\mathsf{TRec}(\lambda', \cdot)$ is identical as long as $\overline{C} = \bar{\lambda} \cdot \overline{M} = \bar{\lambda}' \cdot \overline{M}$.

Soundness: If $C \notin \mathsf{span}(M)$ and $(u', e') \notin \mathsf{DH}(g', h')$ and $(u'', e'') \in \mathsf{DH}(g', h')$ (as is set by PG) then conditions (c1)-(c3) all fail and $C' \notin \mathsf{span}(M')$. The point of the "matrix doubling" technique is that $C' = \lambda \cdot M'$ if and only if $(C, C' \cdot \psi) = (\lambda, \lambda \cdot \psi) \cdot \overline{M}$ for every $\psi \neq 0$, and the key lemma of [2] shows that if $C' \notin \mathsf{span}(M')$ then for any tp there exists at most one ψ value s.t. $\overline{C} = (C', C' \cdot \psi) \in \mathsf{span}(\overline{M}, \mathsf{tp}^T)$, where $\mathsf{span}(\overline{M}, \mathsf{tp}^T)$ denotes the subspace spanned by all the rows of $\overline{M}$ *and* vector tp^T. Therefore, if $C' \notin \mathsf{span}(M')$ then with overwhelming probability $\overline{C} \notin \mathsf{span}(\overline{M}, \mathsf{tp}^T)$, in which case $\overline{C} + \mathsf{tp}^T$ (a component-wise product of $\overline{C}$ and tp^T) is not in $\mathsf{span}(\overline{M})$, hence $\mathsf{K}_S = \mathsf{H} \cdot \mathsf{tprojH} = (\overline{C} \cdot \mathsf{hk}) \cdot (\mathsf{tp}^T \cdot \mathsf{hk}) = (\overline{C} + \mathsf{tp}^T) \cdot \mathsf{hk}$ is independent from $\mathsf{hp} = \overline{M} \cdot \mathsf{hk}$.

Simulation-Soundness: Note that soundness and zero-knowledge require opposite property of CRS π: For soundness we need $(u', e') \notin \mathsf{DH}(g', h')$, because other-

wise $C' \in \mathsf{span}(M')$ even if $C \notin \mathsf{span}(M)$, while for zero-knowledge we need $(u', e') \in \mathsf{DH}(g', h')$ to let TRec compute projH even if $C \notin \mathsf{span}(M)$. This is a problem for simulation-soundness because the set-up needed for the soundness challenge (x, ℓ) is different than the set-up for TRec queries (x', ℓ'). The CKEM of Benhamouda et al. [2] handles this by defining pair (u'', e'') used in matrix M' separately *per each* (x, ℓ) *instance* using the Waters [17] function $W : \{0,1\}^{2\tau} \to G^2$ and a collision–resistant hash H onto $\{0,1\}^{2\tau}$. Namely, π contains $2\tau{+}1$ pairs $\{u_i, e_i\}_{i=0}^{2\tau}$, all random in $\mathsf{DH}(g', h')$, and pair (u'', e'') for instance (x, ℓ) is defined as $W(\mathsf{H}(x, \ell))$ where $W(m) = (u_0, e_0) \cdot \pi_{i=1}^{2\tau}(u_i, e_i)^{m_i}$. Simulation-soundness is shown under the DDH assumption by a simulator which picks (u', e') in G^2 and samples (u_i, e_i)'s with knowledge of $\mathsf{DL}(g', u_i)$ and $\mathsf{DL}(h', e_i)$ in such a way that with high-enough probability two facts hold: (1) If $(u'', e'') = W(\mathsf{H}(x, \ell))$ for the soundness challenge (x, ℓ) then $(u'', e'') \in \mathsf{DH}(g', h')$, which means that if $x \notin \mathsf{L}$ then none of the conditions (c1)-(c3) above are met, hence the same soundness argument as above implies that Snd's output K_S is independent of the adversary's view; (2) For each (x', ℓ') query to TRec, we have $(u'', e'') = W(\mathsf{H}(x', \ell')) \notin \mathsf{DH}(g', h')$, and the simulator uses the knowledge of $\mathsf{DL}(g', u'')$ and $\mathsf{DL}(h', e'')$ to compute (α, β) s.t. $(g')^{\alpha}(u'')^{\beta} = g'$ and $(h')^{\alpha}(e'')^{\beta} = 1$, hence TRec can use $\lambda'_{\mathsf{td}} = (0, ..., 0, \alpha, \beta, -1)$ to compute $\mathsf{projH} = \lambda'_{\mathsf{td}} \cdot \mathsf{hp}$ because $C' = \lambda'_{\mathsf{td}} \cdot M'$.

Sender Simulation-Covertness and Covert Simulation-Soundness. We argue that under DDH assumption the above CKEM satisfies *sender simulation covertness* and *covert simulation soundness* for every language L in class LMI s.t. if ϕ is a mapping that defines L as a class LMI language maps the implicit universe U_x of L statements to $G^n \times G^{n \times m}$ then matrix M has the full row rank m for each $x \in U_x$ and $(C, M) = \phi(x)$. (This has to hold not just for $x \in \mathsf{L}$ because sender covertness must hold regardless whether x is in L.) Note that this condition holds for all languages listed in Section 6.1.

Theorem 1. *Let* L *be a language with implicit universe* U_x*, and let* $\phi : U_x \to G^n \times G^{n \times m}$ *s.t.* $x \in \mathsf{L}$ *if and only if* $\phi(x) \in \mathsf{LMI}_{n,m}$*. CKEM protocol in Figure 2 is* sender simulation-covert *under the DDH assumption for* L *if for each* $x \in U_x$ *and* $(C, M) = \phi(x)$ *matrix* M *has full row rank* m*.*

Proof. We have to show that given parameters π, value $\mathsf{hp} = \bar{M} \cdot \mathsf{hk}$ sent by Snd is indistinguishable from a random vector in $G^{2(m+6)}$ for all $(x,$, given access to oracle $\mathsf{TRec}_{\mathsf{Block}(x,\ell)}(\mathsf{td}, \cdot)$, for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^{\tau})$. (Note that value ψ sent by Snd is already uniformly random.) Since $\overline{M}$ is a diagonal matrix containing two instances of M', the SPHF for $\overline{M}$ consists of two independent instances of an SPHF for M'. Therefore we need only consider how the SPHF acts on M', i.e. consider $\mathsf{hp} = (\mathsf{hp}_1, ..., \mathsf{hp}_{m+6}) = M' \cdot \mathsf{hk}$ for $\mathsf{hk} \leftarrow \mathbb{Z}_p^{n+5}$, and argue that it is indistinguishable from random in G^{m+6} (given access to the TRec oracle). Let $(C, M) = \phi(x)$. Since by assumption M has rank m, and row e_1 has $g{\neq}1$ in the first column, space $S_1 = \mathsf{span}(r_1, ..., r_m, e_1)$ has dimension $m{+}1$, which implies that $\mathsf{hp}_{[1,m+1]}$ is random in G^{m+1}. Consider matrix M^* made of the 2nd and

3rd column of rows e_2, e_3 and the last two columns of rows e_4, e_5, e_6, and the entries in the first column of M' in the same rows, denoted F:

$$\mathbf{F} = \begin{bmatrix} 1 \\ \hline g' \\ \hline\hline 1 \\ \hline 1 \\ \hline g' \end{bmatrix} \qquad \mathbf{M}^* = \left[\begin{array}{cc|cc} g' & h' & 1 & 1 \\ \hline u' & e' & 1 & 1 \\ \hline\hline 1 & 1 & g' & h' \\ \hline 1 & 1 & u'' & e'' \\ \hline 1 & 1 & g' & 1 \end{array}\right] \begin{array}{l} (e_2^*) \\ (e_3^*) \\ (e_4^*) \\ (e_5^*) \\ (e_6^*) \end{array} \tag{4}$$

Consider vector $\mathsf{hp}^* = M^* \cdot \mathsf{hk}^*$ for $\mathsf{hk}^* = (\mathsf{hk}_{2,3}, \mathsf{hk}_{n+4,n+5})$. Observe by inspection of matrix M' in equation (3) that (1) $\mathsf{hp}_{[1,m+1]}$ is a function only of elements $(\mathsf{hk}_1, \mathsf{hk}_{[4,..,n+3]})$ of hk, and (2) $\mathsf{hp}_{[m+2,m+6]} = F \cdot \mathsf{hk}_1 + M^* \cdot \mathsf{hk}^* = F \cdot \mathsf{hk}_1 + \mathsf{hp}^*$. By fact (1) it follows that $\mathsf{hp}_{[1,m+1]}$ is independent of hk^*, because hk^* consists of elements of hk which are disjoint from elements $(\mathsf{hk}_1, \mathsf{hk}_{[4,...,n+3]})$ used in computing $\mathsf{hp}_{[1,m+1]}$. By fact (2) it follows that if we show that $\mathsf{hp}^* = M^* \cdot \mathsf{hk}^*$ is indistinguishable from random tuple in G^5 for $\mathsf{hk}^* \leftarrow G^4$ (given access to the TRec oracle) then this will imply that $\mathsf{hp}_{[m+2,m+6]}$ is also indistinguishable from random in G^5 because hp^* acts like a one-time pad in the above equation for $\mathsf{hp}_{[m+2,m+6]}$. Together with the fact that $\mathsf{hp}_{[1,m+1]}$ is random in G^{m+1}, this will complete the argument for sender simulation covertness.

It remains to argue that for $(\pi, \mathsf{td}) \leftarrow \mathsf{TPG}(1^\tau)$ and any (x, ℓ) output by adversary $\mathcal{A}$ given π and access to oracle $\mathsf{TRec}(\mathsf{td}, \cdot)$, if M^* is defined as above by $(u'', e'') = W(\mathsf{H}(x, \ell))$, then variable $\{\mathsf{hp}^* = M^* \cdot \mathsf{hk}^*\}_{\mathsf{hk}^* \leftarrow G^4}$ is indistinguishable from a random tuple in G^5, given $\mathcal{A}$'s access to oracle $\mathsf{TRec}_{\mathsf{Block}(x,\ell)}(\mathsf{td}, \cdot)$. This argument uses similar game changes as the simulation-soundness argument of [2] for this CKEM recalled above. Similarly as in that proof, consider the following modifications of TPG and TRec: The modified TPG picks sets $(u_i, e_i) = ((g')^{\delta_i}, (h')^{\zeta_i})$ for random $(\delta_i, \zeta_i) \leftarrow \mathbb{Z}_p^2$ for each $i = 0, ..., 2\tau$, and sets $\mathsf{td} = \{\delta_i, \zeta_i\}_{i=0,..,2\tau}$. Note that for every TRec query (x', ℓ'), except for negligible probability it holds that $(u'', e'') \notin \mathsf{DH}(g', h')$ for $(u'', e'') = W(m) = (u_0, e_0) \cdot \pi_{i=1}^{2\tau}(u_i, e_i)^{m_i}$ where $m = \mathsf{H}(x', \ell')$. Therefore the modified TRec can use (δ_i, ζ_i)'s to compute (α, β) s.t. $(g')^\alpha (u'')^\beta = g'$ and $(h')^\alpha (e'')^\beta = 1$, in which case $\lambda'_{\mathsf{td}} = (0, ..., 0, \alpha, \beta, -1)$ satisfies that $C' = \lambda'_{\mathsf{td}} \cdot M'$, and by argument that $\mathsf{TRec}(\mathsf{td}, \cdot)$ oracle is witness-hiding (see the zero-knowledge argument above), we have that the modified TRec oracle acts identically as the original. Since the modified TRec does not use the original trapdoor $\mathsf{td} = \mathsf{DL}(g', u') = \mathsf{DL}(h', e')$, by reduction to DDH we can modify TPG further by picking (u', e') at random G^2 tuple instead of $\mathsf{DH}(g', h')$. After this change note that rows (e_1^*, e_2^*) of M^* are now independent (except for neglgigible probability), hence $\mathsf{hp}_{1,2}^*$, which is a function of only $\mathsf{hk}_{1,2}^*$, can be replaced by a random pair in G^2.

It remains only to argue that $\mathsf{hp}_{[3-5]}^*$ is indistinguishable from random in G^3 for random $\mathsf{hk}_{3,4}^* \leftarrow \mathbb{Z}_p^2$. Denote $(r, s) = (\mathsf{hk}_3^*, \mathsf{hk}_4^*)$, and observe that $\mathsf{hp}_{3,4,5}^* = ((g')^r (h')^s, (u'')^r (e'')^s, (g')^r)$. Let us now move back from the above modifications of $\mathsf{TPG}, \mathsf{TRec}$ to the original $\mathsf{TPG}, \mathsf{TRec}$: First we change back (u', e') distribution from G^2 to $\mathsf{DH}(g', h')$, which is indistinguishable under DDH. Then we replace the modified $\mathsf{TPG}, \mathsf{TRec}$ with the original $\mathsf{TPG}, \mathsf{TRec}$, which use $\mathsf{td} = \mathsf{DL}(g', u') =$

$\mathsf{DL}(h', e')$ instead of $\mathsf{td} = \{\delta_i, \zeta_i\}_{i=0,..,2\tau}$ for $(\delta_i, \zeta_i) = (\mathsf{DL}(g', u_i), \mathsf{DL}(h', e_i))$. This does not change the view by the witness-hiding property of $\mathsf{TRec}(\mathsf{td}, \cdot)$ used above. At this point we can modify TPG to sample e_0 at random from G and pick each e_i for $i > 0$ as $e_i = (h')^{\zeta_i}$ for $\zeta_i \leftarrow \mathbb{Z}_p$. Note that for any $m = \mathsf{H}(x, \ell)$, $e'' = e_0(h')^{\zeta}$ for $\zeta = \sum_{i=1}^{2\tau}(\zeta_i)^{m_i}$, and therefore $(e'')^s = ((h')^s)^{\zeta}(e_0)^s$. We can use this relation to replace pair $(h')^s, (e_0)^s$ used in computing $\mathsf{hp}^*_{3,4,5}$ with a random pair in G, and by reduction to DDH the resulting view is insistinguishable. Note that after this change $\mathsf{hp}^*_{3,4,5} = ((g')^r g_1\,,\ (u'')^r (g_1)^{\zeta} g_2\,,\ (g')^r$ for $g_1, g_2 \leftarrow G^2$ and $r \leftarrow \mathbb{Z}_p$, hence it is uniform in G^3, which completes the proof.

By lemma 1, theorem 1 together with standard, i.e. non-covert, simulation-soundness of this CKEM shown in [2] (and recalled above), imply that this CKEM is also covert simulation-sound for the same language class:

Corollary 1. *Let* L *be a language with implicit universe* U_x*, and let* $\phi : U_x \to G^n \times G^{n\times m}$ *s.t.* $x \in \mathsf{L}$ *if and only if* $\phi(x) \in \mathsf{LMI}_{n,m}$*. CKEM protocol in Figure 2 is* covert simulation-sound *under the DDH assumption for* L *if for each* $x \in U_x$ *and* $(C, M) = \phi(x)$ *matrix* M *has full row rank* m*.*

Receiver Covertness Relative to Cramer-Shoup Decryption. For *receiver covertness* and *trapdoor receiver covertness* we argue that each language listed in Section 6.1 satisies their relaxed versions, namely, if $(\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ is the (CCA-covert) Cramer-Shoup PKE then each of these languages is *(trapdoor) receiver covert relative to* $(\mathsf{Kg}, \mathsf{D}, \theta)$, where θ will restrict the decryption oracle from decrypting on a label corresponding to any ciphertexts contained in statement x. We show that this holds for every language L in the LMI class that satisfies a technical lemma about matrices M which define L (i.e. the matrices M output by ϕ on the implicit language universe U_x), and we exemplify that this lemma holds for $\mathsf{Le}(\mathsf{pk})$, while the corresponding arguments for the remaining languages in Section 6.1 are relegated to Appendix B.

First, note that for the CKEM of Figure 2 the Threshold Receiver Covertness implies Receiver Covertness. This is because if $(x, w) \in \mathcal{R}[\mathsf{L}]$ then $\mathsf{Rec}(\pi, x, \ell, w)$ and $\mathsf{TRec}(\pi, x, \ell, \mathsf{td})$ produce identical transcripts, because they both send out $\mathsf{tp} = \overline{M}^T \cdot \mathsf{tk}$ for $\mathsf{tk} \leftarrow \mathbb{Z}_p^{2m+12}$ for $\overline{M}$ built from M s.t. $(C, M) = \phi(x)$. Since in the Threshold Receiver Covertness game x is unconstrained and in Receiver Covertness $x \in \mathsf{L}$, it suffices to argue the first property. To that end, note that as in the sender covertness argument, since $\overline{M}$ is made of two copies of M', it suffices to argue that $\mathsf{tp} = (M')^T \cdot \mathsf{tk}$ for $\mathsf{tk} \leftarrow \mathbb{Z}_p^{m+6}$ is indistinguishable from a random tuple in G^{n+5}. Note also that we can equivalently define tp as $\mathsf{tk} \cdot (M')$ where tp, tk treated as row vectors. We do not have a general characterization of matrices M output by ϕ on the implicit universe U_x of L statements for which we can argue that $\{\mathsf{tk} \cdot (M')\}_{\mathsf{tk} \leftarrow \mathbb{Z}_p^{m+6}}$ is indistinguishable from a random tuple in G^{n+5}. Note that in the languages of Section 6.1, as is expected, the number of constraints n imposed by the language is always larger than the number of variables m in a witness. (Moreover, sender covertness of this CKEM relies on M having full row rank, which requires that $m \leq n$.) Specifically,

for all languages in Section 6.1 we have $m \leq n-3$, so the rank of M' at most $m+6 \leq n+3$, which means that function $f_{M'} : \mathbb{Z}_p^{m+6} \to G^{n+5}$ s.t. $f_{M'}(x) = x \cdot M'$ is onto a subspace of dimension at most $n+3$ in G^{n+5}, hence there exist linear relations that distinguish values $\mathsf{tp} = f_{M'}(\mathsf{tk})$ from random in G^{n+5}. However, by the way matrix M' is constructed from M we can argue a technical lemma which reduces pseudorandomness of random points in the image of $f_{M'}$ to the pseudorandomness of random points in the image of $f_M : \mathbb{Z}_p^m \to G^n$, $f_M(x) = x \cdot M$, and then we show that this condition is satisfied for each language in Section 6.1.

For the purpose of the lemma below we define the following technical property. Let $\phi : U_x \to G^n \times G^{n \times m}$ be a function which defines an LMI language L. We say that a L is *matrix-map receiver covert relative to* $(\mathsf{PG}', \mathsf{O}, \theta)$ if for every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\mathsf{O}_{\mathsf{Block}(x)}(\mathsf{td}', \cdot)}(\mathsf{st}, f_M(s)\} \approx \{\mathcal{A}_2^{\mathsf{O}_{\mathsf{Block}(x)}(\mathsf{td}', \cdot)}(\mathsf{st}, r)\}$$

for $(\pi', \mathsf{td}') \leftarrow \mathsf{PG}'(1^\tau)$, $(\mathsf{st}, x) \leftarrow \mathcal{A}_1^{\mathsf{O}_{\mathsf{Block}(\theta(x,\cdot))}(\mathsf{td}', \cdot)}(\pi')$, $(C, M) \leftarrow \phi(x)$, $s \leftarrow \mathbb{Z}_p^m$, and $r \leftarrow G^n$.

Lemma 2. *If an* LMI *language* L *is matrix-map receiver covert relative to* $(\mathsf{PG}', \mathsf{O}, \theta)$ *then the CKEM of Figure 2 is (trapdoor) receiver covert relative to* $(\mathsf{PG}', \mathsf{O}, \theta)$ *for* L.

Proof. Let M_C denote the n-by-$(m+1)$ matrix containing m rows of M and C as the $m+1$-st row. Consider adversary $\mathcal{A}$ in the trapdoor receiver covertness game interacting with oracle $\mathsf{TRec}(\pi, x, \ell, \mathsf{td})$ on (x, ℓ) of its choice, given (π, td) output by $\mathsf{PG}(1^\tau)$ and π' for (π', td') output by $\mathsf{PG}'(1^\tau)$. Throughout the execution $\mathcal{A}$ can query oracle $\mathsf{O}_{\mathsf{Block}(\theta(x,\cdot))}(\mathsf{td}', \cdot)$. Note that in this CKEM interaction with oracle $\mathsf{TRec}(\pi, x\ell, \mathsf{td})$ means simply receiving $\mathsf{tp} = (\overline{M})^T \cdot \mathsf{tk}$ for $\mathsf{tk} \leftarrow \mathbb{Z}_p^{2(m+6)}$. We need to argue that $\mathcal{A}$'s view in this game is indistinguishable from a view in a modified game where tp is chosen at random in $G^{2(n+5)}$. By construction of $\overline{M}$ this will follow if we show that $\mathsf{tp} = (M')^T \cdot \mathsf{tk}$ for $\mathsf{tk} \leftarrow \mathbb{Z}_p^{m+6}$ is indistinguishable from random in G^{n+5}. Denote the first $m+1$ elements of tk as $s = \mathsf{tk}_{[1,m+1]}$ and the last 5 as $r = (r_1, r_2, r_3, r_4, r_5) = \mathsf{tk}_{[m+2,m+5]}$. By inspection of matrix M' in equation (3) and matrix M^* in equation (3), note that $\mathsf{tp}_{4,n+3} = s \cdot M_C$, $\mathsf{tp}_{2,3,n+4,n+5} = r \cdot M^*$, and $\mathsf{tp}_1 = (g')^{s_{m+1}+r_2+r_5}$ for $(s, r) \leftarrow \mathbb{Z}_p^{m+1} \times \mathbb{Z}_p^5$. By inspection of M^* note that $\mathsf{tp}_{1,2,3,n+4,n+5}$ is equal to

$$((g')^{s_{m+1}+r_2+r_5}, (g')^{r_1}(u')^{r_2}, (h')^{r_1}(e')^{r_2}, (g')^{r_3+r_5}(u'')^{r_4}, (h')^{r_3}(e'')^{r_4})$$

Consider a modified distribution of $\mathsf{tp}_{1,2,3,n+4,n+5}$ where $(g')^{r_1}, (h')^{r_1}$ are replaced by (g_1, g_2) random in G^2. Since $\mathsf{DL}(g', h')$ is not part of td generated by PG, a reduction which embeds a DDH challenge in tuple (g', h', g_1, g_2), shows that this modification is indistinguishable from the original, assuming DDH. For the same reason, incurring another indistinguishable change in the view assuming DDH, we can also replace $(g')^{r_3}, (h')^{r_3}$ by (g_3, g_4) random in G^2. After this

change $\mathsf{tp}_{1,2,3,n+4,n+5}$ looks like

$$((g')^{s_{m+1}+r_2+r_5}, g_1(u')^{r_2}, g_2(e')^{r_2}, g_3(g')^{r_5}(u'')^{r_4}, g_4(e'')^{r_4})$$

which is random in G^5 *for every value of* s_{m+1}, over the choice of random (r_2, r_4, r_5) in $\mathbb{Z}_p^3$ and (g_1, g_2, g_3, g_4) in G^4. Therefore even though s_{m+1} participates in the equation for $\mathsf{tp}_{[4,n+3]} = s \cdot M$, we can replace $\mathsf{tp}_{1,2,3,n+4,n+5}$ with a random tuple in G^5 without changing the view. Note that $\mathsf{tp}_{[4,n+3]} = s \cdot M_C = s_{[1,m]} \cdot M + s_{m+1} \cdot C$. By the *matrix-map receiver covertness* property of this language this view is indistinguishable from the one where $s_{[1,m]} \cdot M$ is replaced with a random vector in G^n. By a one-time-pad argument it follows that $\mathsf{tp}_{[4,n+3]}$ is random in G^n, and hence tp random in G^{n+5}, which ends the proof.

Let $(\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ be the Cramer-Shoup encryption. Below we show that the conditions of the above lemma hold relative to $(\mathsf{Kg}, \mathsf{D}, \theta_{\mathsf{Le}})$ for language $\mathsf{Le}(\mathsf{pk})$ where $\theta_{\mathsf{Le}}((\mathsf{ct}, \mathsf{m}, \ell), (\mathsf{ct}', \ell'))$ returns 1 if $\ell' = \ell$, i.e. if the decryption oracle is prevented from decrypting under the same label as the label of the ciphertext in the language statement. Similar arguments for all the other languages in Section 6.1 are included in Section B.1 in Appendix B. In each case predicate $\theta_{\mathsf{L}}(x, \cdot)$ prevents the $\mathsf{D}(\mathsf{sk}, \mathrm{cot})$ oracle from decrypting any ciphertext whose labels match those contained in statement x of L. All of these arguments rely on the same variant of the CCA-covertness property of Cramer-Shoup encryption, used also in lemma 3 below, namely that assuming DDH and collision-resistance of H, ciphertext ct^* formed as $\{(g_1, g_2, h, cd^{\xi})^s\}_{s \leftarrow \mathbb{Z}_p}$ is pseudorandom for $\xi = \mathsf{H}(\ell, z)$ for any string z, given access to the decryption oracle $\mathsf{D}_{\mathsf{sk}}(\cdot, \cdot)$ which decrypts ciphertext,label pairs (ct', ℓ') as long as $\ell' \neq \ell$.

Note that by lemma 2, the fact that each of the languages L in Section 6.1 is matrix-map receiver covert relative $(\mathsf{Kg}, \mathsf{D}, \theta_{\mathsf{L}})$ for a corresponding restriction θ_{L}, as shown in lemma 3 below for $\mathsf{Le}(\mathsf{pk})$ and in lemma 4 in Section B.1 for the remaining languages, implies that the CKEM in Figure 2 is (trapdoor) receiver covert relative to the same tuple $(\mathsf{Kg}, \mathsf{D}, \theta_{\mathsf{L}})$ for each L.

Lemma 3. *Under the DDH assumption language* Le *is matrix-map receiver covert relative to* $(\mathsf{Kg}, \mathsf{D}, \theta_{\mathsf{Le}(\mathsf{pk})})$.

Proof. Let $(\mathsf{pk}, \mathsf{sk})\mathsf{Kg}(1^\tau)$ and $\mathcal{A}^{\mathsf{D}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ outputs $x = (\mathsf{ct}, \mathsf{m}, \ell)$. By inspection of M defined by ϕ for Le, see equation (5) in Appendix B, we see that $\mathsf{ct}^* = f_M(s) = s \cdot M = (g_1, g_2, h, cd^{\xi})^s$ for $s \leftarrow \mathbb{Z}_p$ and $\xi = \mathsf{H}(\ell, u_1, u_2, e)$ where $\mathsf{ct} = (u_1, u_2, e, v)$. Note that on input (ct', ℓ'), oracle $\mathsf{D}_{\mathsf{Block}(\theta_{\mathsf{Le}(\mathsf{pk})}(x, \cdot))}(\mathsf{sk}, \cdot)$ returns $\mathsf{D}(\mathsf{ct}, \ell')$ as long as $\ell' \neq \ell$. Note that ct^* is not a properly formed Cramer-Shoup ciphertext, because ξ is computed on ct, not ct^*. However, the proof of CCA covertness of Cramer-Shoup encryption can be extended to show that for *any* (ℓ, u_1, u_2, e), value $\mathsf{ct}^* = (u_1^*, u_2^*, e^*, v^*) = (g_1, g_2, h, cd^{\xi})^s$ for $\xi = \mathsf{H}(\ell, u_1, u_2, e)$ is indistinguishable from a random tuple in G^4, for s uniform in $\mathbb{Z}_p$, even given access to $\mathsf{D}_{\mathsf{Block}(\theta_{\mathsf{Le}}(x, \cdot))}(\mathsf{sk}, \cdot)$, i.e. an oracle which returns $\mathsf{D}_{\mathsf{sk}}(\mathsf{ct}', \ell')$ on input (ct', ℓ') as long as $\ell' \neq \ell$. This extension follows from two facts: First, by collision-resistance of H except for negligible probability the decryption oracle will use

value $\xi' = \mathsf{H}(\ell', u_1', u_2', e')$, where $\mathsf{ct}' = (u_1', u_2', e', v')$, which satisfies $\xi' \neq \xi$. Secondly, in the proof of CCA-covertness of the Cramer-Shoup encryption in Appendix A, the decryption oracle is shown to be "harmless" in two steps (as in the CCA-security proof for this encryption in [6]): First, we argue that if $(u_1', u_2') \in \mathsf{DH}(g_1, g_2)$ then the decryption oracle does not provide information that would be useful in distinguishing ct^* from a random tuple. Second, for the argument that a ciphertext query s.t. $(u_1', u_2') \notin \mathsf{DH}(g_1, g_2)$ is rejected except for negligible probability we referred to [6], and it is easy to see that this argument holds whenever ξ' defined by the decryption query satisfies $\xi' \neq \xi$ where ξ defines the distribution of covertness challenge ct^* as $\{(g_1, g_2, h, cd^{\xi})^s\}_{s \leftarrow \mathbb{Z}_p}$.

6.3 Covert CKEM for LMI Languages in ROM

An essential tool in a paper on efficient covert mutual authentication [11] was a compiler which used a covert commitment with associated SPHF to convert a Σ-protocol (a three-round public-coin HVZK system) with some additional properties for a given language L into a CKEM for L which satisfied the notion of covertness defined in [11]. The first thing we note is that the linear map image languages have the sigma protocols required. The CKEM covertness properties of [11] are incomparable to the ones we define here: The CKEM notion of [11] included receiver covertness and *strong* sender covertness, which is a covert counterpart of strong soundness of ZK proofs, i.e. that an efficient extractor can extract witness w from a receiver who distinguishes interacting with the sender (including the resulting key) from interacting with a random beacon. (The simulator in the covert 2PC protocol of section 7 extracts players' inputs from covert CCA ciphertexts, and does not need to extract any further witnesses used in the proofs.) This strong covertness property of [2] implies covert soundness of CKEM defined here, but not covert simulation-soundness or sender covertness. However, it is easy to see that the RO-model variant of the CKEM of [11] satisfies both properties as well, which we briefly argue below.

In Figure 3 we show the random oracle model version of the CKEM of [11], with notation adapted to an LMI language L. Let $\mathsf{H}_i(\cdot)$ be a short-cut for $\mathsf{H}(i, \cdot)$. Assume that functions ϕ, γ map instance,witness pairs of language L into instances $x = (C, M) \in G^n \times G^{n \times m}$ and witnesses $w \in \mathbb{Z}_p^m$ of $\mathsf{LMI}_{n,m}$. Recall a sigma protocol for $\mathsf{LMI}_{n,m}$: The prover picks random $w' \leftarrow \mathbb{Z}_p^m$, sends $D = w' \cdot M$ to the verifier, and on verifier's public coin challenge e chosen uniformly in $\mathbb{Z}_p$, it outputs $z = w' + ew$ (multiplication by a scalar a vector addition in $\mathbb{Z}_p^m$). The verifier accepts if $D = z \cdot M - eC$. Note that if $C = w \cdot M$ then $D = (w' + ew) \cdot M - ew \cdot M = w' \cdot M$. Special soundness follows from the fact that two accepting transcipts (z, e) and (z', e') for $e' \neq e$ imply that $D = z \cdot M - eC = z' \cdot M - e'C$, which means that $C = w \cdot M$ for $w = (e - e')^{-1} \cdot (z - z')$. Special honest-verifier zero-knowledge simulator picks random z in $\mathbb{Z}_p^m$ and e in $\mathbb{Z}_p$ and computes $D = z \cdot M - eC$, which perfectly matches the honest verifier's view of the real prover. Consider a covert commitment which simplifies the commitment of Section 4, namely $\mathsf{Com}_{g_1,g_2}(\mathsf{m}) = ((g_1)^r, (g_2)^r (g_1)^{\mathsf{m}})$. This commitment is perfectly binding for $\mathsf{m} \in \mathbb{Z}_p$, and it

is hiding and covert under the DDH assumption for random g_2. Note that an SPHF for language $\mathsf{Lc} = \{(\mathsf{cmt}, \mathsf{m}) \,|\, \mathsf{cmt} = \mathsf{Com}_{g_1,g_2}(\mathsf{m})\}$ is a well-known SPHF $(\mathsf{KG}, \mathsf{Hash}, \mathsf{PHash})$ for the language of Diffie-Hellman tuples, i.e. $\mathsf{KG}(g_1, g_2)$ picks $\mathsf{hk} \leftarrow \mathbb{Z}_p^2$ and $\mathsf{hp} = (g_1)^{\mathsf{hk}_1}(g_2)^{\mathsf{hk}_2}$, $\mathsf{Hash}((\mathsf{cmt}, \mathsf{m}), \mathsf{hk}) = (\mathsf{cmt}_1)^{\mathsf{hk}_1}(\mathsf{cmt}_2/\mathsf{m})^{\mathsf{hk}_2}$, and $\mathsf{PHash}((\mathsf{cmt}, \mathsf{m}), r, \mathsf{hp}) = (\mathsf{hp})^r$.

On inputs (g_1, g_2) and $(C, M) = \phi(x)$, and on R's input w s.t. $C = w \cdot M$:

R: Pick $w' \leftarrow \mathbb{Z}_p^m$ and $r \leftarrow \mathbb{Z}_p$, set $D = w' \cdot M$, $\mathsf{cmt} \leftarrow \mathsf{Com}_{g_1,g_2}(\mathsf{H}_2(D); r)$, $e = \mathsf{H}_1(x, \mathsf{cmt})$, $z = w' + ew$, and send (cmt, z) to S.

S: Set $D = z \cdot M - eC$ for $e = \mathsf{H}_1(x, \mathsf{cmt})$, generate $(\mathsf{hk}, \mathsf{hp}) \leftarrow \mathsf{KG}(g_1, g_2)$, send hp to R and output $\mathsf{K}_S = \mathsf{Hash}((\mathsf{cmt}, \mathsf{H}_2(D)), \mathsf{hk})$.

R: Output $\mathsf{K}_R = \mathsf{PHash}((\mathsf{cmt}, \mathsf{H}_2(D)), r, \mathsf{hp})$.

Fig. 3. ROM version of CKEM for Linear Map Image Language due to Jarecki [11]

Theorem 2. *The CKEM of Figure 3 is covert zero-knowledge and simulation-sound for any* LMI *language* L*, assuming the DDH assmumption in ROM.*

Proof. The algorithm TRec uses the fact that H is a random oracle H as its *trapdoor* as follows: TRec on input $(C, M) = \phi(x)$ picks $z \leftarrow \mathbb{Z}_p^m$, $e \leftarrow \mathbb{Z}_p$, and $r \leftarrow \mathbb{Z}_p$, computes $D = z \cdot M - eC$ and $\mathsf{cmt} \leftarrow \mathsf{Com}_{g_1,g_2}(\mathsf{H}_2(D); r)$, sets $\mathsf{H}_1(x, \mathsf{cmt})$ to e, aborting if $\mathsf{H}_1(x, \mathsf{cmt})$ was already set, and sends (cmt, z) to S. When TRec gets hp from S, it can compute $\mathsf{K}_R = \mathsf{PHash}((\mathsf{cmt}, \mathsf{H}_2(D)), r, \mathsf{hp})$ as in R, because TRec holds the same witness r that Rec does, i.e. that $\mathsf{m} = \mathsf{H}_2(D)$ is committed in cmt for $D = z \cdot M - eC$ and $e = \mathsf{H}_1(x, \mathsf{cmt})$.

Zero Knowledge: Since for every m, commitment $\mathsf{Com}(\mathsf{m})$ samples a random element from space of size $|G|$, the probability that TRec aborts because $\mathsf{H}_1(x, \mathsf{cmt})$ was already queried is negligible. Hence, by the same argument as in the case of the standard Fiat-Shamir NIZK (D, e, z), the distribution of tuples (cmt, e, z) produced in this simulation is statistically close to that produced by the real prover.

(Trapdoor) Receiver Covertness: Message (cmt, z) sent by either TRec or Rec is indistinguishable from a random string because in both cases z is uniform in $\mathbb{Z}_p^m$ and Com is a covert commitment under the DDH assumption.

Sender Covertness: S's message $\mathsf{hp} = (g_1)^{\mathsf{hk}_1}(g_2)^{\mathsf{hk}_2}$ is uniform in G.

Covert Simulation-Soundness: The strong (i.e. "proof-of-knowledge") covert soundness shown for this CKEM in [11] implies the standard covert soundness of this CKEM in ROM, i.e. that no efficient adversary interacting with S (and H) on $x \notin \mathsf{L}$ can distinguish S's messages *and* its output K_S from random, i.e. specifcially that $(\mathsf{hp}, \mathsf{K}_S)$ pair is statistically close to uniform in G^2. It is easy to see that this remains true given access to a TRec simulator described above simulating the $((\mathsf{cmt}, z), \mathsf{K}_R)$ interactions with Rec for any $x' \neq x$, and this is

because the NIZK challenge is computed using $\mathsf{H}_1(x,\cdot)$ on the soundness challenge x and using $\mathsf{H}_1(x',\cdot)$ on all instances simulated by TRec, and in ROM these are all independent instances of random functions, hence simulation $\mathsf{TRec}(x')$ on any $x' \neq x$ does not affect the view of the interaction with $\mathsf{S}(x)$.

7 Covert Computation of General 2-Party Functions

We show a protocol Π_{COMP}, in Figure 4, which realizes the concurrent 2-party covert computation functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ in the CRS model. The protocol we show uses a covert variant of Yao's garbled circuit protocol [16], with a variant of the cut-and-choose approach to get security against malicious players, see e.g. [13]. A standard way of implementing a cut-and-choose method involves protocol tools which are inherently non-covert: The first tool is that the circuit garbling party, B, sends commitments to n copies of the garbled circuit and then follows up by decommitting a randomly chosen half of them, so that party A can verify that the opened circuits are formed correctly *and* that they were committed in B's first message. However, if B sends a commitment followed by a decommitment, this decommitment can be verified publicly, hence party A can distinguish B from a random beacon regardless of the inputs which A enters into the computation. Secondly, a cut-and-choose protocol can also use zero-knowledge proofs, e.g. to prove that the OT's are performed correctly or that the keys opened for different circuit copies correspond to the same inputs, and zero-knowledge proofs are also inherently non-covert. Here we show that (concurrent and simulation-sound) covert CKEM's can be effectively used in both cases: First, we use CKEM's in place of all zero-knowledge proofs. Secondly, we replace a commit/decommit sequence with a commitment c, release of the committed plaintext m, and a covert CKEM performed on a statement that there exists decommitment d (the CKEM receiver's witness) s.t. d decommits c to m. We use a perfectly binding commitment so that the above language is non-trivial. Specifically, we implement the commitment scheme using covert Cramer-Shoup encryption, which assures non-malleability, allows for straight-line extraction of player's inputs via the decryption oracle, and its arithmetic structure enables efficient CKEM on related statements.

The resulting protocol realizes the concurrent *covert* 2PC at the cost of $O(1)$ rounds, $O(\tau|C|\tau)$ bandwidth, $O(\tau|C|)$ symmetric cipher operations, and $O(\tau|W|)$ exponentiations τ is the security parameter, $|C|$ is the number of circuit gates and $|W|$ is the number of input bits. This comes very close to the secure (but *not* covert) cut-and-choose 2PC protocols which have the same asymptotic bandwidth and symmetric crypto costs, but require only $O(\tau^2)$ rather than $O(\tau|W|)$ exponentiations due to OT extension techniques (which we do not try to implement here, although we see no inherent reasons why they should be hard to adopt). Regarding the exact costs, the protocol exchanges 9 messages when implemented with the CKEM's of [2], and since it uses a simple variant of cut-and-choose method, it requires transmission of $n = 2.4 \cdot \tau$ copies of the garbled circuit to upper-bound statistical cheating probability by $2^{-\tau}$ per protocol

instance. Both of these parameters are a factor of about 2 worse than the most efficient standard, i.e. non-covert, versions of cut-and-choose Yao's 2PC protocol [13]. Similarly the CKEM's of [2] are about a factor of 2 worse than the known ZK proofs for the same languages.

Covert Garbled Circuit. Before presenting the protocol, we specify how it uses the tools of Section 4 and the CKEM's of Section 6. Let $f : \{0,1\}^{n_x} \times \{0,1\}^{n_y} \to \{0,1\}^{n_z} \times \{0,1\}^{n_v}$ and $g : \{0,1\}^{n_x} \times \{0,1\}^{n_y} \to \{0,1\}$. Let f_z, f_v satisfy $f(x,y) = (f_z(x,y), f_v(x,y))$. Let x, y, c, d, t, u be bitstrings s.t. $|x| = n_x$, $|y| = n_y$, $|c| = |d| = n_z + n_v\tau$, $|u| = 1$, and $t = t_1^0|t_1^1|...|t_{n_v}^0|t_{n_v}^1$ where $|t_i^b| = \tau$ for all i, b. We will use covert Yao's garbling procedure GCgen of [16] for function $f|_g$ defined as follows:

$$f|_g((x,c),(y,d,t,u)) = \begin{cases} c \oplus d & \text{if } \ g(x,y) = 0 \vee u = 0 \\ f_z(x,y)|t_1^{v[1]}|...|t_{n_v}^{v[n_v]} & \text{otherwise, where } v = f_v(x,y) \end{cases}$$

Let $\overline{X} = X \cup C$ and $\overline{Y} = Y \cup D \cup T \cup U$. If s is any part of the circuit inputs $(x,c),(y,d,t,u)$ and $w \in W$ then we overload notation by using $s[w]$ to denote the bit of s corresponding to input wire w. To enable efficient CKEM's for related languages we modify GCgen so it chooses wire keys $k^{w,b}$ corresponding to A's input wires, i.e. for $w \in \overline{X}$, as random elements in G, but keys $k^{w,b}$ corresponding to B's input wires, i.e. for $w \in \overline{Y}$, as random elements in $\mathbb{Z}_p$. Note that either key type can be used to derive a standard symmetric key, e.g. using a strong extractor with a seed specified in the CRS. We use encryption E instead of commitment Com to commit to the wire keys, but we encrypt these keys differently, namely as $\mathsf{E}_{\mathsf{pk}}(k^{w,b})$ for $w \in \overline{X}$ and as $\mathsf{E}_{\mathsf{pk}}(g^{k^{w,b}})$ for $w \in \overline{Y}$.

Covert CKEM's. Protocol Π_{COMP} uses CKEM's for two languages in class LMI, both formed as conjunctions of various LMI languages listed in Section 6.1: Language LA of correctly formed wire-bit ciphertexts sent by A, and language LB of correctly formed messages sent by B. LA consists of pair $\{\mathsf{ct}^w\}_w, \{\mathsf{ct}_i^w\}_{i,w}$ s.t. all ciphertext,label pairs $(\mathsf{ct}^w, (\ell_A, w))$ and $(\mathsf{ct}_i^w, (\ell_A, i, w))$ are in $\mathsf{Lbit}(\mathsf{pk})$. Language LB, omitting labels, consists of the following statements:

(1) $\{(k_i^w, \mathsf{ct}^w, \mathsf{ck}_i^{w,0}, \mathsf{ck}_i^{w,1})\}_{i \notin S, w \in \overline{Y} \setminus D}$ which consists of statements in $\mathsf{Ldis}(\mathsf{pk})$,
(2) $\{(k_i^w, \mathsf{ck}_i^{w,0}, \mathsf{ck}_i^{w,1})\}_{i \notin S, w \in D}$ which consists of statements in $\mathsf{Ldis}(\mathsf{pk})'$,
(3) $\{(\mathsf{gc}_i, \mathsf{cgc}_i)\}_{i \in [n]}$ s.t. each pair $(\mathsf{cgc}_i, H(\mathsf{gc}_i))$ is in $\mathsf{Le}(\mathsf{pk})$,
(4) $\{(k_i^{w,b}, \mathsf{ck}_i^{w,b})\}_{i \in S, w \in \overline{X}, b}$ which consists of statements in $\mathsf{Le}(\mathsf{pk})$,
(5) $\{(k_i^{w,b}, \mathsf{ck}_i^{w,b})\}_{i \in S, w \in \overline{Y}, b}$ s.t. each pair $(g^{k_i^{w,b}}, \mathsf{ck}_i^{w,b})$ is in $\mathsf{Le}(\mathsf{pk})$
(6) $\{(\mathsf{otr}_i^w, \mathsf{ct}_i^w, \mathsf{ck}_i^{w,0}, \mathsf{ck}_i^{w,1})\}_{i \notin S, w \in \overline{X}}$ which are statements in $\mathsf{Lotr}(\mathsf{pk})$.

Additional Tools and Notation. Let $n = 2.4 \cdot k$ where k is the statistical security parameter (in Theorem 3 we assume $k{=}\tau$). Let $\mathsf{SG}(n)$ be an algorithm which generates a random subset of $n/2$ elements in $[n]$. Let F be a PRF with τ-bit keys, arguments, and outputs. Let G^ℓ be a PRG with τ-bit inputs and ℓ-bit outputs. Let $(\mathsf{SE}, \mathsf{SD})$ be a covert symmetric encryption implemented as $\mathsf{SE}_K^s(\mathsf{m}) = \mathsf{G}^{|\mathsf{m}|}(\mathsf{F}(K,s)) \oplus \mathsf{m}$ and $\mathsf{SD}_K^s(\mathsf{ct}) = \mathsf{G}^{|\mathsf{ct}|}(\mathsf{F}(K,s)) \oplus \mathsf{ct}$. By convention we

$\mathsf{PG}(1^\tau)$ picks $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Kg}(1^\tau)$ and sets $\pi_{\mathsf{ckem}} \leftarrow \mathsf{pk}$.

B1: on input $(\mathsf{Input2}, A, y, \mathsf{sid})$ and $\ell_B = (B, A, \mathsf{sid})$:
set $(\mathsf{gc}_i, \{k_i^{w,b}\}_{w\in W,b}) \leftarrow \mathsf{GCgen}_{f,g}(\mathsf{r}_i^{\mathsf{gc}})$ and $\mathsf{cgc}_i \leftarrow \mathsf{Com}_{\mathsf{pk}}^{(\ell_B,i)}(\mathsf{gc}_i; \mathsf{r}_i^{\mathsf{cgc}})$ for $i\in[n]$;
set $\mathsf{ck}_i^{w,b}$ to $\mathsf{E}_{\mathsf{pk}}^{(\ell_B,w,b)}(k_i^{w,b}; \mathsf{r}_{i,w,b}^{\mathsf{ck}})$ for $w \in \overline{X}$ and $\mathsf{E}_{\mathsf{pk}}^{(\ell_B,w,b)}(g^{k_i^{w,b}}; \mathsf{r}_{i,w,b}^{\mathsf{ck}})$ for $w \in \overline{Y}$, for all i, b;
send $\mathsf{m}_B^1 = (\{\mathsf{cgc}_i\}_{i\in[n]}, \{\mathsf{ck}_i^{w,b}\}_{i\in[n],w\in W,b})$ to A.

A1: on input $(\mathsf{Input1}, B, x, \mathsf{sid})$ and m_B^1, and $\ell_A = (A, B, \mathsf{sid})$:
sample r^{SG}, set $S \leftarrow \mathsf{SG}(n; \mathsf{r}^{\mathsf{SG}})$, pick $c_i \leftarrow \{0,1\}^{n_z+n_v\tau}$ for $i\in[n]$;
set $\mathsf{x}_A \leftarrow (\{\mathsf{ct}^w = \mathsf{E}_{\mathsf{pk}}^{(\ell_A,w)}(g^{x[w]}; \mathsf{r}_w^{\mathsf{ct}})\}_{w\in X}, \{\mathsf{ct}_i^w = \mathsf{E}_{\mathsf{pk}}^{(\ell_A,i,,w)}(g^{c_i[w]}; \mathsf{r}_{w,i}^{\mathsf{ct}})\}_{i\in[n],w\in C})$;
set $\mathsf{w}_A \leftarrow (\{x[w], \mathsf{r}_w^{\mathsf{ct}}\}_{w\in X}, \{c_i[w], \mathsf{r}_{w,i}^{\mathsf{ct}}\}_{i\in[n],w\in C})$, send $\mathsf{m}_A^1 = (\mathsf{r}^{\mathsf{SG}}, \mathsf{x}_A)$ to B.

A,**B** run $\mathsf{CKEM}_{\mathsf{LA}}$ on (x_A, ℓ_A) and A's input w_A; let B output K_B and A output K'_B.

B2: on m_A^1 and K_B:
generate $S \leftarrow \mathsf{SG}(n; \mathsf{r}^{\mathsf{SG}})$, set $u \leftarrow 1$, pick $t \leftarrow \{0,1\}^{2n_v\tau}$ and $d_i \leftarrow \{0,1\}^{n_z+n_v\tau}$ for $i\in[n]$;
set $\mathsf{ct}^{\mathsf{B}} \leftarrow \{\mathsf{ct}^w = \mathsf{E}_{\mathsf{pk}}^{(\ell_B,w)}(g^{\overline{y}[w]}; \mathsf{r}_w^{\mathsf{ct}})\}_{w\in\overline{Y}\setminus D}$, $\mathsf{ks}_i^B \leftarrow \{k_i^w = k_i^{w,\overline{y}_i[w]}\}_{w\in\overline{Y}}$ for $\overline{y} = y|t|u$, $\overline{y}_i = \overline{y}|d_i$;
and $\mathsf{otr}_i \leftarrow \{\mathsf{otr}_i^w = \mathsf{OTrsp}_{\mathsf{pk}}(\mathsf{ct}_i^w, k_i^{w,0}, k_i^{w,1}; \mathsf{r}_{i,w}^{\mathsf{otr}})\}_{w\in\overline{X}}$ for $i\notin S$, where $\mathsf{ct}_i^w = \mathsf{ct}^w$ for $w\in X$;
send $\mathsf{m}_B^2 = \mathsf{SE}_{\mathsf{K}_B}^0[\mathsf{ct}^{\mathsf{B}}, \{\mathsf{r}_i^{\mathsf{gc}}\}_{i\in S}, \{\mathsf{gc}_i, \mathsf{ks}_i^B, \mathsf{otr}_i\}_{i\notin S}]$ to A.

A2: on K'_B and m_B^2:
decrypt $(\mathsf{ct}^{\mathsf{B}}, \{\mathsf{r}_i^{\mathsf{gc}}\}_{i\in S}, \{\mathsf{gc}_i, \mathsf{ks}_i^B, \mathsf{otr}_i\}_{i\notin S})$ as $\mathsf{SD}_{\mathsf{K}'_B}^0(\mathsf{m}_B^2)$;
set $\mathsf{ks}_i^A \leftarrow \{k_i^w = \mathsf{OTfin}_{\mathsf{pk}}(\overline{x}_i[w], \mathsf{r}_{w,i}^{\mathsf{ct}}, \mathsf{otr}_i^w)\}_{w\in\overline{X}}$ for $i\notin S$, $\overline{x}_i = x|c_i$, and $\mathsf{r}_{w,i}^{\mathsf{ct}} = \mathsf{r}_w^{\mathsf{ct}}$ for $w\in X$;
set $(\mathsf{gc}_i, \{k_i^{w,b}\}_{w,b}) \leftarrow \mathsf{GCgen}_{f,g}(\mathsf{r}_i^{\mathsf{gc}})$ for $i\in S$ and $w_i \leftarrow \mathsf{GCev}(\mathsf{gc}_i, (\mathsf{ks}_i^A \cup \mathsf{ks}_i^B))$ for $i\notin S$.

A,**B** run $\mathsf{CKEM}_{\mathsf{LB}}$ on (x_B, ℓ_B) and B's input w_B for $\mathsf{x}_B = (\{k_i^w, \mathsf{ct}^w, \mathsf{ck}_i^{w,0}, \mathsf{ck}_i^{w,1}\}_{i\notin S,w\in\overline{Y}\setminus D}$, $\{k_i^w, \mathsf{ck}_i^{w,0}, \mathsf{ck}_i^{w,1}\}_{i\notin S,w\in D}, \{\mathsf{gc}_i, \mathsf{cgc}_i\}_{i\in[n]}, \{k_i^{w,b}, \mathsf{ck}_i^{w,b}\}_{i\in S,w\in W,b}, \{\mathsf{otr}_i^w, \mathsf{ct}_i^w, \mathsf{ck}_i^{w,b}\}_{i\notin S,w\in\overline{X},b})$, and w_B contains input y and randomness of B. Let A output K_A and B output K'_A.

A3: If $\exists R\subset[n]$ s.t. $|R|=n/4$ and $\exists w$ s.t. $\forall_{i\in R}\ w_i = w$ then set $(z|t_1|...|t_{n_v}) := w$ and $\mathsf{m}_A^2 \leftarrow \mathsf{SE}_{\mathsf{K}_A}^0(\mathsf{F}(\mathsf{K}'_B, 1), t_1, ..., t_{n_v})$; otherwise set $z := \perp$ and $\mathsf{m}_A^2 \leftarrow \{0,1\}^{\tau(n_v+1)}$. Send m_A^2 to B.

B3: Set $\tau|t_1|...|t_{n_v} \leftarrow \mathsf{SD}_{\mathsf{K}'_A}^0(\mathsf{m}_A^2)$. Parse t as $t_1^0|t_1^1|...|t_{n_v}^0|t_{n_v}^1$. If $\tau = \mathsf{F}(\mathsf{K}_B, 1)$ and $t_j \in \{t_j^0, t_j^1\}$ for all $j\in[n_v]$ then set $\mathsf{m}_B^3 \leftarrow \mathsf{F}(\mathsf{K}_B, 2) \oplus \mathsf{F}(\mathsf{K}'_A, 1)$ and $\forall_j$ set $v[j] := b$ s.t. $t_j = t_j^b$; otherwise pick m_B^3 at random in $\{0,1\}^\tau$ and set $v := \perp$. Send m_B^3 to A and output (Output, v).

A4: If $\mathsf{m}_B^3 \neq \mathsf{F}(\mathsf{K}'_B, 2) \oplus \mathsf{F}(\mathsf{K}_A, 1)$ then set (overwrite) $z := \perp$. Output (Output, z).

Fig. 4. UC Covert Protocol Π_{COMP} for 2-Party Function Computation $\mathsf{F}_{\mathsf{C}(f|g)}$

will use r^{P} to denote the randomness to either run procedure P or generate object P. Letter b always stands for a bit, and expressions $\{...\}_b$ stand for $\{...\}_{b\in\{0,1\}}$.

Theorem 3. *Protocol Π_{COMP} in Figure 4 realizes the concurrent 2-party covert computation functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ in the CRS model, assuming cPKE is a covert CCA public key encryption, F is a PRF, G is a PRG, $(\mathsf{GCgen}, \mathsf{GCev})$ is a covert version of Yao's garbling circuit algorithm, $(\mathsf{OTreq}, \mathsf{OTrsp}, \mathsf{OTfin})$ is a covert OT, and $\mathsf{CKEM}_{\mathsf{LA}}$ and $\mathsf{CKEM}_{\mathsf{LB}}$ are covert zero-knowledge and simulation-sound CKEM's for languages resp. LA and LB.*

Proof. Figures 5 and 6 together contain the code of an ideal-model adversary $\mathcal{A}^*$, i.e. a simulator, which interacts with functionality $\mathsf{F}_{\mathsf{C}(f|g)}$ and the real-world adversary Adv in such a way that for every efficient Adv and environment $\mathcal{Z}$, the environment's view of an interaction with Adv and honest players performing protocol Π_{COMP} is indistinguishable from $\mathcal{Z}$'s view of an interaction with $\mathsf{Adv}, \mathcal{A}^*, \mathsf{F}_{\mathsf{C}(f|g)}$. We split $\mathcal{A}^*$'s into two figures for presentation convenience, but these are two parts of the same algorithm: Figure 5 describes how $\mathcal{A}^*$ handles an interaction with $\mathsf{F}_{\mathsf{C}(f|g)}$ which pertain to any honest ideal-world party B, while Figure 6 describes how $\mathcal{A}^*$ handles an interaction with $\mathsf{F}_{\mathsf{C}(f|g)}$ which pertain to any honest ideal-world party A. The $\mathcal{A}^*$ code we provide in these figures assumes that the CRS is chosen by setting $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(1^\tau)$, $R \leftarrow \{0,1\}^\tau$, and $\pi \leftarrow (\mathsf{pk}, R)$, and that value sk is given to $\mathcal{A}^*$ as the simulation trapdoor.

We will split the argument that $\mathcal{Z}$'s (for any Adv) interaction with real-world honest players is indistinguishable from an interaction with $\mathcal{A}^*$, $\mathsf{F}_{\mathsf{C}(f|g)}$, and dummy ideal-world honest players, into four parts, distinguishing by two types of honest parties, A and B, and breaking each of these cases into two subcases depending on whether their inputs, resp. x and y, are bitstrings in $\{0,1\}^{n_x}$ resp. $\{0,1\}^{n_y}$, or the non-participation signals $\perp$. In each case we will show that the interaction between Adv and an honest party following protocol Π_{COMP} on a session with identifier sid is indistinguishable from the corresponding session executed by $\mathcal{A}^*$ interacting with $\mathsf{F}_{\mathsf{C}(f|g)}$ and a dummy (ideal-world) honest party. We will call an interaction of $\mathcal{Z}$ (for fixed Adv) with the algorithms we specify a *security game*, denoted G_i, and we will write $G_i \approx G_j$ to denote that $\mathcal{Z}$'s output in G_i is indistinguishable of $\mathcal{Z}$'s output in G_j.

Case 1A, honest B on input $y^* \in \{0,1\}^{n_y}$: Let G_0 be the interaction of $\mathcal{Z}$ and Adv in the real world where player B gets input $(\mathsf{Input2}, A, y^*, \mathsf{sid})$ from $\mathcal{Z}$ for $y^* \neq \perp$. Let G_1 be a modification of G_0, which uses the private key sk corresponding to the key pk in the CRS, to decrypt wire-input ciphertexts $\{\mathsf{ct}^w\}_{w\in X}$, $\{\mathsf{ct}_i^w\}_{i\in[n],w\in C}$ which Adv sends in message m_A^1 to B on this session, and if any of these ciphertexts does not decrypt to a bit then G_1 sets $x := \perp$, at the end of $\mathsf{CKEM}_{\mathsf{LA}}$ it picks K_B as an independent random string, and then it continues computation as B in protocol Π_{COMP}. $G_1 \approx G_0$ by the simulation soundness of $\mathsf{CKEM}_{\mathsf{LA}}$: The reduction uses sk, in particular to test whether x_A defined by m_A^1 on this session is in LA. If $\mathsf{x}_A \notin \mathsf{LA}$, which is the only case G_1 differs from G_0, then if $\mathcal{Z}$ distinguishes between G_0 and G_1 then the reduction distinguishes between the real and the random key $\mathsf{K}_B = \mathsf{K}_S$ output by $\mathsf{Snd}(\pi, (\mathsf{x}_A, \ell_A))$.

On behalf of honest B, on trapdoor sk and input $(\mathsf{Input2}, \mathsf{sid}, A, B)$ from $\mathsf{F}_{\mathsf{C}(f|g)}$:

(1) compute $\{\mathsf{cgc}_i, \{\mathsf{ck}_i^{w,b}\}_{w,b}\}_{i\in[n]}$ and send to A as B does in step B1;

(2) on $\mathsf{m}_A^1 = (\mathsf{r}^{\mathsf{SG}}, \mathsf{x}_A)$ for $\mathsf{x}_A = (\{\mathsf{ct}^w\}_{w\in X}, \{\mathsf{ct}_i^w\}_{i,w\in C})$ from A in A1, decrypt all ct's using sk to obtain $x, c_1, ..., c_n$, overwrite $x{:=}\perp$ if any decryption returns $\perp$ or not a bit, and send $(\mathsf{Input1}, B, x, \mathsf{sid})$ to $\mathsf{F}_{\mathsf{C}(f|g)}$ and receive $(\mathsf{Output}, z, \mathsf{sid})$ from $\mathsf{F}_{\mathsf{C}(f|g)}$;
If $x = \perp$ then in steps (4,6) below pick $\mathsf{m}_B^2, \mathsf{m}_B^3$ at random and in step (5) run $\mathsf{Rec}^{\$(\tau)}$.

(3) run $\mathsf{Snd}(\pi, (\mathsf{x}_A, \ell_A))$ in $\mathsf{CKEM}_{\mathsf{LA}}$ interacting with A, let K_B be Snd's output;

(4) set $y{:=}0^{n_y}$, $u{:=}0$, $t \leftarrow \{0,1\}^{2n_v\tau}$; if $z = \perp$ then $\forall i$ pick $d_i \leftarrow \{0,1\}^{n_z+n_v\tau}$; o/w set $t' \leftarrow \{0,1\}^{n_v\tau}$ and $d_i \leftarrow c_i \oplus (z|t')$ $\forall i$; compute ct^{B} and $\{\mathsf{ks}_i^B, \mathsf{otr}_i\}_{i\notin S}$ as in step B2, and encrypt them under K_B to A with $\{\mathsf{r}_i^{\mathsf{gc}}\}_{i\in S}$ and $\{\mathsf{gc}_i\}_{i\notin S}$ from step (1);

(5) run $\mathsf{Rec}(\pi, (\mathsf{x}_B, \ell_B), \mathsf{w}_B)$ in $\mathsf{CKEM}_{\mathsf{LB}}$ on inputs computed as in Π_{COMP}, output K'_A;

(6) on m_A^2 from A, decrypt it as $\tau|t_1|...|t_{n_v}$ using K'_A; if $\tau = \mathsf{F}(\mathsf{K}_B, 1)$ and $t_1|...|t_{n_v} = t'$ then send $\mathsf{m}_B^3 = \mathsf{F}(\mathsf{K}_B, 2) \oplus \mathsf{F}(\mathsf{K}'_A, 1)$ to A and $(\mathsf{Output}, \mathsf{sid}, B, \mathsf{T})$ to $\mathsf{F}_{\mathsf{C}(f|g)}$, and otherwise send random m_B^3 in $\{0,1\}^\tau$ to A and $(\mathsf{Output}, \mathsf{sid}, B, \mathsf{F})$ to $\mathsf{F}_{\mathsf{C}(f|g)}$.

Fig. 5. Simulator $\mathcal{A}^*$ (part 1) showing that Π_{COMP} UC-realizes covert 2PC funct. $\mathsf{F}_{\mathsf{C}(f|g)}$

Let G_2 modify G_1 s.t. B acts like a random beacon on a session where $x = \perp$ as above, i.e. it sends random strings as messages $\mathsf{m}_B^2, \mathsf{m}_B^3$ and runs $\mathsf{Rec}^{\$(\tau)}$ in $\mathsf{CKEM}_{\mathsf{LB}}$. $G_2 \approx G_1$ because, (1) by the PRF property of F, the key used in encryption SE in m_B^2 is indistinguishable from value $\mathsf{F}(\mathsf{K}_B, 2)$ which is used as a one-time pad in m_B^3, so we can replace them by independent random strings, which in particular means that m_B^3 can be a random string, (2) by the PRG property of G, m_B^2 can be replaced by a random string, and (3) since G_2 doesn't use Rec's output K'_A from $\mathsf{CKEM}_{\mathsf{LB}}$, by covert zero-knowledge of this CKEM we can replace Rec with $\mathsf{Rec}^{\$(\tau)}$. Note that G_2 acts like $\mathcal{A}^*$ if $x = \perp$.

We will consider a modification which changes values $y, \{d_i\}, t, v$ which B inputs into garbled circuit computation. Let D denotes the distribution of the circuit inputs $(\overline{x}, \overline{y}) = ((x, \{c_i\}), (y, \{d_i\}, t, v))$ in G_2 as a function of the randomness of $\mathcal{Z}$, Adv and G_2: The (Adv, env) together specify $x, \{c_i\}, y^*$, we assume that $x \neq \perp$, and G_2 sets $y{:=}y^*$, $v{:=}1$, and t and all d_i's are random strings. Let D' be an alternative distribution defined as follows: Given $(x, \{c_i\})$ decrypted from m_A^1 and y^* specified by $\mathcal{Z}$ (and assuming $x \neq \perp$), set $y{:=}0^{n_y}$, $v{:=}0$, pick t at random in $\{0,1\}^{2n_v\tau}$, and if $g(x, y^*) = 0$ then pick each d_i at random in $\{0,1\}^{n_z+n_v\tau}$, but if $g(x, y^*) = 1$ then let $(z, v){:=}f(x, y^*)$, define $t'_j{:=}t_j^{v[j]}$ for each $j{\in}[n_v]$ where $t = t_1^0|t_1^1|...|t_{n_v}^0|t_{n_v}^1$ (in other words t'_j's are chosen to encode the bits of B's output v given the authenticator values B chose in t), and let $d_i = c_i \oplus (z|t'_1|...|t'_{n_v})$ for each i.

Consider game G_3 which is like G_2 except the ciphertexts $\mathsf{ct}^{\mathsf{B}} = \{\mathsf{ct}^w\}_{w\in\overline{Y}\setminus D}$ are computed as encryptions of (y, t, v) chosen according to D' instead of D, but the keys in ks_i^B are still chosen according to $\overline{y}$ chosen as in G_2. Since B in G_3 will not have correct witnesses in $\mathsf{CKEM}_{\mathsf{LB}}$, it will run TRec using trapdoor sk

in that step. $G_3 \approx G_2$ because, (1) by the simulation soundness of $\mathsf{CKEM}_{\mathsf{LB}}$, we can replace Rec by TRec in game G_2, at which point the game does not use the randomness in ciphertexts in ct^{B}, and (2) by the CCA security of cPKE, we can change the domain in which the plaintexts (y, t, v) in these ciphertexts are drawn, from D to D': The reduction will use the decryption oracle to decrypt ciphertexts sent by Adv in m_A^1, but these have different labels than the ciphertexts in ct^{B}, and it will also use this decryption oracle to implement TRec, but again on ciphertexts with different labels. (The reduction will also use sk to implement the code of $\mathcal{A}^*$ on protocol sessions with other sid's, but these ciphertexts will therefore also pertain to different labels.)

Consider G_4 in which not only the plaintexts in ct^{B} are chosen according to D', but also B picks the key sets $\mathsf{ks}_i^B = \{k_i^w\}_{w \in \overline{Y}}$ using values $(y, \{d_i\}, t, v)$ chosen in D' instead of in D. $G_4 \approx G_3$ by reduction to B's security in Yao's garbled circuit procedure GCgen, because the distribution of outputs of the n copies of circuit $f|_g$ on inputs defined by $\overline{x}, \overline{y}$ (conditioned on $x \neq \perp$) is the same for $\overline{x}, \overline{y}$ sourced in D and in D': Let $w_i = f|_g(x, c_i, y, d_i, t, v)$ for $i \in [n]$. In the case case $g(x, y^*) = 0$ in G_3 (i.e. circuit inputs are drown from D) we have $w_i = c_i \oplus d_i$ for each i, but in the same case in G_4 we have $w_i = c_i \oplus d_i$ for each i as well because v is set to 0. (Note that it does not matter if $g(x, 0^{n_y}) = 0$, hence we can set $y{:=}0^{n_y}$ in G_4: This is why we add the "simulation bit" v to circuit $f|_g$.) In the case $g(x, y^*) = 1$, in G_3 we have $w_i = (z|t_1|...|t_{n_v})$ for each i where $t_1|...|t_{n_v}$ is a random string which encodes B's output v for $(z, v) = f(x, y^*)$. But the same happens in G'' in this case: Since $v = 0$, we have $w_i = c_i \oplus d_i$ for each i, but since $d_i = c_i \oplus (z|t'_1|...|t'_{n_v})$, we have $w_i = z|t'_1|...|t'_{n_v}$ for each i, where $t'_1|...|t'_{n_v}$ are also random strings which encode v for $(z, v) = f(x, y^*)$.

Let G_5 be like G_4 except it runs Rec instead of TRec in $\mathsf{CKEM}_{\mathsf{LB}}$. Note that G_5 has all the witnesses it needs because G_5 forms ciphertexts ct^{B} and key sets ks_i^B in a consistent way, but using values $\overline{y}$ chosen from D^2 rather than D^1. $G_5 \approx G_4$ because of covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LB}}$.

Consider G_6 which modifies G_5 by changing how B computes its output in step B3 as follows: If m_A^2 decrypts under K_A to $\tau|t'_1|...|t'_{n_v}$ for $\tau = \mathsf{F}(\mathsf{K}_B, 1)$ then output v, otherwise output $\perp$. $G_6 \approx G_5$ by security of the Yao's garbled circuit procedure GCgen and the OT scheme $(\mathsf{OTreq}, \mathsf{OTrsp}, \mathsf{OTfin})$, because garbled circuit evaluation hides everything about B's inputs except the circuit output, and that contains only values $t'_1, ..., t'_{n_v}$ in t. In particular Adv cannot return any other value in t, except for negligible probability, and thus B's output is either v or $\perp$ except for negligible probability

Finally, note that G_6 proceeds like the simulator $\mathcal{A}^*$ interacting with $\mathsf{F}_{\mathsf{C}(f|g)}$ and ideal-world honest B who receives $(\mathsf{Input2}, A, y^*, \mathsf{sid})$ from $\mathcal{Z}$: The only difference is that G_5 evaluates $g(x, y^*)$ and $f(x, y^*)$ locally, while in the simulation $\mathcal{A}^*$ sends x to $\mathsf{F}_{\mathsf{C}(f|g)}$ and both functions are evaluated by $\mathsf{F}_{\mathsf{C}(f|g)}$. However, note that $\mathsf{F}_{\mathsf{C}(f|g)}$ replies to $\mathcal{A}^*$ with $z = \perp$ if and only if $g(x, y^*) = 0$, and that (1) $\mathcal{A}^*$ follows the same procedure on $z = \perp$ as G_6 does if $g(x, y*) = 0$, and (2) $\mathcal{A}^*$ follows the same procedure on $z = f_z(x, y^*) \neq \perp$ as G_6 does using $z = f_z(x, y^*)$ when $g(x, y) = 1$. Moreover, in both cases B can output only $v = f_v(x, y^*)$ or

$\perp$, and the condition which controls which is the case is the same in both interactions, i.e. B outputs v if and only m_A^3 decrypts to $\mathsf{F}(\mathsf{K}_B, 1)|t'_1|...|t'_{n_v}$. Hence we conclude that $\mathcal{Z}$'s view of an interaction with honest B on input $y^* \neq \perp$ is indistinguishable from its view of an interaction with $\mathcal{A}^*$ and an ideal dummy player B communicating through $\mathsf{F}_{\mathsf{C}(f|g)}$.

Case 1B, honest B on input $y^* = \perp$: Let G_0 be the interaction of $\mathcal{Z}$ and Adv in the real world where player B gets input $(\mathsf{Input2}, A, \perp, \mathsf{sid})$ from $\mathcal{Z}$. Note that in this case B is a random beacon, i.e. its messages $\mathsf{m}_B^1, \mathsf{m}_B^2, \mathsf{m}_B^3$ are random strings of the appropriate length, and we will assume for notational convenience that B follows $\mathsf{Snd}^{\$(\tau)}$ in $\mathsf{CKEM}_{\mathsf{LA}}$ and $\mathsf{TRec}^{\$(\tau)}$ in $\mathsf{CKEM}_{\mathsf{LB}}$. Also B always sends $\perp$ back to $\mathcal{Z}$ as its computation "output".

Let G_1 be a modification of G_0 in which B forms m_B^1 as in the case $y^* \neq \perp$, i.e. it forms $\{\mathsf{gc}_i, \{k_i^{w,b}\}_{w,b}\}_i$ using $\mathsf{GCgen}_{f,g}$ and forms commitments $\{\mathsf{cgc}_i\}$ and wire-key encryptions $\{\mathsf{ck}_i^{w,b}\}$ correctly. $G_1 \approx G_0$ because the rest of G_1's computation does not involve these values, this follows from the covertness of the symmetric encryption used in ciphertexts tables by GCgen and by the covertness of encryption E (commitment Com is an instance of E).

Let G_2 be a further modification that runs Snd on statement in m_A^1 instead of $\mathsf{Snd}^{\$(\tau)}$ in $\mathsf{CKEM}_{\mathsf{LA}}$. $G_2 \approx G_1$ by covert soundness of $\mathsf{CKEM}_{\mathsf{LA}}$.

Let G_3 be a further modification which in step B2 sets $y{:=}0^{n_y}$, $v{:=}0$, and picks t and each d_i as random bitstrings and forms ct^{B} in m_B^2 as encryptions of y, t, u instead of as random strings. $G_3 \approx G_2$ by covertness of encryption cPKE.

Let G_4 be a modification where m_B^2 includes the correct sets $\{\mathsf{r}_i^{\mathsf{gc}}\}_{i \in S}$ and $\{\mathsf{ks}_i^B, \mathsf{otr}_i\}_{i \notin S}$. $G_4 \approx G_3$ because $\mathsf{r}_i^{\mathsf{gc}}$'s and the wire keys in ks_i^B's are random (given commitments cgc_i and ciphertexts ck_i), and because OT is sender-covert, given that the OTrsp payloads $k_i^{w,0}, k_i^{w_1}$ for $w \in \overline{X}$ are random in G.

The next modification, G_5, computes everything in m_B^2 correctly, including the garbled circuits $\{\mathsf{gc}_i\}_{i \notin S}$. By covertness of Yao's garbled circuit generation GCgen (and sender-security of the OT scheme), each circuit gc_i and a set of keys $\mathsf{ks}_i^B, \mathsf{ks}_i^A$ where ks_i^A is implied by otr_i, adversary's advantage in distinguishing $(\mathsf{gc}_i, \mathsf{ks}_i^B, \mathsf{ks}_i^A)$ from random is at most negligibly greater than an advantage in distinguishing the circuit output $f|_g(x, c_i, y, d_i, t, u)$ from random. However, $f|_g(x, c_i, y, d_i, t, u) = c_i \oplus d_i$ because $u = 0$, and since each d_i is chosen at random by G_5, these outputs are random strings, hence it follows that $G_5 \approx G_4$.

In the next modification, G_6, we replace $\mathsf{TRec}^{\$(\tau)}$ with TRec in $\mathsf{CKEM}_{\mathsf{LA}}$, and $G_6 \approx G_5$ by covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LA}}$.

Next, in G_7 we will follow algorithm B in step B3 (note that at this point we have all the inputs used in this step, including key K'_A output by TRec in $\mathsf{CKEM}_{\mathsf{LA}}$), except that G_7 always outputs m_B^3 chosen at random, i.e. even if the t_i's decrypted from m_A^2 satisfy the constraint $t_j \in \{t_j^0, t_j^1\}$ where t_j^0, t_j^1 are parts of t which was chosen in step B2 (note that this is the only case where B would output a non-rejection and form m_B^3 in a non-random way). We have that $G_7 \approx G_6$ by security of the Yao's garbling circuit procedure, because the outputs of $f|_g(x, c_i, y, d_i, t, u)$ in the case $u = 0$ contain no information about t

(and neither is t used in any other computation of G_7), the probability that this non-rejection constraint is satisfied is negligible.

In modification, G_8, we replace TRec in $\mathsf{CKEM}_{\mathsf{LB}}$ with Rec running on witnesses created in steps B1 and B2: Note that at this point B computes all the values in $\mathsf{m}_B^1, \mathsf{m}_B^2$ correctly, except that it picks its circuit inputs $y, \{d_i\}, t, u$ as the simulator $\mathcal{A}^*$ does in the case $\mathsf{F}_{\mathsf{C}(f|g)}$ returns $z = \perp$. We have $G_8 \approx G_7$ by covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LB}}$, since covert zero-knowledge implies indistinguishability of an interaction with TRec and Rec *and* the keys K they output.

Next, let G_9 be a modification of G_8 where in step B3 we remove the clause introduced in game G_7 i.e. in game G_9 message m_B^3 is formed as in protocol Π_{COMP} in case t_i's decrypted from m_A^2 satisfy the constraint $t_j \in \{t_j^0, t_j^1\}$ where pairs t_j^0, t_j^1 form string t. However, by the same argument from the security of Yao's garbling procedure, we have that t is indistinguishable to Adv, hence the probability of this clause acting is negligible, and consequently $G_9 \approx G_8$.

Note that G_9 proceeds like the simulator $\mathcal{A}^*$ interacting with $\mathsf{F}_{\mathsf{C}(f|g)}$ and ideal-world honest B who receives $(\mathsf{Input2}, A, \perp, \mathsf{sid})$ from $\mathcal{Z}$: Game G_9 simply assumes that $z = \perp$, but $\mathsf{F}_{\mathsf{C}(f|g)}$ sends $z = \perp$ to $\mathcal{A}^*$ in case the input of the ideal-world player B is $\perp$. Also, in G_9 player B always outputs $\perp$, but this is also the output that the ideal-world player B outputs in interaction with $\mathsf{F}_{\mathsf{C}(f|g)}$ if its computation input is $\perp$.

On behalf of <u>honest A</u>, on trapdoor sk and input $(\mathsf{Input1}, \mathsf{sid}, A, B)$ from $\mathsf{F}_{\mathsf{C}(f|g)}$:

(1) on $\mathsf{m}_B^1 = \{\mathsf{cgc}_i, \{\mathsf{ck}_i^{w,b}\}_{w,b}\}_{i\in[n]}$ from B, set $x{:=}0^{n_x}$ and $c_i \leftarrow \{0,1\}^{n_z+n_v\tau}\ \forall i$, compute $\mathsf{x}_A{:=}(\{\mathsf{ct}^w\}_{w\in X}, \{\mathsf{ct}_i^w\}_{i,w\in C})$, w_A and message m_A^1 as A does in step A1;

(2) run $\mathsf{Rec}(\pi, (\mathsf{x}_A, \mathsf{w}_A, \ell_A))$ in $\mathsf{CKEM}_{\mathsf{LA}}$ interacting with B, let K'_B be Rec's output;

(3) on m_B^2 from B, decrypt it using K'_B to get ct^{B}, $\{\mathsf{r}_i^{\mathsf{gc}}\}_{i\in S}$, $\{\mathsf{gc}_i, \mathsf{ks}_i^B, \mathsf{otr}_i\}_{i\notin S}$; decrypt each ct^w in ct^{B} using sk to obtain each bit of y, t, u, overwrite $y{:=}\perp$ if any decryption output is not a bit or $u = 0$; and send $(\mathsf{Input2}, A, y, \mathsf{sid})$ to $\mathsf{F}_{\mathsf{C}(f|g)}$ and receive $(\mathsf{Output}, v, \mathsf{sid})$ back;

(4) compute $(\mathsf{gc}_i, \{k_i^{w,b}\}) \leftarrow \mathsf{GCgen}_{f,g}(\mathsf{r}_i^{\mathsf{gc}})$ for $i \in S$ to complete statement x_B, and run $\mathsf{Snd}(\pi, (\mathsf{x}_B, \ell_B))$ in $\mathsf{CKEM}_{\mathsf{LB}}$ on x_B, ℓ_B formed as in Π_{COMP}, let K_A be Snd's output;

(5) if $v = \perp$ then set $\mathsf{release?}{:=}\mathsf{F}$ and set m_A^2 as a random string, otherwise set $\mathsf{release?}{:=}\mathsf{T}$ and $\mathsf{m}_A^2 \leftarrow \mathsf{SE}^0_{\mathsf{K}_A}(\mathsf{F}(\mathsf{K}'_B, 1), t_1, ..., t_{n_v})$ where t_i's encode v received from $\mathsf{F}_{\mathsf{C}(f|g)}$ given t decrypted from ct^{B} above; send m_A^2 to B.

(6) given m_B^3, if $\mathsf{m}_B^3 \neq \mathsf{F}(\mathsf{K}_B, 2) \oplus \mathsf{F}(\mathsf{K}'_A, 1)$ then (re)set $\mathsf{release?}{:=}\mathsf{F}$; send $(\mathsf{Output}, \mathsf{sid}, A, \mathsf{release?})$to $\mathsf{F}_{\mathsf{C}(f|g)}$.

Fig. 6. Simulator $\mathcal{A}^*$ (part 2) showing that Π_{COMP} UC-realizes covert 2PC funct. $\mathsf{F}_{\mathsf{C}(f|g)}$

Case 2A, honest A on input $x^* \in \{0,1\}^{n_x}$: Let G_0 be the interaction of $\mathcal{Z}$ and Adv in the real world where A gets input $(\mathsf{Input1}, B, x^*, \mathsf{sid})$ for $x^* \neq \perp$.

Let G_1 be a modification of G_0, which in step A2 executes as follows: Use sk corresponding to pk in the CRS to decrypt each wire-input ciphertext ct^w in ct^{B} sent by Adv in m_B^2, and to obtain each bit of values y, t, u which Adv effectively inputs into each garbled circuit computation. Overwrite $y := \perp$ if any decryption output is not a bit or if $u = 0$. Also, do not use GCev to compute $\{w_i\}_{i \notin S}$, but instead, pick each w_i at random if $g(x^*, y) = 0$ or $u = 0$, and otherwise set each w_i to $w = z|t_1|...|t_{n_v}$ where $z = f_z(x^*, y)$ and $t_1, ..., t_{n_v}$ are the authenticator values in t which encode $v = f_v(x^*, y)$. Otherwise G_1 performs all other steps as G_0. $G_1 \approx G_0$ by the security of the Yao's garbling circuit procedure and by the covert simulation-soundness of $\mathsf{CKEM}_{\mathsf{LB}}$: To see this consider first Adv which implants errors in at least $n/4$ of the circuits, i.e. in $(\mathsf{gc}_i, \{k_i^{w,b}\}_{w,b})$ committed in m_B^1 for all $i \in S'$ for $S' \subset [n]$ s.t. $|S'| \geq n/4$. By the simulation soundness of $\mathsf{CKEM}_{\mathsf{LB}}$ (note that the reduction testing the soundness of $\mathsf{CKEM}_{\mathsf{LB}}$ must have access to the trapdoor sk corresponding to the public parameters $\pi = (\mathsf{pk}, R)$) and by binding of commitment Com, we have that except for negligible probability key K_A output by this CKEM is indistinguishable for Adv, and therefore in this case $G_1 \approx G_0$ because m_A^2 is pseudorandom to Adv and m_B^3 can satisfy A's non-abort constraint with at most negligible probability. We are left with the case that there are errors in fewer than $n/4$ garbled circuits, in which case for every S the majority of circuits evaluated in step A2 are correct. By a similar argument as above we can discount the case that B forms any ciphertext in ct^{B} incorrectly or cheats in any other part of messages $\mathsf{m}_B^1, \mathsf{m}_B^2$, because otherwise $G_1 \approx G_0$ because K_A would be pseudorandom to Adv. We are left with the case that keys ks_i^B and OT response vectors otr_i encrypted in m_B^2 are formed correctly for $i \notin S$, in which case for at least $n/4$ indexes $i \notin S$ we have that value w_i which A computes in G_0 is equal to $f|_g(x^*, c_i, y, d_i, t, u)$ for some d_i and y, t, u encrypted in ct^{B}. Now, note that if $g(x^*, y) = 0$ or $u = 0$ then $w_i = c_i \oplus d_i$. Since each c_i is random and visible to Adv only in ciphertexts in $\{\mathsf{ct}_i^w\}_{w \in C}$, which are non-malleable, hence in particular encryption of y in ct^{B} cannot be related to c_i's, values w_i are indistinguishable from random in this case. If, on the other hand, $g(x^*, y) = 1$ and $u = 1$ then $w_i = (z|t_1|...|t_{n_v})$ for t_i's which encode v (given t encrypted in ct^{B}) for $(z, v) = f(x^*, y)$. Finally, note that G_2 forms w_i's exactly the same way in both cases.

Consider a modification G_2 of G_1 which replaces Rec in $\mathsf{CKEM}_{\mathsf{LA}}$ with TRec, and uses K'_A it outputs in the subsequent steps. We have that $G_2 \approx G_1$ by covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LA}}$.

Consider a modification G_3 of G_2 where ciphertexts $\{\mathsf{ct}^w\}_{w \in X}$ in m_A^1 are formed by encrypting $x = 0^{n_x}$. Since G_2 doesn't use witnesses in this encryption in $\mathsf{CKEM}_{\mathsf{LA}}$, it follows that $G_3 \approx G_2$ by the reduction to CCA security of encryption cPKE. (Note that this reduction needs to decrypt the ciphertexts sent by Adv in m_B^2, but that these ciphertexts pertain to different labels than those in m_A^1, as well as the ciphertexts pertaining to sessions with other sid's, but these ciphertexts will also have different labels.)

Note that game G_3 proceeds like the simulator $\mathcal{A}^*$ interacting with $\mathsf{F}_{\mathsf{C}(f|g)}$ and ideal-world honest A who receives $(\mathsf{Input1}, B, x^*, \mathsf{sid})$ from $\mathcal{Z}$ for $x^* \in \{0,1\}^{n_x}$.

The difference between G_3 and $\mathcal{A}^*$ are only syntactic: G_3 evaluates $g(x^*, y)$ and $f(x^*, y)$ locally (setting y to $\perp$ if $u = 0$ beforehand), while in the simulation $\mathcal{A}^*$ performs the same $y := 0$ overwrite if $u = 0$, sends y to $\mathsf{F}_{\mathsf{C}(f|g)}$, and $\mathsf{F}_{\mathsf{C}(f|g)}$ evaluates both functions and sends $v = f_v(x^*, y)$ to $\mathcal{A}^*$, but in the end both processes set pair z, v on inputs $(x^*, \{c_i\}, y, \{d_i\}, t, u)$ in the same way. Secondly, G_3, case $v \neq \perp$, sets all w_i's for $i \notin S$ as $(z|t_1|...|t_{n_v})$ for t_i's which encode v, but this means that message m_A^2 in G_3 is set to a random string if $v = \perp$ and to encryption under K_A of $\max|t_1|...|t_{n_v}$ for $\tau = \mathsf{F}(\mathsf{K}'_B, 1)$ if $v \neq \perp$. Note that $\mathcal{A}^*$ sets m_A^2 in exactly the same way. Finally, game G_3 outputs z determined by $f|_g$ or $\perp$ based on the same condition, i.e. whether $\mathsf{m}_B^3 = \mathsf{F}(\mathsf{K}'_B, 2) \oplus \mathsf{F}(\mathsf{K}_A, 1)$, which decides whether $\mathsf{F}_{\mathsf{C}(f|g)}$ outputs z or $\perp$ to the dummy ideal-world player A in interaction with $\mathcal{A}^*$.

Case 2B, honest A on input $x^* = \perp$: Let G_0 be the interaction of $\mathcal{Z}$ and Adv in the real world where player A gets input $(\mathsf{Input1}, B, \perp, \mathsf{sid})$ from $\mathcal{Z}$. Note that in this case A is a random beacon, i.e. its messages $\mathsf{m}_A^1, \mathsf{m}_A^2$ are random strings of the appropriate length, and we will assume for notational convenience that A follows $\mathsf{TRec}^{\$(\tau)}$ in $\mathsf{CKEM}_{\mathsf{LA}}$ and $\mathsf{Snd}^{\$(\tau)}$ in $\mathsf{CKEM}_{\mathsf{LB}}$. Also A always sends $\perp$ back to $\mathcal{Z}$ as its computation "output".

Let G_1 be a modification of G_0 in which A sets $x{:=}0^{n_x}$ and $c_i \leftarrow \{0,1\}^{n_z+n_v\tau}$ for all i and then forms ciphertexts $\{\mathsf{ct}^w\}_{w\in X}$ and $\{\mathsf{ct}_i^w\}_{i,w\in C}$ in m_A^1 by encrypting the bits of x and c_i's as in step (1) of $\mathcal{A}^*$. We have that $G_1 \approx G_0$ by the reduction to CCA security of encryption cPKE. (Note that this reduction will need to decrypt ciphertexts on sessions with other sid's, but these ciphertexts will pertain to different labels.)

Let G_2 be like G_1 but in $\mathsf{CKEM}_{\mathsf{LA}}$ it runs TRec (but G_2 ignores TRec's local output K'_B). $G_2 \approx G_1$ by covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LA}}$.

Let G_3 be a modification of G_2 which in step A2 uses K'_B output by TRec in $\mathsf{CKEM}_{\mathsf{LA}}$ to decrypt ct^{B}, $\{\mathsf{r}_i^{\mathsf{gc}}\}_{i\in S}$, $\{\mathsf{gc}_i, \mathsf{ks}_i^B, \mathsf{otr}_i\}_{i\notin S}$ from m_B^2, and completes x_B by computing $(\mathsf{gc}_i, \{k_i^{w,b}\}) \leftarrow \mathsf{GCgen}_{f,g}(\mathsf{r}_i^{\mathsf{gc}})$ for $i \in S$. This creates no difference in Adv's view.

Let G_4 replace TRec with Rec in $TCKEMCorA$, and uses K'_B it outputs in step A2. $G_4 \approx G_3$ by covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LA}}$.

Let G_5 be like G_4 but in $\mathsf{CKEM}_{\mathsf{LB}}$ it runs $\mathsf{Snd}(\pi, (\mathsf{x}_B, \ell_B))$ on statement x_B computed in step A2. $G_5 \approx G_4$ by covert zero-knowledge of $\mathsf{CKEM}_{\mathsf{LB}}$.

Note that G_5 proceeds like the simulator $\mathcal{A}^*$ interacting with $\mathsf{F}_{\mathsf{C}(f|g)}$ and ideal-world honest A who receives $(\mathsf{Input1}, B, \perp, \mathsf{sid})$ from $\mathcal{Z}$: Game G_5 is like $\mathcal{A}^*$ simplified by the assumption that $\mathsf{F}_{\mathsf{C}(f|g)}$ returns $(\mathsf{Output}, v, \mathsf{sid})$ to $\mathcal{A}^*$ for $v = \perp$, in which case m_A^3 is set as random. However, this is indeed the case for A's session which was started by $\mathcal{Z}$ on input $x^* = \perp$. Also, note that in G_5 player A always outputs $\perp$, but this is also the output of the ideal-world player A in interaction with $\mathsf{F}_{\mathsf{C}(f|g)}$ if $x^* = \perp$.

References

1. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, pages 119–135, 2001.
2. F. Benhamouda, G. Couteau, D. Pointcheval, and H. Wee. Implicit zero-knowledge arguments and applications to the malicious setting. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 107–129, 2015.
3. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–, Washington, DC, USA, 2001. IEEE Computer Society.
4. N. Chandran, V. Goyal, R. Ostrovsky, and A. Sahai. Covert multi-party computation. In *FOCS*, pages 238–248, 2007.
5. C. Cho, D. Dachman-Soled, and S. Jarecki. Efficient concurrent covert computation of string equality and set intersection. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 164–179, 2016.
6. R. Cramer and V. Shoup. Universal hash proofs and and a paradigm for adaptive chosen ciphertext secure public-key encryption. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(072), 2001.
7. G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 74–89, 1999.
8. V. Goyal and A. Jain. On the round complexity of covert computation. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 191–200, New York, NY, USA, 2010. ACM.
9. N. J. Hopper, L. von Ahn, and J. Langford. Provably secure steganography. *IEEE Trans. Computers*, 58(5):662–676, 2009.
10. S. Jarecki. Practical covert authentication. In H. Krawczyk, editor, *Public-Key Cryptography PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 611–629. Springer Berlin Heidelberg, 2014.
11. S. Jarecki. Practical covert authentication. In *Public-Key Cryptography*, pages 611–629, 2014.
12. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Y. Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer Berlin Heidelberg, 2011.
13. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *J. Cryptology*, 28(2):312–350, 2015.
14. M. Naor and B. Pinkas. Computationally secure oblivious transfer. *J. Cryptology*, 18(1):1–35, 2005.
15. M. D. Raimondo and R. Gennaro. Provably secure threshold password-authenticated key exchange. *J. Comput. Syst. Sci.*, 72(6):978–1001, 2006.
16. L. von Ahn, N. Hopper, and J. Langford. Covert two-party computation. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 513–522, New York, NY, USA, 2005. ACM.

17. B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 114–127, 2005.

A Covertness of Cramer-Shoup Encryption

Proof of CCA-Covertness of Cramer-Shoup PKE. Recall the Cramer-Shoup encryption in Section 4, and recall its proof of CCA security given in [6]. Consider a (purely syntactic) restriction on an attacker $\mathcal{A}$ in the standard CCA-security game which specifies only one challenge message m^* and the encryption challenge pair is formed as $m_1 \leftarrow m^*$ and $m_0 \leftarrow G$. Recall the CCA-security proof of [6]: It shows a simulator $\mathcal{S}$ which on input a four-tuple $T = (g_1, g_2, u_1, u_2)$ of G elements interacts with attacker $\mathcal{A}$ in such a way that if $T \leftarrow \mathsf{DH}$, i.e. if (g_1, g_2, u_1, u_2) is a random Diffie-Hellman tuple, then for any choice of challengers bit b, $\mathcal{A}$'s view of real execution (denoted $\mathsf{R}(b)$) is statistically indistinguishable from $\mathcal{A}$'s view of an interaction with $\mathcal{S}$ (denoted $\mathcal{S}(b,T)$). Simulator $\mathcal{S}$ on input (b,T) for $T = (g_1, g_2, u_1, u_2)$, uses a different representation of the private key by picking $\mathsf{sk}' = (x_1, x_2, y_1, y_2, z_1, z_2) \leftarrow \mathbb{Z}_p^6$ and setting $(c, d, h) \leftarrow (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, g_1^{z_1} g_2^{z_2})$, (note that h is set differently from the real execution). Furthermore, $\mathcal{S}$ forms the challenge ciphertext $\mathsf{ct}^* = (u_1, u_2, e, v)$ using (u_1, u_2) from its input 4-tuple, and setting $e = u_1^{z_1} u_2^{z_2} m_b$ and $v = u_1^{x_1+y_1\xi} u_2^{x_2+y_2\xi}$ for $\xi = \mathsf{H}(\ell, u_1, u_2, e)$. Furthermore, $\mathcal{S}(b,T)$ decrypts $\mathcal{A}$'s ciphertext queries $(\bar{\mathsf{ct}}, \bar{\ell})$ for $\bar{\mathsf{ct}} = (\bar{u}_1, \bar{u}_2, \bar{e}, \bar{v})$ slightly differently than in $\mathsf{R}(b)$: It decrypts only if $\bar{v} = \bar{u}_1^{x_1+y_1\bar{\xi}} \bar{u}_2^{x_2+y_2\bar{\xi}}$ for $\bar{\xi} = \mathsf{H}(\bar{\ell}, \bar{u}_1, \bar{u}_2, \bar{e})$, as in $\mathsf{R}b$, but it forms the decrypted plaintext as $\bar{e} \cdot \bar{u}_1^{z_1} \bar{u}_2^{z_2}$. The proof in [6] shows that $\mathcal{A}$'s view of $\mathcal{S}(1,T)$ and $\mathcal{S}(0,T)$ are statistically close if $T \leftarrow G^4$. Moreover DDH implies that $\mathcal{A}$'s view of $\mathcal{S}(b,T)$ for $T \leftarrow \mathsf{DH}$ is indistinguishale from $\mathcal{A}$'s view of $\mathcal{S}(b,T)$ for $T \leftarrow G^4$.

In particular it follows that $\mathcal{A}$'s view of $\mathsf{R}(1)$, where $\mathsf{ct}^* = \mathsf{E}(\mathsf{pk}, m^*, \ell^*)$, is indistinguishable from $\mathcal{A}$'s view of $\mathcal{S}(0,T)$ for $T \leftarrow G^4$. Note that if $b = 0$ then $\mathcal{S}$ can pick $e \leftarrow G$ and the view does not change because $m_0 \leftarrow G$ in the original game. Consider Sim_1 which picks $g_1, u_1, u_2 \leftarrow G^3$ and $w \leftarrow \mathbb{Z}_p$, sets $g_2 \leftarrow (g_1)^w$, and then runs $\mathcal{S}(g_1, g_2, u_1, u_2)$. Clearly, the view of $\mathcal{S}_1$ is identical to the view of $\mathcal{S}(b = 0, T)$ for $T \leftarrow G^4$. Consider $\mathcal{S}_2$ which proceeds as $\mathcal{S}_1$ except the decryption oracle is modified to decrypt $(\bar{\mathsf{ct}}, \bar{\ell})$ query, for $\bar{c} = (\bar{u}_1, \bar{u}_2, \bar{e}, \bar{v})$, only if (1) $\bar{u}_2 = (\bar{u}_1)^w$ and (2) $\bar{v} = \bar{u}_1^{x_1+y_1\bar{\xi}} \bar{u}_2^{x_2+y_2\bar{\xi}}$ for $\bar{\xi} = \mathsf{H}(\bar{\ell}, \bar{u}_1, \bar{u}_2, \bar{e})$. The CCA-security proof in [6] uses a crucial technical lemma that, except for a negligible probability, the simulator $\mathcal{S}$ (as well as the real decryptor) rejects a decryption query unless $(g_1, g_2, \bar{u}_1, \bar{u}_2) \in \mathsf{DH}$. It follows that view presented by $\mathcal{S}_2$ is statistically close to that of $\mathcal{S}_1$, and hence also indistinguishable from the view of the real interaction on $\mathcal{A}$'s challenge plaintext m^*.

Note that values (u_1, u_2, e) in the challenge ciphertext ct^* in produced by $\mathcal{S}_2$ are three random elements of G. To see that the value v in ct^* is also random in G, examine the information which an all-powerful observer (who can compute discrete logs) sees about values (x_1, x_2, y_1, y_2) used by $\mathcal{S}_2$. Let r_1, r_2 be such that

that $(u_1, u_2) = (g_1^{r_1}, g_2^{r_2})$. Note that values (g_1, g_2, u_1, u_2) determine (w, r_1, r_2), values c, d detemine $\bar{x} = x_1 + wx_2$ and $\bar{y} = y_1 + wy_2$, value v determines $\alpha = r_1(x_1 + \xi y_1) + wr_2(x_2 + \xi y_2)$, and the values which the decryption oracle uses are w and (since $\bar{u}_2 = (\bar{u}_2)^w$) $\beta = (x_1 + \bar{\xi} y_1) + w(x_2 + \bar{\xi} y_2)$. However, note that $\beta = (x_1 + wx_2) + \bar{\xi}(y_1 + wy_2) = \bar{x} + \bar{\xi}\bar{y}$, which means that β does not reveal any more information than $\bar{x}, \bar{y}$. Note that $(\bar{x}, \bar{y}, \alpha)$ are computed as follows:

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ \alpha \end{bmatrix} = \begin{bmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1 & wr_2 & r_1\xi & wr_2\xi \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{bmatrix}$$

If $r_1 \neq r_2$ then the rows in the above matrix are linearly independent, and in that case $(\bar{x}, \bar{y}, \alpha)$ are uniform in $\mathbb{Z}_p^3$ for $(x_1, x_2, y_1, y_2) \leftarrow \mathbb{Z}_p^4$. It follows that ct^* produced by $\mathcal{S}_2$ is statitistically close to uniform in $\mathbb{Z}_p^4$ (over the choice of r_1, r_2 determined by u_1, u_2).

Now we can "move backwards" and modify simulation $\mathcal{S}_2$ so that it looks like the real execution except $\mathsf{ct}^* \leftarrow \mathbb{Z}_p^4$. Consider $\mathcal{S}_3$ which chooses $z \leftarrow \mathbb{Z}_p$ instead of (z_1, z_2) sets $h = g_1^z$ and decrypts as $\bar{e}/(\bar{u}_1)^z$. By equation (1) in the decryption test, and since $z = z_1 + wz_2$, we can see that $\mathcal{S}_3$ presents an identical view to $\mathcal{S}_2$. Finally, consider $\mathcal{S}_4$ which picks $(g_1, g_2, u_1, u_2, e, v)$ at random in G^6, and therefore does not know w, and uses only condition (2) above in the decryption test, as in the real Cramer-Shoup decryption. By the same technical lemma above the view produced by $\mathcal{S}_4$ is statistically close to that of $\mathcal{S}_3$. Since $\mathcal{S}_4$ now acts like the CCA challenger except the ciphertext challenge ct^* is sampled uniformly in G^4, this concludes the proof.

B Linear Map Image Languages for Covert 2PC.

We list the LMI languages used in the covert 2PC protocol of Section 7, we explain why these languages are in the LMI class, specifying the mapping between the language instance x and the (C, M) pair which defines the instance of LMI. Let $(\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ be the CCA-covert Cramer-Shoup PKE. All languages below are implicitely parametrized by the public key pk output by $\mathsf{Kg}(1^\tau)$, which also specifies the prime-order group setting (g, G, p). (Formally, the public key pk will be part of the language statement in each language below.) For each language below, it will be the case that the decryption key sk generated together with pk by $\mathsf{Kg}(1^\tau)$ is a trapdoor that allows for efficient verification of language membership.

In section B.1 we show the arguments that each language L below is *matrix-map receiver covert relative* $(\mathsf{Kg}, \mathsf{D}, \theta_{\mathsf{L}})$ for θ_{L} which restricts the adversary from decrypting ciphertexts whose labels are the same as the ciphertexts pertaining to the language statement.

Encryption Correctness. Let $\mathsf{Le}(\mathsf{pk})$ contain correct (ciphertext, label, plaintext) tuples, i.e.

$$\mathsf{Le}(\mathsf{pk}) = \{(\mathsf{ct}, \mathsf{m}, \ell) \text{ s.t. } \mathsf{ct} \in \mathsf{E}_{\mathsf{pk}}(\mathsf{m}, \ell)\}$$

Le statements can be mapped onto statements in $\mathsf{LMI}_{4,1}$, because $((\mathsf{ct}, \mathsf{m}, \ell), r) \in \mathcal{R}[\mathsf{Le}(\mathsf{pk})]$ for $\mathsf{pk} = ((g, G, p, \mathsf{H}), g_1, g_2, c, d, h)$ and $\mathsf{ct} = (u_1, u_2, e, v)$ if and only if $\mathsf{ct} = \mathsf{E}_{\mathsf{pk}}(\mathsf{m}; r)$ for some $r \in \mathbb{Z}_p$, which holds iff $C = M \cdot r$ for $C = (u_1, u_2, e/\mathsf{m}, v)$, $M = (g_1, g_2, h, cd^{\xi})$, and $\xi = \mathsf{H}(\ell, u_1, u_2, e)$.

Instance $(\mathsf{ct}, \mathsf{m}, \ell)$ of $\mathsf{Le}(\mathsf{pk})$ implies the following matrix M' (compare eq. 3 in Section 6) which contains 4-by-1 matrix M in its first row and vector C in the second row:

$$\mathbf{M}' = \left[\begin{array}{ccc|cccc|cc} 1 & 1 & 1 & g_1 & g_2 & h & (cd^{\xi}) & 1 & 1 \\ g' & 1 & 1 & u_1 & u_2 & e/\mathsf{m} & v & 1 & 1 \\ \hline 1 & g' & h' & 1 & 1 & 1 & 1 & 1 & 1 \\ g' & u' & e' & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & g' & h' \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & u'' & e'' \\ g' & 1 & 1 & 1 & 1 & 1 & 1 & g' & 1 \end{array}\right] \tag{5}$$

Encryption of a Bit. Another example is language $\mathsf{Lbit}(\mathsf{pk})$ of (shifted) encryptions of a bit, i.e.

$$\mathsf{Lbit}(\mathsf{pk}) = \{(\mathsf{ct}, \ell) \text{ s.t. } \exists b \ (\mathsf{ct}, g^b, \ell) \in \mathsf{Le}(\mathsf{pk}) \wedge b \in \{0, 1\}\}$$

This language can be expressed using only arithmetic constraints, i.e. without resorting to general disjunctions. The above constraints can be restated as $(u_1, u_2, e, v) = ((g_1)^r, (g_2)^r, h^r g^b, (cd^{\xi})^r)$ and $b(b-1) = 0$ for some $r, b \in \mathbb{Z}_p$. However, the second constraint can be expressed as the constraint that $1 = (u_1)^b (g_1)^{\lambda}$ and $1 = (e/g)^b h^{\lambda}$ for some $\lambda \in \mathbb{Z}_p$. This is because if $u_1 = (g_1)^r$ then $(u_1)^b (g_1)^{\lambda} = g_1^{rb+\lambda}$ so the first one of these constraints is equivalent to $rb + \lambda = 0$, and if $e = h^r g^b$ then $(e/g)^b h^{\lambda} = h^{rb+\lambda} g^{(b-1)b}$, hence the two constraints imply that $(b-1)b = 0$. Therefore $\mathsf{Lbit}(\mathsf{pk})$ statements can be mapped onto statements (C, M) of $\mathsf{LMI}_{6,3}$, where $C = (u_1, u_2, e, v, 1, 1)$, matrix M consists of columns $(g_1, 1, 1)$, $(g_2, 1, 1)$, $(h, g, 1)$, $(cd^{\xi}, 1, 1)$, $(1, u_1, g_1)$, and $(1, e/g, h)$, where $\xi = \mathsf{H}(\ell, u_1, u_2, e)$, with witness set as $w = (r, b, \lambda)$ for $\lambda = -rb$.

Instance (ct, ℓ) of $\mathsf{Lbit}(\mathsf{pk})$ therefore implies the following matrix M' (compare eq. 3 in Section 6) which contains 6-by-3 matrix M in rows 1-3 and vector C in row 4:

$$\mathbf{M}' = \left[\begin{array}{ccc|cccccc|cc} 1 & 1 & 1 & g_1 & g_2 & h & (cd^{\xi}) & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & g & 1 & u_1 & e/g & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & h & 1 & 1 \\ g' & 1 & 1 & u_1 & u_2 & e & v & 1 & 1 & 1 & 1 \\ \hline 1 & g' & h' & 1 & & & \ldots & & 1 & 1 & 1 \\ g' & u' & e' & 1 & & & \ldots & & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & & & \ldots & & 1 & g' & h' \\ 1 & 1 & 1 & 1 & & & \ldots & & 1 & u'' & e'' \\ g' & 1 & 1 & 1 & & & \ldots & & 1 & g' & 1 \end{array}\right] \tag{6}$$

Disjunction of Encryptions. In the covert 2PC protocol of Section 7 we consider language $\mathsf{Ldis}(\mathsf{pk})$ of tuples $(\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1)$ s.t. (1) ct encrypts bit b and (2) either ck_0 encrypts g^{m} and $b = 0$ or ck_1 encrypts g^{m} and $b = 1$.

$$\mathsf{Ldis}(\mathsf{pk}) = \{(\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell, \ell_0, \ell_1) \text{ s.t. } \exists b\ (\mathsf{ct}, b, \ell), (\mathsf{ck}_b, g^{\mathsf{m}}, \ell_b) \in \mathsf{Le}(\mathsf{pk})\}$$

One can define this language using a disjunction but it can also be expressed using a set of arithmetic constraints. First we have the encryption constraints, i.e. that there exists $b, \mathsf{m}_0, \mathsf{m}_1, r, r_0, r_1$ s.t.

$$\begin{aligned}
\mathsf{ct} = \ & (u_1, u_2, e, v) \ = ((g_1)^r, (g_2)^r, h^r g^b, (cd^{\xi})^r) \\
\mathsf{ck}_0 = & (u_1^0, u_2^0, e^0, v^0) = ((g_1)^{r_0}, (g_2)^{r_0}, h^{r_0} g^{m_0}, (cd^{\xi_0})^{r_0}) \\
\mathsf{ck}_1 = & (u_1^1, u_2^1, e^1, v^1) = ((g_1)^{r_1}, (g_2)^{r_1}, h^{r_1} g^{m_1}, (cd^{\xi_1})^{r_1})
\end{aligned}$$

for appropriately computed hash values ξ, ξ_0, ξ_1. Secondly, we have a constraint that $0 = b(b-1)$, which can be expressed as above, with constraints that $1 = (u_1)^b (g_1)^{\lambda}$ and $1 = (e/g)^b h^{\lambda}$ for some λ. Third, we have a constraint that there exists m' s.t. $m_0 = m - b \cdot (m - m')$ and $m_1 = m' + b \cdot (m - m')$, which guarantees that $m = m_b$. This can be re-stated that there exist s, δ, m' s.t. $s = b \cdot \delta$, $\delta = m - m'$, $m_0 = m - s$, and $m_1 = m' + s$. The last three constraints can be expressed as $g^m = g^{\delta} g^{m'}$, $g^m = g^{m_0} g^s$, and $1 = g^{m_1} (g^{-1})^{m'} (g^{-1})^s$, while the multiplicative constraint $s = b \cdot \delta$ can be expressed using the same indirection as in the case of constraint $0 = b(b-1)$, namely that $1 = e^{\delta} h^{\gamma} (g^{-1})^s$ and $1 = (u_1)^{\delta} (g_1)^{\gamma}$ for some γ. Summing up we have 19 linear constraints on a witness containing 11 values in $\mathbb{Z}_p$.

Specifically, instance $(\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell, \ell_0, \ell_1)$ of $\mathsf{Ldis}(\mathsf{pk})$ gives the following matrix M' (compare eq. 3 in Section 6) with 19-by-11 matrix M in rows 1-11 and vector C in row 8. To reduce space let $f = cd^{\xi}$, $f_0 = cd^{\xi_0}$, $f_1 = cd^{\xi_1}$:

$$\left[\begin{array}{ccc|cc|ccc|cc|cccc|cccc|cccc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & g_2 & h & f & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & u_1 & e/g & 1 & 1 & g & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & h & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & g_2 & h & f_0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & g & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & g_2 & h & f_1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & g^{-1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & g & 1 & g & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & g^{-1} & g & g & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & u_1 & e & 1 & 1 & g & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & g_1 & h & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
g' & 1 & 1 & 1 & 1 & 1 & g^m & g^m & 1 & 1 & u_1 & u_2 & e & v & u_1^0 & u_2^0 & e^0 & v^0 & u_1^1 & u_2^1 & e^1 & v^1 & 1 & 1 \\
\hline\hline
1 & g' & h' & 1 & & & & & & & & & & & & & & & & & & 1 & 1 & 1 \\
\hline
g' & u' & e' & 1 & & & & & & & & & & & & & & & & & & 1 & 1 & 1 \\
\hline\hline
1 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & & 1 & g' & h' \\
\hline
1 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & & 1 & u'' & e'' \\
\hline
g' & 1 & 1 & 1 & & & & & & & & & & & & & & & & & & 1 & g' & 1
\end{array}\right] \tag{7}$$

Correct OT Reponse. Another LMI-type language in covert 2PC of Section 7 is $\mathsf{Lotr}(\mathsf{pk})$, which involves verification that the response message OTrsp in the OT of Aiello et al. [1] (see Section 4) was computed on $\mathsf{m}_0, \mathsf{m}_1$ committed in $\mathsf{ck}_0, \mathsf{ck}_1$, i.e. that (1) $\mathsf{otr} = (s_0, t_0, s_1, t_1)$ where $(s_0, t_0) = (g_1^{\alpha_0} h^{\beta_0}, {u_1}^{\alpha_0}(e)^{\beta_0}\mathsf{m}_0)$ $(s_1, t_1) = (g_1^{\alpha_1} h^{\beta_1}, {u_1}^{\alpha_1}(e/g)^{\beta_1}\mathsf{m}_1)$ for some $(\alpha_0, \beta_0, \alpha_1, \beta_1) \in \mathbb{Z}_p^4$, where the receiver's OT message was $\mathsf{ct} = (u_1, u_2, e, v)$ for some u_2, v; and (2) that $\mathsf{m}_0, \mathsf{m}_1$ in the two equations above are encrypted respectively in $\mathsf{ck}_0, \mathsf{ck}_1$. In other words:

$$\begin{aligned}\mathsf{Lotr}(\mathsf{pk}) \;=\; \{\; &(\mathsf{otr}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell_0, \ell_1) \quad \text{s.t.} \quad \exists \mathsf{m}_0, \mathsf{m}_1, \mathsf{r} \\ &(\mathsf{ck}_0, \mathsf{m}_0, \ell_0) \in \mathsf{Le}(\mathsf{pk}) \wedge (\mathsf{ck}_1, \mathsf{m}_1, \ell_1) \in \mathsf{Le}(\mathsf{pk}) \wedge \mathsf{otr} = \mathsf{OTrsp}_{\mathsf{pk}}(\mathsf{ct}, \mathsf{m}_0, \mathsf{m}_1; \mathsf{r}) \;\}\end{aligned}$$

If ciphertexts $\mathsf{ck}_0, \mathsf{ck}_1$ are formed using the basic, i.e. *not* "shifted", version of the encryption then the plaintexts m_i (for $i = 0, 1$) is in the base in both t_i and in the e^i component of $\mathsf{ck}_i = (u_1^i, u_2^i, e^i, v^i)$, so we can cancel these plaintexts out by replacing (for $i = 0, 1$) constraints $e^i = h^r \mathsf{m}_i$ implied by $(\mathsf{ck}_i, \mathsf{m}_i, \ell_i) \in \mathsf{Le}(\mathsf{pk})$ with constraints $t_i/e_i = {u_1}^{\alpha_i}(e/g^i)^{\beta_i}(h^{-1})^r$. This results in 10 linear constraints with 6 variables, which means that statements in $\mathsf{Lotr}(\mathsf{pk})$ can be mapped onto statements in $\mathsf{LMI}_{10,6}$.

Instance $(\mathsf{otr}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell_0, \ell_1)$ of $\mathsf{Lotr}(\mathsf{pk})$ results in the following matrix M' (compare eq. 3 in Section 6) which contains 10-by-6 matrix M in rows 1-6 and vector C in row 7, where $\xi_i = \mathsf{H}(\ell^i, u_1^i, u_2^i, e^i)$ for $i = 0, 1$:

$$\mathbf{M}' = \left[\begin{array}{ccc|ccc|ccc|cccc|cc}
1 & 1 & 1 & g_1 & g_2 & (cd^{\xi_0}) & 1 & 1 & 1 & 1 & h^{-1} & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & g_1 & g_2 & (cd^{\xi_1}) & 1 & 1 & 1 & h^{-1} & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & u_1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & h & e & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & g_1 & u_1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & h & e/g & 1 & 1 \\
g' & 1 & 1 & u_1^0 & u_2^0 & v^0 & u_1^1 & u_2^1 & v^1 & s_0 & t_0/e^0 & s_1 & t_1/e^1 & 1 & 1 \\
\hline
1 & g' & h' & 1 & & & & \ldots & & & & & 1 & 1 & 1 \\
\hline
g' & u' & e' & 1 & & & & \ldots & & & & & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & & & & \ldots & & & & & 1 & g' & h' \\
\hline
1 & 1 & 1 & 1 & & & & \ldots & & & & & 1 & u'' & e'' \\
\hline
g' & 1 & 1 & 1 & & & & \ldots & & & & & 1 & g' & 1
\end{array}\right] \tag{8}$$

B.1 Proofs of matrix-map receiver covertness

We show that languages $\mathsf{Lbit}(\mathsf{pk})$, $\mathsf{Ldis}(\mathsf{pk})$, and $\mathsf{Lotr}(\mathsf{pk})$ above are *matrix-map receiver covert relative* $(\mathsf{Kg}, \mathsf{D}, \theta)$ for condition θ that restricts decryption of ciphertexts whose labels are the same as those in the language statement:

$\theta_{\mathsf{Lbit}}(x, (\mathsf{ct}', \ell'))$ returns 1 for $x = (\mathsf{ct}, \ell)$ s.t. $\ell' = \ell$;
$\theta_{\mathsf{Ldis}}(x, (\mathsf{ct}', \ell'))$ returns 1 for $x = (\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell, \ell_0, \ell_1)$ s.t. $\ell' \in \{\ell, \ell_0, \ell_1\}$;
$\theta_{\mathsf{Lotr}}(x, (\mathsf{ct}', \ell'))$ returns 1 for $x = (\mathsf{otr}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell_0, \ell_1)$ s.t. $\ell' \in \{\ell_0, \ell_1\}$;

Lemma 4. *Under the DDH assumption each language* L *among* Lbit, Ldis, *and* Lotr *is matrix-map receiver covert relative to* $(\mathsf{Kg}, \mathsf{D}, \theta_{\mathsf{L}})$.

Proof. As in the proof of matrix-map receiver covertness of $\mathsf{Le}(\mathsf{pk})$ (lemma 3, section 6.2) we will use a technical property which is used in the proof of CCA-covertness (and security) of Cramer-Shoup encryption. Namely, assuming DDH and collision-resistance of H, tuple ct^* formed as $(g_1, g_2, h, cd^{\xi})^s$, for random $s \leftarrow \mathbb{Z}_p$ and $\xi = \mathsf{H}(\ell, z)$ for any string z, is inditsinguishable from a random tuple in G^4, even on access to the oracle $\mathsf{D}_{\mathsf{sk}}(\cdot,\cdot)$ which decrypts ciphertext,label pairs (ct', ℓ') as long as $\ell' \neq \ell$.

(1) In the case of Lbit, consider statement $x = (ct, \ell)$. Condition $\theta_{\mathsf{Lbit}}(x, \cdot)$ allows decryption of any $(\mathsf{ct}', \ell'))$ as long as $\ell' \neq \ell$. See matrix M in equation (6). Value $r = s \cdot M$ for $s = (s_1, s_2, s_3)$ random in $\mathbb{Z}_p^3$ can be written as $r = (g_1, g_2, h, cd^{\xi}, 1, 1)^{s_1} \cdot (r_2)^{s_2} \cdot (1, 1, 1, 1, g_1, h)^{s_3}$ where r_2 is the second row of M. Since $\xi = \mathsf{H}(\ell, ...)$, by the technical lemma above r is indistinguishable from a modification which replaces the first component by $(b_1, b_2, b_3, b_4, 1, 1)$ for random (b_1, b_2, b_3, b_4) in G^4. By the same token the view remains indinstinguishable if we replace $(1, 1, 1, 1, g_1, h)^{s_3}$ by $(1, 1, 1, 1, b_5, b_6)$ for random b_5, b_6 in G^2. (The last transformation holds by the same technical property because g_1, h is a selection of two elements in tuple (g_1, g_2, h, cd^{ξ}).) At this point r is uniform in G^6, which proves the lemma.

(2) In the case of Ldis, consider statement $x = (\mathsf{m}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell, \ell_0, \ell_1)$. Condition $\theta_{\mathsf{Ldis}}(x, \cdot)$ allows decryption of any $(\mathsf{ct}', \ell'))$ as long as $\ell' \notin \{\ell, \ell_0, \ell_1\}$. See matrix M in equation (7), and let $r = s \cdot M$ for $s \in \mathbb{Z}_p^{11}$. Note that row one contains tuple (g_1, g_2, h, cd^{ξ}) in columns 8-11, row four contains tuple $(g_1, g_2, h, cd^{\xi_0})$ in columns 12-15, and row six contains tuple $(g_1, g_2, h, cd^{\xi_1})$ in columns 16-19. (Note also that all these rows contain no other non-one elements.) Since $\xi = \mathsf{H}(\ell, ...)$ $\xi_0 = \mathsf{H}(\ell_0, ...)$, and $\xi_1 = \mathsf{H}(\ell_1, ...)$, and the decryption oracle does not decrypt ciphertexts accompanied by any of these three labels, the technical lemma above applies to all three tuples, which means that elements $r_{[8,19]}$ in r can be replaced by 12 random group elements, given random $s_{1,4,6}$ in $\mathbb{Z}_p^3$. It remains to argue that $r_{[1,7]}$ is indistinguishable from a random tuple in G^7 over random $s_{2,3,5,[7-11]}$ in $\mathbb{Z}_p^8$. Note that row 11 contains (g_1, h) in columns 1-2, so a similar argument as used for Lbit in point (1) above implies that, given random s_{11}, elements $r_{1,2}$ can be replaced by random elements of G. Similarly row 3 contains (g_1, h) in columns 6-7, hence by the same argument elements $r_{6,7}$ can also be replaced by random in G. Finally, we can "use" elements g on the diagonal in rows 8-10 and columns 3-5 to see that for random $s_{8,9,10} \leftarrow \mathbb{Z}_p^3$, values $r_{3,4,5}$ are random in G. At this point r is uniform in G^{11}, which proves the lemma.

(3) In the case of Lotr, consider statement $x = (\mathsf{otr}, \mathsf{ct}, \mathsf{ck}_0, \mathsf{ck}_1, \ell_0, \ell_1)$. Condition $\theta_{\mathsf{Lotr}}(x, \cdot)$ allows decryption of any $(\mathsf{ct}', \ell'))$ as long as $\ell' \notin \{\ell_0, \ell_1\}$. See matrix M in equation (8), and let $r = s \cdot M$ for $s \in \mathbb{Z}_p^6$. Note that row one contains tuple (g_1, g_2, cd^{ξ_0}) in columns 1-3 and then h^{-1} in column 8, and row two contains tuple (g_1, g_2, cd^{ξ_1}) in columns 4-6 and then h^{-1} in column 10, for $\xi_0 = \mathsf{H}(\ell_0, ...)$ and $\xi_1 = \mathsf{H}(\ell_1, ...)$. Since this is equivalent to having tuples $(g_1, g_2, h, cd^{\xi_0})$ and $(g_1, g_2, h, cd^{\xi_1})$ in these columns, the technical lemma above applies to these tuples, which means that elements $r_{1,2,3,4,5,6,8,10}$ in r can be

replaced by 8 random group elements, given random $s_{1,2}$ in $\mathbb{Z}_p^2$. For columns 7 and 9 we can see by the presence of g_1 in rows respectively 3 and 5 that r_7 and r_9 are both random for random s_3, s_5, which shows that r is indistinguishable from random in G^{10}, and proves the lemma.