Efficient and Universally Composable Committed Oblivious Transfer and Applications

Juan A. Garay[1], Philip MacKenzie[1], and Ke Yang[2]

[1] Bell Labs – Lucent Technologies, Murray Hill, NJ, USA
{garay,philmac}@research.bell-labs.com
[2] Carnegie Mellon University, Pittsburgh, PA, USA
yangke@cs.cmu.edu

Abstract. Committed Oblivious Transfer (COT) is a useful cryptographic primitive that combines the functionalities of bit commitment and oblivious transfer. In this paper, we introduce an extended version of COT (ECOT) which additionally allows proofs of relations among committed bits, and we construct an efficient protocol that securely realizes an ECOT functionality in the universal-composability (UC) framework. Our construction is more efficient than previous (non-UC) constructions of COT, involving only a constant number of exponentiations and communication rounds. Using the ECOT functionality as a building block, we construct efficient UC protocols for general two-party and multi-party functionalities, each gate requiring a constant number of ECOT's.

1 Introduction

Committed Oblivious Transfer (COT) was introduced by Crépeau [17] (under the name "Verifiable Oblivious Transfer") as a natural combination of $\binom{2}{1}$-Oblivious Transfer [21] and Bit Commitment. At the start of the computation Alice is committed to bits a_0 and a_1 and Bob is committed to bit b; at the end Bob is committed to a_b and knows nothing about $a_{\bar{b}}$, while Alice learns nothing about b. One can see that this allows each party engaged in an oblivious transfer to be certain that the other party is performing the oblivious transfer operation on their declared inputs.[1] This has been shown to be useful in [18], who construct a protocol for general secure multi-party computation in the model of [28] using COT.

In this paper we show how to improve on previous constructions of COT in the areas of efficiency and universal composability. In terms of efficiency, the protocol we construct for COT uses only a constant number of exponentiations and communication rounds per transfer.[2] In contrast, the most efficient previously

[1] This contrasts with standard oblivious transfer, where some other method (perhaps another cryptographic building block, or verification at some higher layer protocol) is required to guarantee that parties are performing their part of the transfer on their declared inputs.

[2] Security is proved under some standard number theoretic assumptions, discussed later.

known construction of COT [18] uses $O(k)$ invocations of OT (thus implying at least the same number of public-key operations using known constructions) and bit commitments, and $O(k)$ rounds, for k a security parameter. Furthermore, we show that our protocol securely realizes an ideal COT functionality in the recently-proposed *universal composability* (UC) framework by Canetti [9], in the *common reference string* (CRS) model. Recall that to define security in this framework, one first specifies an "ideal functionality" describing the desired behavior of the protocol using a trusted party, and then one proves that a particular protocol operating in the real world "securely realizes" this ideal functionality, by showing that no "environment" would be able to distinguish (1) an adversary operating in the real world with parties running this protocol from (2) an "ideal adversary" operating in an ideal process consisting of dummy parties that simply call the ideal functionality. A main virtue of this framework is that the security of protocols thus defined is preserved under a general composition operation called "universal composition," which essentially means that protocols remain secure even when composed with other protocols that may be running concurrently in the same system. We give a more detailed review of the UC framework later in the paper. We note that a similar framework was independently proposed by Pfitzmann and Waidner [37,38]. Intuitively, these two frameworks are similar, although there are a number of technical differences. We choose to use the UC framework in this paper.[3]

Our protocol actually realizes an enhanced COT functionality, which we call *ECOT*, where in addition to oblivious transfer, one can prove certain relations among committed bits (in particular, among three bits). To demonstrate the usefulness of this functionality, we show that using ECOT as a building block, any well-formed two-party and multi-party functionality can be securely realized *efficiently* in the universal composability framework. Plugging in our protocol for realizing the ECOT into this construction, we have an efficient protocol for any well-formed two-party and multi-party functionality in the CRS model.

Canetti *et al.* [11] were the first to show that such functionalities are indeed realizable in this model, even under general cryptographic assumptions and regardless of the number of corrupted parties. More specifically, [11] follows the general "two-phase" approach of [27] of first designing a solution for the case of honest-but-curious parties, and then turning it into a solution for the actively malicious adversary, using a "compiler." The compiler adds a zero-knowledge proof to every message, proving that it is consistent with the history and the (committed) private input and the randomness. Notice that since the "consistency" proofs are for relations involving the execution of Turing machines, they are quite complex and it is unlikely that they admit efficient protocols; rather, proofs for general NP statements are used (which involve a reduction to an

[3] The ideal-process/real-world formulation of security and the simulator-based paradigm were initiated by Goldreich *et al.* [27]. From then on, there have been many definitions in this (now standard) paradigm, with emphasis on different aspects. For a number of examples, see Goldwasser and Levin [30], Micali and Rogaway [32], Beaver [2,3], and Canetti [8], for the formulations that preceed the UC framework.

NP-complete problem like Hamiltonian Cycle), making the compiler a major source of inefficiency. Canetti *et al.* make the protocol in [27] secure in the UC framework by replacing the basic primitives (namely, oblivious transfer, bit commitment, and zero-knowledge) with their universally composable counterparts. The resultant protocol becomes universally composable, but remains rather inefficient. In this paper we follow a different approach. By incorporating stronger security into the basic building block (i.e., ECOT), we are able to build protocols secure against adaptive and malicious adversaries *directly*, eliminating the need for a (normally inefficient) compiler. In this way, we are able to construct protocols that are efficient and at the same time enjoy a high level of security.

Our results. We now present a more detailed account of our results. We start by defining an ECOT functionality ($\mathcal{F}_{\mathrm{ECOT}}$), which, as mentioned above, additionally allows the sender to prove relations on three committed bits to the receiver. Then we construct a protocol to realize the ECOT functionality. The starting point for our construction is the standard Pedersen commitment scheme [35]. Then we build an OT protocol over these commitments that is loosely based on the (non-concurrent version of the) OT protocol of Garay and MacKenzie [23] (which in turn is based on the OT protocol of Bellare and Micali [4]). Zero-knowledge (ZK) proofs are required in this OT protocol, and thus we work in a hybrid model with ideal ZK functionalities. Naturally, the constructions for proving relations on three committed bits also use these ideal ZK functionalities. Finally, to construct efficient protocols that securely realize these ZK functionalities, we construct a special type of honest-verifier ZK protocol for each desired relations, and then we use a result by Garay *et al.* [24] that shows how to convert this special type of honest-verifier ZK protocol into a universally-composable ZK protocol. These results are presented in Section 3.

The ECOT functionality can be naturally extended into one that performs $\binom{4}{1}$-transfers (instead of $\binom{2}{1}$) and proves relations on four committed bits (as opposed to three). We call this extended functionality $\mathcal{F}^4_{\mathrm{ECOT}}$, and show how to construct it using the original $\mathcal{F}_{\mathrm{ECOT}}$ functionality as a building block. Equipped with $\mathcal{F}^4_{\mathrm{ECOT}}$, we then show how to securely realize a two-party functionality that we call *Joint Gate Evaluation* ($\mathcal{F}_{\mathrm{JGE}}$), which, as its name indicates, allows two parties to securely compute any Boolean function over two bits shared between them. Essentially, the protocol realizing this functionality uses a construction similar to that of [27] for the computation of the multiplication gate. However, distinctive features of the protocol are that it deals directly with adaptively malicious parties, and its efficiency: only a constant number of exponentiations and communication rounds per gate evaluation. Joint Gate Evaluation is presented in Section 4.

Finally, we use $\mathcal{F}_{\mathrm{JGE}}$ to securely realize — efficiently — any adaptively well-formed two-party and multi-party functionality, which is expressed by an arithmetic circuit over GF(2), in a universally-composable way. Again, since the realization is directly for the actively malicious adversary, and by means of an efficient building block, the overall computational complexity is a small constant times the number of gates in the representation of the functionality, and the number of rounds is a constant times the depth of the circuit. The treatment of two-party functionalities is presented in Section 5, while the case of multi-party functionalities, with the one-to-many extensions and realizations of the required building blocks, is discussed in Section 6. Putting everything together, we construct efficient and universally composable two-party and multi-party computation protocols that are secure against adaptive adversaries in the *erasing* model, where we allow parties to erase certain information reliably.

As a technical note, we use the gate-by-gate approach from [27], and make sure that each gate is computed efficiently. We do not use the "encrypted circuit" approach due to Yao [40], which yields constant-round protocols but is rather inefficient in terms of communication complexity, since one needs to prove in zero-knowledge that the encrypted circuit is correct and these proofs are unlikely to admit efficient protocols.

Related work. We already mentioned prior work on COT [17,18]. Although the protocols presented there are generic and hence may be implemented with or without computational assumptions (e.g., using primitives based on quantum channels), they are less efficient by at least a factor of k, where k is the security parameter, and furthermore, they are not universally composable. (As a side note, a "stand-alone" version of our ECOT protocol would be substantially simpler, in particular with respect to the implementation of the necessary ZK proofs.)

We can also compare the ECOT functionality to the functionalities defined in [11], who use a "two-phase" approach to construct universally composable two-party/multiple-party computation protocols. In the first phase, where they construct a protocol secure against semi-honest adversaries, an important tool is the OT functionality. In the second phase, where they exhibit a "compiler" that turns protocols in the first phase into ones secure against malicious adversaries, an important tool is the "commit-and-prove" functionality, which proves general NP statements. In some sense, the ECOT functionality may be viewed as a "combination" of the OT functionality and the commit-and-prove functionality. However, we stress that since ECOT only needs to prove very simple relations (among three bits), it can be realized more efficiently.[4]

Recently, Damgård and Nielsen [20] presented efficient universally composable multi-party computation protocols using a different approach. Their construction is based on an efficient MPC protocol by Cramer *et al.* [14], which in turn is based on threshold homomorphic cryptosystems. Compared to our result, the Damgård-Nielsen construction works in a slightly stronger model, namely the *public key infrastructure* (PKI) model, where a trusted party not only generates a common reference string (which contains the public keys of all the paries), but

[4] We note that a commitment functionality with the capability to perform proofs on committed bits was also proposed by Damgård and Nielsen [19], along with efficient protocols realizing it under some specific number-theoretic assumptions. However, it was not shown that their functionality could be used in constructing protocols for general secure multi-party computation, and more specifically, oblivious transfer.

also a private string for each party (as the party's secret key). On the other hand, their protocol is secure against adaptive adversaries in the so-called *non-erasing* model, where the parties are not allowed to erase any information, while our construction is secure in the erasing model only.

Due to space limitation, proofs are omitted from this extended abstract, but may be found in the full version of this paper [25].

2 Preliminaries and Definitions

All our results are in the *common reference string* (CRS) model, which assumes that there is a string uniformly generated from some distribution and is available to all parties at the start of a protocol. This is a generalization of the *public random string* model, where a uniform distribution over fixed-length bit strings is assumed.

For a distribution Δ, we say $a \in \Delta$ to denote any element that has non-zero probability in Δ, i.e., any element in the support of Δ. We say $a \overset{R}{\leftarrow} \Delta$ to denote a is randomly chosen according to distribution Δ. For a set S, we say $a \overset{R}{\leftarrow} S$ to denote that a is uniformly drawn from S.

Ω-protocols. We will use a special type of zero-knowledge protocols, namely, *Ω-protocols* [24], which are variants of the so-called Σ-protocols [15,13]. Very roughly speaking, Σ-protocols are three-round, public-coin, honest-verifier zero-knowledge protocols, and Ω-protocols are proof-of-knowledge Σ-protocols with a straight-line extractor. See [24,25] for a detailed description of Ω-protocols.

The universal composability framework. The universal composability framework was suggested by Canetti for defining the security and composition of protocols [9]. In this framework one first defines an "ideal functionality" of a protocol, and then proves that a particular implementation of this protocol operating in a given computational environment securely realizes this ideal functionality. The basic entities involved are n players $P_1, \ldots, P_n$, an adversary $\mathcal{A}$, and an environment $\mathcal{Z}$. The real execution of a protocol π, run by the players in the presence of $\mathcal{A}$ and an environment machine $\mathcal{Z}$, with input z, is modeled as a sequence of *activations* of the entities. The environment $\mathcal{Z}$ is activated first, generating in particular the inputs to the other players. Then the protocol proceeds by having $\mathcal{A}$ exchanging messages with the players and the environment. Finally, the environment outputs one bit, which is the output of the protocol.

The security of the protocols is defined by comparing the real execution of the protocol to an ideal process in which an additional entity, the ideal functionality $\mathcal{F}$, is introduced; essentially, $\mathcal{F}$ is an incorruptible trusted party that is programmed to produce the desired functionality of the given task. Let $\mathcal{S}$ denote the adversary in this idealized execution. The players are replaced by dummy players, who do not communicate with each other; whenever a dummy player is activated, its input is forwarded to $\mathcal{F}$ by $\mathcal{S}$, who can see the "public header" of the input.[5] As in the real-life execution, the output of the protocol execution is the one-bit output of $\mathcal{Z}$. Now a protocol π *securely realizes* an ideal functionality $\mathcal{F}$ if for any real-life adversary $\mathcal{A}$ there exists an ideal-execution adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and players running π in the real-life execution, or with $\mathcal{S}$ and $\mathcal{F}$ in the ideal execution. More precisely, if the two binary distribution ensembles, $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, describing $\mathcal{Z}$'s output after interacting with adversary $\mathcal{A}$ and players running protocol π (resp., adversary $\mathcal{S}$ and ideal functionality $\mathcal{F}$), are computationally indistinguishable (denoted $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$).

Protocols typically invoke other sub-protocols. In this framework the *hybrid model* is like a real-life execution, except that some invocations of the sub-protocols are replaced by the invocation of an instance of an ideal functionality $\mathcal{F}$; this is called the "$\mathcal{F}$-hybrid model." We are designing and analyzing protocols in the CRS model, and so they will be operating in the $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$-hybrid model, where $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$ is the functionality that chooses a string from distribution $\mathcal{D}_k$ and hands it to all parties. Further, we will consider the "multi-session extension of $\mathcal{F}$" of Canetti and Rabin [12], denoted $\hat{\mathcal{F}}$, which runs multiple copies of $\mathcal{F}$ by identifying each copy by a special *sub-session identifier.*

The definition of $\hat{\mathcal{F}}_{\text{ZK}}^{R}$, the multi-session extension of $\mathcal{F}_{\text{ZK}}^{R}$, is shown below. Note the two types of indices: the *sid*, which differentiates messages to $\hat{\mathcal{F}}_{\text{ZK}}^{R}$ from messages sent to other functionalities, and *ssid*, the sub-session identifier, which is unique per input message (or proof).

Functionality $\hat{\mathcal{F}}_{\text{ZK}}^{R}$

$\hat{\mathcal{F}}_{\text{ZK}}^{R}$ proceeds as follows, running with security parameter k, parties $P_1, \ldots, P_n$, and an adversary $\mathcal{S}$:

– Upon receiving $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ from P_i: If $R(x, w)$ then send $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ to P_j and $\mathcal{S}$ and halt. Otherwise, ignore.

Refer to [9,11] for further description of the UC framework.

3 Universally Composable Committed Oblivious Transfer

In this section we present the $\mathcal{F}_{\text{ECOT}}$ functionality, an extension of COT where in addition to the oblivious transfer, the sender can prove to the receiver (Boolean) relations among the committed bits. We will later use this functionality to implement an efficient *Joint Gate Evaluation* functionality, which in turn will enable efficient and universally composable multi-party computation. The functionality $\mathcal{F}_{\text{ECOT}}$ is shown below. Informally, a party P_i commits to a bit b by sending an ecot-commit message to the ideal functionality $\mathcal{F}_{\text{ECOT}}$, and P_i can later open this bit by sending an ecot-open message with appropriate commitment identifier

[5] This feature was added to the UC framework in [11].

(cid) value. For P_i to obliviously transfer a bit to P_j, P_i needs to commit two bits b_0 and b_1 and P_j needs to commit to one bit b_t; after sending an $\mathsf{ecot\text{-}transfer}$ to $\mathcal{F}_{\mathrm{ECOT}}$, the bit b_{b_t} is transferred to P_j and automatically committed by $\mathcal{F}_{\mathrm{ECOT}}$ on behalf of P_j. Meanwhile, P_i does not learn anything, except that a transfer took place. Furthermore, the functionality also allows a party P_i to prove to P_j that three bits b_0, b_1, and b_2 it committed to satisfy a particular binary relation by sending an $\mathsf{ecot\text{-}prove}$ message to $\mathcal{F}_{\mathrm{ECOT}}$.

As a convention, we use $\mathsf{op}_m^{(2)}$ to denote a function on two bits, where $m \in \{0,1\}^4$ is the string of bits of the Boolean function's truth table (output column). We also often identify m with the integer whose binary representation is m. (For example, $m = 1$ represents the AND function, whose truth table is 0001.)

As a technical note, we note that the Open phase is not strictly necessary since it can be simulated by the Prove phase. Take $\mathsf{op}_{0000}^{(2)}$ and $\mathsf{op}_{1111}^{(2)}$, which correspond to the all-zero and all-one functions. Then, by proving that $\mathsf{op}_{0000}^{(2)}(b_0, b_1) = b_2$ for arbitrary bits b_0 and b_1, one essentially opens bit b_2 to 0; similarly, by proving that $\mathsf{op}_{1111}^{(2)}(b_0, b_1) = b_2$, one opens b_2 to 1. We choose to include the Open phase in the functionality for clarity and efficiency (the Open phase can be realized more efficiently than the simulated Prove phase).

Functionality $\mathcal{F}_{\mathrm{ECOT}}$

$\mathcal{F}_{\mathrm{ECOT}}$ proceeds as follows, running with parties $P_1, ..., P_n$ and an adversary $\mathcal{S}$.

- **Commit phase:** When receiving from P_i a message $\langle \mathsf{ecot\text{-}commit}, sid, cid, P_j, b\rangle$, record $\langle cid, P_i, P_j, b\rangle$, send message $\langle \mathsf{ECOT\text{-}RECEIPT}, sid, cid, P_i, P_j\rangle$ to P_i, P_j and $\mathcal{S}$, and ignore all future messages of the form $\langle \mathsf{ecot\text{-}commit}, sid, cid, P_j, *\rangle$ from P_i and $\langle \mathsf{ecot\text{-}transfer}, sid, cid, *, *, *, P_i\rangle$ from P_j.
- **Prove phase:** When receiving from P_i a message $\langle \mathsf{ecot\text{-}prove}, sid, ssid, cid_0, cid_1, cid_2, P_j, m\rangle$, if the following three tuples, $\langle cid_0, P_i, P_j, b_0\rangle$, $\langle cid_1, P_i, P_j, b_1\rangle$, $\langle cid_2, P_i, P_j, b_2\rangle$, are all recorded, and $\mathsf{op}_m^{(2)}(b_0, b_1) = b_2$, then send message $\langle \mathsf{ECOT\text{-}PROOF}, sid, ssid, cid_0, cid_1, cid_2, P_i, m\rangle$ to P_j and $\mathcal{S}$; otherwise do nothing.
- **Transfer phase:** When receiving from P_i a message $\langle \mathsf{ecot\text{-}transfer}, sid, cid, cid_0, cid_1, tcid, P_j\rangle$, if the following three tuples $\langle cid_0, P_i, P_j, b_0\rangle$, $\langle cid_1, P_i, P_j, b_1\rangle$, and $\langle tcid, P_j, P_i, b_t\rangle$, are all recorded, send message $\langle \mathsf{ECOT\text{-}DATA}, sid, cid, P_i, P_j, cid_0, cid_1, tcid, b_{b_t}\rangle$ to P_j, record tuple $\langle cid, P_j, P_i, b_{b_t}\rangle$, and send message $\langle \mathsf{ECOT\text{-}RECEIPT}, sid, cid, P_i, P_j, cid_0, cid_1, tcid\rangle$ to P_i and $\mathcal{S}$, and ignore all future messages of the form $\langle \mathsf{ecot\text{-}commit}, sid, cid, P_i, *\rangle$ from P_i and $\langle \mathsf{ecot\text{-}transfer}, sid, cid, *, *, *, P_j\rangle$ from P_j. Otherwise, do nothing.
- **Open phase:** When receiving from P_i a message $\langle \mathsf{ecot\text{-}open}, sid, cid, P_i, P_j\rangle$, if the tuple $\langle cid, P_i, P_j, b\rangle$ is recorded, send message $\langle \mathsf{ECOT\text{-}DATA}, sid, cid, P_i, P_j, b\rangle$ to both $\mathcal{S}$ and P_j; otherwise, do nothing.

Before presenting a protocol that securely realizes $\mathcal{F}_{\mathrm{ECOT}}$, we first discuss some preliminary constructions that will be used as building blocks.

3.1 Building Blocks

In [24], Garay *et al.* introduced a technique to transform any Ω-protocol into a universally composable protocol by using a digital signature scheme that is existentially unforgeable against adaptive chosen-message attacks. Their transformation is efficient, if the digital signature scheme admits an efficient proof of knowledge protocol. In particular, they proved the following result.

Theorem 1 ([24]). *Under the strong RSA assumption or the DSA assumption, for every relation R that admits an Ω-protocol Π, there exists a three-round protocol $\mathsf{UC}[\Pi]$ that securely realizes the $\hat{\mathcal{F}}^{R}_{\mathrm{ZK}}$ ideal functionality in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model against adaptive adversaries, assuming erasing. Furthermore, the (additive) overhead of $\mathsf{UC}[\Pi]$ to Π is constant number of exponentiations plus the generation of a signature.*

See [25] for discussion on the Strong RSA assumption.

Drawing from standard techniques in the literature (e.g., [6,7,22,31,24]), we are able to construct efficient Ω-protocols for the following relations; by then "plugging" them into Theorem 1, we obtain efficient universally composable zero-knowledge protocols for these relations. Due to space limitations, the detailed construction of these Ω-protocols appears in the full version [25].

1. **"OR" of two discrete logs:**

$$R_{\mathsf{OR\text{-}DL}}((y_0, g_0, y_1, g_1), (x_0, x_1)) = R_{\mathsf{DL}}((y_0, g_0), x_0) \ \vee \ R_{\mathsf{DL}}((y_1, g_1), x_1)$$

2. **"OR"/"AND" relation of six discrete logs:**

$$\begin{aligned} &R_{\mathsf{OR\text{-}N\text{-}DL}}((y_0, y_1, y_2, y_3, y_4, y_5, g), (x_0, x_1, x_2, x_3, x_4, x_5)) = \\ &\quad ((R_{\mathsf{DL}}((y_0, g), x_0) \vee R_{\mathsf{DL}}((y_1, g), x_1)) \wedge R_{\mathsf{DL}}((y_2, g), x_2)) \ \vee \\ &\quad (R_{\mathsf{DL}}((y_3, g), x_3) \wedge R_{\mathsf{DL}}((y_4, g), x_4) \wedge R_{\mathsf{DL}}((y_5, g), x_5)) \end{aligned}$$

3. **Partial equality of representations:**

$$R_{\mathsf{PEREP}}((x_0, g_0, g_1, x_1, g_2, g_3), (\alpha_0, \alpha_1, \alpha_2))$$

4. **"OR" of partial equality of representations:**

$$\begin{aligned} R_{\mathsf{OR\text{-}PEREP}}((x_0, g_0, g_1, x_1, g_2, g_3, y_0, h_0, h_1, y_1, h_2, h_3), (\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1, \beta_2)) = \\ R_{\mathsf{PEREP}}((x_0, g_0, g_1, x_1, g_2, g_3), (\alpha_0, \alpha_1, \alpha_2)) \ \vee \\ R_{\mathsf{PEREP}}((y_0, h_0, h_1, y_1, h_2, h_3), (\beta_0, \beta_1, \beta_2)) \end{aligned}$$

3.2 The UCECOT Protocol

We now present UCECOT, a protocol that securely realizes the $\mathcal{F}_{\mathrm{ECOT}}$ ideal functionality in the $(\mathcal{F}_{\mathrm{CRS}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}DL}}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}N\text{-}DL}}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{PEREP}}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}PEREP}}})$-hybrid model, where the CRS consists of (p, q, g, h) such that q and p are primes satisfying $q|(p-1)$ and $g, h \in \mathbb{Z}_p^*$ are random elements satisfying $\mathsf{order}(g) = \mathsf{order}(h) = q$. p and q will also serve as the public parameters in the relations R_{DL}, R_{PEREP}, and their compositions.

We first describe the protocol.

Commit phase: On receiving private input $\langle \mathsf{ecot\text{-}commit}, sid, cid, P_j, b\rangle$, assuming that cid is not used before, party P_i picks a random $r \overset{R}{\leftarrow} \mathbb{Z}_q$, computes $B \leftarrow g^r \cdot h^b \mod p$, sends message $(\mathsf{ucecot\text{-}commit}, sid, cid, B)$ to party P_j, message $(\mathsf{zk\text{-}prover}, sid, cid, P_i, P_j, (B, g, B/h, g), (r, r))$ to $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}DL}}}$, and outputs $\langle \mathsf{ECOT\text{-}RECEIPT}, sid, cid, P_i, P_j\rangle$. After receiving the messages from P_i and $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}DL}}}$ respectively, P_j outputs $\langle \mathsf{ECOT\text{-}RECEIPT}, sid, cid, P_i, P_j\rangle$.

Essentially P_i sends a Pedersen commitment [35] of bit b to P_j and uses the $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}DL}}}$ ideal functionality to prove that he either knows the discrete log of B (in which case P_i is committing to bit 0) or the discrete log of B/h base g (in which case P_i is committing to bit 1).

Prove phase: Suppose P_i has committed bits b_0, b_1, and b_2 to P_j using cids cid_0, cid_1, and cid_2, respectively. Further assume that their corresponding Pedersen commitments are $B_0 = g^{r_0} \cdot h^{b_0}$, $B_1 = g^{r_1} \cdot h^{b_1}$, and $B_2 = g^{r_2} \cdot h^{b_2}$. Now, upon receiving private input $\langle \mathsf{ecot\text{-}prove}, sid, ssid, cid_0, cid_1, cid_2, P_j, m\rangle$, P_i is to prove to P_j that $\mathsf{op}_m^{(2)}(b_0, b_1) = b_2$, using sub-session id $ssid$. We first consider the situation where $m = 1110$, in which case $\mathsf{op}_m^{(2)}$ is the NAND operation. In this situation, P_i sends message $(\mathsf{ucecot\text{-}prove}, sid, ssid, cid_0, cid_1, cid_2, m)$ to P_j and sends message $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, (B_0, B_1, B_2/h, B_0/h, B_1/h, B_2, g), (r_0, r_1, r_2, r_0, r_1, r_2))$ to $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}N\text{-}DL}}}$. After receiving the corresponding message from $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}N\text{-}DL}}}$, P_j outputs $\langle \mathsf{ECOT\text{-}PROOF}, sid, ssid, cid_0, cid_1, cid_2, P_i, m\rangle$.

Intuitively, P_i is proving that $(((b_0 = 0) \vee (b_1 = 0)) \wedge (b_2 = 1)) \vee ((b_0 = 1) \wedge (b_1 = 1) \wedge (b_2 = 0))$.

In the case of any other binary operations $\mathsf{op}_m^{(2)}$, it can be written as a composition of NANDs and then proved step by step. P_i will need to commit to all the intermediate bits and prove each NAND operation is correct. For example, consider the case where $m = 0001$ is the AND operation. Notice that $x \wedge y = \overline{\overline{x \wedge y} \wedge \overline{x \wedge y}}$ Therefore, to prove that $b_2 = b_0 \wedge b_1$, P_i needs to commit to a new bit $b_3 = \overline{b_0 \wedge b_1}$ using the protocol in the Commit phase, and then prove that both $b_3 = \overline{b_0 \wedge b_1}$ and that $b_2 = \overline{b_3 \wedge b_3}$.

Transfer phase: Suppose P_i has committed bits b_0 and b_1, and P_j has committed bit b_t, using identifiers cid_0, cid_1, and $tcid$, respectively. Further assume that the corresponding Pedersen commitments are $B_0 = g^{r_0} \cdot h^{b_0}$, $B_1 = g^{r_1} \cdot h^{b_1}$, and $B_t = g^{r_t} \cdot h^{b_t}$. Now, upone receiving private input ecot-transfer, $sid, cid, cid_0, cid_1, tcid, P_j\rangle$, assuming that cid is not used before, P_i is to obliviously transfers bit b_{b_t} to P_j, using session id sid and the commitment id cid for the new bit b_{b_t}. Intuitively, P_i sends two Pedersen commitments, C_0 and C_1, where C_0 is a commitment to b_0 using base B_t, and C_1 is a commitment to b_1 using base B_t/h. It also sends A_0 and A_1 generated using the same randomness as C_0 and C_1. If $b_t = 0$, then P_j knows the discrete log of B_t and can check if C_0 is a commitment to zero or not, and if $b_t = 1$, then P_j knows the discrete log of B_t/h and can check if C_1 is a commitment to zero or not.

Now we proceed to the details. P_i randomly picks $a_0, a_1 \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $A_0 \leftarrow g^{a_0}$, $A_1 \leftarrow g^{a_1}$, $C_0 \leftarrow B_t^{a_0} \cdot h^{b_0}$, and $C_1 \leftarrow (B_t/h)^{a_1} \cdot h^{b_1}$. P_i then sends message (ucecot-transfer, $sid, cid, cid_0, cid_1, tcid, A_0, A_1, C_0, C_1$) to P_j and sends the following four messages to the ideal functionality $\hat{\mathcal{F}}_{\text{ZK}}^{R_{\text{PEREP}}}$.[6]

$$(\mathsf{zk\text{-}prover}, sid, cid \circ 00, P_i, P_j, (C_0, h, B_t, B_0, h, g), (b_0, a_0, r_0))$$
$$(\mathsf{zk\text{-}prover}, sid, cid \circ 01, P_i, P_j, (C_1, h, B_t/h, B_1, h, g), (b_1, a_1, r_1))$$
$$(\mathsf{zk\text{-}prover}, sid, cid \circ 10, P_i, P_j, (A_0, g, 1, C_0, B_t, h), (a_0, 0, b_0))$$
$$(\mathsf{zk\text{-}prover}, sid, cid \circ 11, P_i, P_j, (A_1, g, 1, C_1, B_t/h, h), (a_1, 0, b_1))$$

After this, P_i erases a_0 and a_1.

After receiving the message from P_i and four messages from the ideal functionality $\hat{\mathcal{F}}_{\text{ZK}}^{R_{\text{PEREP}}}$, P_j does the following (otherwise P_j aborts).

If $b_t = 0$, then check if $A_0^{r_t} = C_0 \bmod p$, and set $b \leftarrow 0$ if yes and $b \leftarrow 1$ otherwise; if $b_t = 1$, then check if $A_1^{r_t} = C_1 \bmod p$, and sets $b \leftarrow 0$ if yes and $b \leftarrow 1$ otherwise. Now b is the bit P_j receives.

Next, P_j picks a random $r \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $B \leftarrow g^r \cdot h^b \bmod p$, sends message (ecot-commit, sid, cid, B) to party P_i, sends message (zk-prover, $sid, cid, P_j, P_i, (C_0, h, A_0, B, h, g, C_1, h, A_1, B, h, g), (b, r_t, r,$
$b, r_t, r)$) to ideal functionality
$\hat{\mathcal{F}}_{\text{ZK}}^{R_{\text{OR-PEREP}}}$, and outputs $\langle$ECOT-DATA, $sid, cid, P_i, P_j, cid_0, cid_1, tcid, b\rangle$. Finally, after receiving messages from P_j and $\hat{\mathcal{F}}_{\text{ZK}}^{R_{\text{OR-PEREP}}}$, P_i outputs $\langle$ECOT-RECEIPT, $sid, cid, P_i, P_j, cid_0, cid_1, tcid\rangle$.

Open phase: Suppose P_i has committed a bit b to P_j using session id sid, and commitment id cid. Further assume that the commitment is $B = g^r \cdot h^b \bmod p$. Now upon receiving private input $\langle$ecot-open, $sid, cid, P_i, P_j\rangle$, P_i opens the bit b by sending message (ucecot-open, sid, cid, b, r) to P_j, who then verifies that $B = g^r \cdot h^b \bmod p$, and outputs (ECOT-DATA, sid, cid, P_i, P_j, b) if the verification is valid.

This is exactly the opening of a Pedersen commitment.

In the full version, we show:

[6] We assume that all the id's are binary strings, and we use "$a \circ b$" to indicate the concatenation of string a with string b.

Theorem 2. *Under the DDH assumption, protocol* UCECOT *securely realizes the* $\mathcal{F}_{\mathrm{ECOT}}$ *ideal functionality in the* $(\mathcal{F}_{\mathrm{CRS}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}DL}}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}N\text{-}DL}}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{PEREP}}}, \hat{\mathcal{F}}_{\mathrm{ZK}}^{R_{\mathsf{OR\text{-}PEREP}}})$*-hybrid model against adaptive, malicious adversaries, assuming erasing.*

4 Joint Gate Evaluation

In this section we show how to securely realize a two-party functionality that we call *Joint Gate Evaluation* ($\mathcal{F}_{\mathrm{JGE}}$) in the $\mathcal{F}_{\mathrm{ECOT}}$-hybrid model in the presence of a malicious, adaptive adversary. Informally, $\mathcal{F}_{\mathrm{JGE}}$ allows two parties to jointly evaluate any binary operation on two bits, and this will allow us to construct general two-party computation protocols on top of $\mathcal{F}_{\mathrm{JGE}}$. We first present the functionality, shown below.

Functionality $\mathcal{F}_{\mathrm{JGE}}$

$\mathcal{F}_{\mathrm{JGE}}$ proceeds as follows, running with parties $P_1, ..., P_n$, and adversary $\mathcal{S}$.

- **Commit phase:** When receiving from P_i a message $\langle \mathsf{commit}, sid, cid, P_j, b\rangle$, record $\langle cid, \{P_i, P_j\}, b\rangle$, send message $\langle \mathsf{RECEIPT}, sid, cid, P_i, P_j\rangle$ to P_i, P_j and $\mathcal{S}$, and ignore all further messages of the form $\langle \mathsf{commit}, sid, cid, x, *\rangle$ and $\langle \mathsf{eval}, sid, cid, *, *, x, *\rangle$ from P_j or P_j, where $x \in \{P_i, P_j\}$.
- **Evaluate phase:** When receiving from P_i a message $\langle \mathsf{eval}, sid, cid, cid_0, cid_1, P_j, m\rangle$, if both $\langle cid_0, \{P_i, P_j\}, b_0\rangle$ and $\langle cid_1, \{P_i, P_j\}, b_1\rangle$ are recorded, then compute $b = \mathsf{op}_m^{(2)}(b_0, b_1)$, record $\langle cid, \{P_i, P_j\}, b\rangle$, send message $\langle \mathsf{EVAL\text{-}RECEIPT}, sid, cid, cid_0, cid_1, P_i, P_j, m\rangle$ to P_i, P_j and $\mathcal{S}$, and ignore all further messages of the form $\langle \mathsf{commit}, sid, cid, x, *\rangle$ and $\langle \mathsf{eval}, sid, cid, *, *, x, *\rangle$ from P_i or P_j, where $x \in \{P_i, P_j\}$. Otherwise, do nothing.
- **Open phase:** When receiving from P_i a message $\langle \mathsf{open}, sid, cid, P_j\rangle$, if the tuple $\langle cid, \{P_i, P_j\}, b\rangle$ is recorded, then send message $\langle \mathsf{DATA}, sid, cid, P_i, P_j, b\rangle$ to P_j; otherwise, do nothing.

At a high level, the approach we will use to realize functionality $\mathcal{F}_{\mathrm{JGE}}$ is similar to that in [26,11]. In particular, each bit stored in $\mathcal{F}_{\mathrm{JGE}}$ will be XOR-shared by P_i and P_j, and each gate evaluation will be done by a $\binom{4}{1}$-oblivious transfer. However, our resulting construction is directly secure against a malicious, adaptive adversary, and therefore we do not need the "compiler" used in [26, 11]. This "direct" (as opposed to the "two-phase") approach makes our protocol much more efficient.

In particular, we will realize the $\mathcal{F}_{\mathrm{JGE}}$ functionality using a further generalization of the $\mathcal{F}_{\mathrm{ECOT}}$ functionality, which we call $\mathcal{F}_{\mathrm{ECOT}}^4$. The Commit and Open phases of $\mathcal{F}_{\mathrm{ECOT}}^4$ are identical to those of $\mathcal{F}_{\mathrm{ECOT}}$, but the Transfer phase performs a $\binom{4}{1}$-transfer (instead of $\binom{2}{1}$), while the Prove phase proves relations consisting of Boolean functions of *three bits* (as opposed to two).

The detailed descriptions of $\mathcal{F}^4_{\mathrm{ECOT}}$ and a protocol that securely realizes it in the $\mathcal{F}_{\mathrm{ECOT}}$-hybrid model appear in [25].

We now give a high-level idea of how the protocol will realize the $\mathcal{F}_{\mathrm{JGE}}$ functionality in the $\mathcal{F}^4_{\mathrm{ECOT}}$-hybrid model. In the protocol, each bit b of identifier *cid* stored in $\mathcal{F}_{\mathrm{JGE}}$ is shared between P_i and P_j additively. More precisely, P_i has a bit b_1 and P_j has a bit b_2 such that $b = b_1 \oplus b_2$. Furthermore, each of b_1 and b_2 is a random bit by itself. Both P_i and P_j will commit to their bits to each other using identifier *cid*. To open this bit to P_i, P_j opens its share, b_2, to P_i, who then computes $b = b_1 \oplus b_2$.

In order to evaluate $c = \mathsf{op}^{(2)}_m(a, b)$, suppose P_i holds a_1 and b_1 as shares of a and b, and P_j holds a_2 and b_2, respectively. Then, P_i generates a random bit $c_1 \overset{R}{\leftarrow} \{0, 1\}$ and computes four bits $o_{00}, o_{01}, o_{10}, o_{11}$, which are the "candidate bits" for c_2, P_j's share of bit c. Which bit is c_2 depends on P_j's shares a_2 and b_2. The actual bits are computed as in the table below.

(a_2, b_2)	P_j's output c_2
$(0, 0)$	$o_{00} = c_1 \oplus \mathsf{op}^{(2)}_m(a_1, b_1)$
$(0, 1)$	$o_{01} = c_1 \oplus \mathsf{op}^{(2)}_m(a_1, (b_1 \oplus 1))$
$(1, 0)$	$o_{10} = c_1 \oplus \mathsf{op}^{(2)}_m((a_1 \oplus 1), b_1)$
$(1, 1)$	$o_{11} = c_1 \oplus \mathsf{op}^{(2)}_m((a_1 \oplus 1), (b_1 \oplus 1))$

P_i then commits to the bits $c_1, o_{00}, o_{01}, o_{10}, o_{11}$ and proves to P_j the relations in the table using the Prove phase of $\mathcal{F}^4_{\mathrm{ECOT}}$. (We use m_0, m_1, m_2, m_3 to denote the encodings of these relations.) Next, P_i and P_j engage in a $\binom{4}{1}$-oblivious transfer so that P_j receives bit $o_{a_2 b_2}$, which is P_j's share of bit c.

The full description of UCJGE, the protocol that securely realizes $\mathcal{F}_{\mathrm{JGE}}$ in the $\mathcal{F}^4_{\mathrm{ECOT}}$-hybrid model, as well as the proof of the following theorem, appear in [25].

Theorem 3. *Protocol* UCJGE *securely realizes the $\mathcal{F}_{\mathrm{JGE}}$ functionality in the $\mathcal{F}^4_{\mathrm{ECOT}}$-hybrid model against malicious, adaptive adversaries.*

5 Efficient and Universally Composable Two-Party Computation

In this section we show how to securely realize any adaptively well-formed two-party functionality in the presence of malicious adaptive adversaries in the $\mathcal{F}_{\mathrm{JGE}}$-hybrid model. Our construction is similar to the constructions in [27,26,11] for semi-honest adversaries. However, since our $\mathcal{F}_{\mathrm{JGE}}$ functionality is secure in the presence of malicious adversaries, we are able to obtain a two-party protocol secure against malicious adversaries directly.

We first review some of the assumptions about two-party functionalities we use in our paper, which are also used in [11]. We let $\mathcal{F}$ be an ideal two-party functionality, and we let P_1 and P_2 be the participating parties. We assume that $\mathcal{F}$ may be represented via a family $\mathcal{C}_{\mathcal{F}}$ of Boolean circuits, the kth circuit representing an activation of $\mathcal{F}$ with security parameter k. Without loss of generality, we assume the circuits are composed entirely of NAND gates.[7]

For simplicity, we assume that in each activation, (1) at most one party has an input to $\mathcal{F}$ with at most k bits, (2) each party may receive at most k bits as output from $\mathcal{F}$, (3) $\mathcal{F}$ is a deterministic function, and (4) the local state of $\mathcal{F}$ after each activation can be described by at most k bits. The initial state of $\mathcal{F}$ is described by k zero bits. We assume that messages sent from $\mathcal{A}$ to $\mathcal{F}$ are ignored, and there are no messages from $\mathcal{F}$ to $\mathcal{A}$.

We note that the "deterministic function" assumption about $\mathcal{F}$ is without loss of generality, since we can always realize a probabilistic functionality $\mathcal{F}$ using a deterministic one $\mathcal{F}'$ as follows. Assuming that $\mathcal{F}$ needs k random bits, then $\mathcal{F}'$ receives a k-bit string as auxiliary input from each participating party upon the first activation, and then runs $\mathcal{F}$ using the XOR of these strings as the random bit string. It is easy to see that the simple protocol where each party sends a random k-bit string as the auxiliary input to $\mathcal{F}'$ securely realizes the ideal functionality $\mathcal{F}$ in the $\mathcal{F}'$-hybrid model.[8]

The following protocol $\Pi_{\mathcal{F}}$ realizes an activation of $\mathcal{F}$ when P_1 sends a message to $\mathcal{F}$. (The case for P_2 is analogous.) We assume that both P_1 and P_2 hold an *sid* as auxiliary input. When P_1 is activated with input (sid, v), it initiate a protocol with P_2 to perform a joint gate-by-gate evaluation of the appropriate circuit in $\mathcal{C}_{\mathcal{F}}$.

Formally, they carry out the following protocol.

Initialization: When P_1 receives (sid, v), it checks if this is the first activation of $\mathcal{F}$, and if so it sets up the internal state. Then it commits to its private input.

Setting up the internal state: For $i = 1, 2, ..., k$, P_1 sends messages $\langle \mathsf{commit}, sid, cid_i, P_2, 0\rangle$ and then $\langle \mathsf{open}, sid, cid_i, P_2\rangle$ to $\mathcal{F}_{\text{JGE}}$. P_1 waits to receive the appropriate receipts. P_2 aborts if any of the bits are not zero. Effectively, P_1 commits to the initial internal state of $\mathcal{F}$ (which is all zeros), and by opening them immediately, it proves to P_2 that these bits are indeed all-zero.[9]

Committing to the private input: For $i = 1, 2, ..., k$, P_1 sends messages $\langle \mathsf{commit}, sid, cid_i, P_1, v_i\rangle$ to $\mathcal{F}_{\text{JGE}}$ and waits to receive the appropriate

[7] This is entirely for simplicity. Note that the $\mathcal{F}_{\text{JGE}}$ functionality can be used to evaluate any gate of fan-in two.

[8] Note that if adaptive corruptions are allowed, then this is actually only true for adaptively well-formed functionalities. See [11] for a discussion on this point, and the modifications necessary for an ideal adversary in the case of probabilistic functions.

[9] Note that the *cid*'s used here and elsewhere in the protocol must all be unique bit strings that indicate the bit's use in the circuit. For instance, the cid_i here could be the bit encoding of $\langle \mathsf{state}, i\rangle$.

receipts. Here we assume that $v = v_1 v_2 \cdots v_k$.[10] P_2 simply records the receipts received from $\mathcal{F}_{\mathrm{JGE}}$.

Gate-by-gate evaluation: For each NAND gate in the circuit, P_1 determines the commitment identifiers associated with the inputs to that NAND gate, say cid_0 and cid_1, creates a new unique commitment identifier cid, sends message $\langle \mathsf{eval}, sid, cid, cid_0, cid_1, P_2, 1110\rangle$ to $\mathcal{F}_{\mathrm{JGE}}$, and waits for the appropriate receipt. Here $m = 1110$ is the encoding of the NAND operation. P_2 simply records the receipts received from $\mathcal{F}_{\mathrm{JGE}}$.

Output: P_2 verifies from all its receipts that P_1 had $\mathcal{F}_{\mathrm{JGE}}$ perform the correct computation on the appropriate bits. Then for each output bit of $\mathcal{F}$, it is either an internal state bit, or a bit addressed to either P_1 or P_2 (we have assumed that $\mathcal{F}$ does not communicate with $\mathcal{A}$). In the former case, P_1 and P_2 do not need to do anything. They simply store the identifier of this bit, so that they can use it in the next activation. In the latter case, assuming that this bit, with identifier cid, is addressed to P_2, P_1 sends a message $\langle \mathsf{open}, sid, cid, P_2\rangle$ to $\mathcal{F}_{\mathrm{JGE}}$ and P_2 extracts the bit b from the message $\langle \mathsf{DATA}, sid, cid, \{P_1, P_2\}, b\rangle$ received from $\mathcal{F}_{\mathrm{JGE}}$. The protocol for the case for a bit addressed to P_1 is the same, but with P_1 and P_2 switched.

Messages that are out of order are dealt with using tagging, as in [11].

Theorem 4. *Let $\mathcal{F}$ be a two-party adaptively well-formed functionality. Then $\Pi_{\mathcal{F}}$ securely realizes $\mathcal{F}$ in the $\mathcal{F}_{\mathrm{JGE}}$-hybrid model, in the presence of malicious adaptive adversaries.*

Proof appears in [25].

6 Efficient and Universally Composable Multi-party Computation

In this section we show how to extend the results from previous sections to securely realize any well-formed multi-party functionality in the presence of malicious adaptive adversaries corrupting an arbitrary number of parties. Our construction is similar to that in [11] for semi-honest adversaries. But, again as in the two-party case, we are able to construct building blocks that can withstand malicious adversaries, and therefore our construction is secure against malicious, adaptive adversaries directly.

In order to securely realize $\mathcal{F}$, we basically follow the same approach as in the two-party case. However, we first need to extend some of our constructions from previous sections to suit the multiple-party case.

[10] To indicate the use of each of these bits in the circuit, one could, for instance, set cid_i to be the bit encoding of $\langle \mathsf{input}, P_1, a, i\rangle$, where a is the activation number.

6.1 Broadcast and the One-to-Many ZK Functionalities

We assume an *authenticated broadcast* channel available to all participating parties. The channel is modeled by the broadcast functionality $\mathcal{F}_{\text{BC}}$ below. The functionality guarantees the authenticity of a message, i.e., that no party P_i can fake a message from P_j. This is also the assumption used in [11], and we refer the readers to [11,29] for more in-depth discussions.

Functionality $\mathcal{F}_{\text{BC}}$

$\mathcal{F}_{\text{BC}}$ proceeds as follow, running with parties $P_1, ..., P_n$ and an adversary $\mathcal{S}$.

– Upon receiving a message $(\mathsf{broadcast}, sid, \mathcal{P}, x)$ from P_i, where $\mathcal{P}$ is a set of parties, send $(\mathsf{BCAST\text{-}MSG}, sid, P_i, \mathcal{P}, x)$ to all parties in $\mathcal{P}$ and $\mathcal{S}$, and halt.

We also need an extension of the ZK functionality, namely the one-to-many ZK functionality, denoted by $\mathcal{F}_{\text{mZK}}$. Intuitively, this functionality allows a single prover to prove a theorem to multiple verifiers simultaneously. We give the formal definition in [25].

We observe that the UCZK construction by Garay *et al.* [24] can be naturally extended to a one-to-many UCZK protocol with the additional broadcast functionality. Roughly speaking, P_i (the prover) runs an independent copy of the two-party UCZK protocol with every party $P_j \in \mathcal{P}$ using a unique sid, and all messages are broadcast. Each P_j accepts if and only if all the conversations are accepting. It is straightforward to construct an ideal adversary $\mathcal{S}$. If the prover is uncorrupted, $\mathcal{S}$ simply runs a multi-party UCZK simulator for every copy of the UCZK protocol. If the prover is corrupted and there is at least one uncorrupted verifier, $\mathcal{S}$ can extract the witness. If all parties are corrupted, the simulation is straightforward. The conversion remains efficient. Therefore we have the following theorem.

Theorem 5. *Under the strong RSA assumption or the DSA assumption, for every relation R that admits an Ω-protocol Π, there exists a three-round protocol* $\mathsf{UC}[\Pi]$ *that securely realizes the $\mathcal{F}^R_{\text{mZK}}$ ideal functionality in the $(\mathcal{F}_{\text{CRS}}, \mathcal{F}_{\text{BC}})$-hybrid model against adaptive adversaries, assuming erasing. Furthermore, the computation complexity of* $\mathsf{UC}[\Pi]$ *is that of Π plus constant number of exponentiations and the generation of a signature, times the number of receiving parties.*

6.2 Multi-party ECOT

We also extend the $\mathcal{F}_{\text{ECOT}}$ functionality to the multi-party case, where the proof phase is replaced by a one-to-many proof and the receipts are sent to all participating parties. A formal definition of $\mathcal{F}_{\text{mECOT}}$ appears in [25].

It is straightforward to extend the UCECOT protocol to the multiple-party case. One simply replaces the $\mathcal{F}_{\text{ZK}}$ functionalities by the $\mathcal{F}_{\text{mZK}}$ functionalities and replaces the point-to-point messages by broadcast messages. We denote the extended protocol by $\mathsf{UCmECOT}$, and we have the following theorem.

Theorem 6. *Under the DDH assumption, protocol* $\mathsf{UCmECOT}$ *securely realizes the* $\mathcal{F}_{\mathrm{mECOT}}$ *ideal functionality in the* $(\mathcal{F}_{\mathrm{CRS}}, \mathcal{F}_{\mathrm{BC}}, \hat{\mathcal{F}}_{\mathrm{mZK}}^{R_{\mathsf{OR\text{-}DL}}}, \hat{\mathcal{F}}_{\mathrm{mZK}}^{R_{\mathsf{OR\text{-}N\text{-}DL}}}, \hat{\mathcal{F}}_{\mathrm{mZK}}^{R_{\mathsf{PEREP}}}, \hat{\mathcal{F}}_{\mathrm{mZK}}^{R_{\mathsf{OR\text{-}PEREP}}})$*-hybrid model against adaptive adversaries, assuming erasing.*

6.3 Multi-party Joint Gate Evaluation

We extend the joint gate evaluation functionality to the multi-party case. Functionality $\mathcal{F}_{\mathrm{mJGE}}$ is shown below. The only changes with respect to the two-party case are that the receipts are sent to all participating parties, and that all parties have to agree for the opening to take place.

Functionality $\mathcal{F}_{\mathrm{mJGE}}$

$\mathcal{F}_{\mathrm{JGE}}$ proceeds as follows, running with parties $P_1, ..., P_n$, and adversary $\mathcal{S}$.

- **Commit phase:** When receiving from P_i a message $\langle \mathsf{commit}, sid, cid, \mathcal{P}, b\rangle$, if $P_i \in \mathcal{P}$, then record $\langle cid, \mathcal{P}, b\rangle$, send message $\langle \mathsf{RECEIPT}, sid, cid\rangle$ to all parties in $\mathcal{P}$ and $\mathcal{S}$, and ignore all further messages of the form $\langle \mathsf{commit}, sid, cid, *, *\rangle$ and $\langle \mathsf{eval}, sid, cid, *, *, *, *\rangle$.
- **Evaluate phase:** When receiving from P_i a message $\langle \mathsf{eval}, sid, cid, cid_0, cid_1, \mathcal{P}, m\rangle$, if $P_i \in \mathcal{P}$ and both $\langle cid_0, \mathcal{P}, b_0\rangle$ and $\langle cid_1, \mathcal{P}, b_1\rangle$ are recorded, then compute $b = \mathsf{op}_m^{(2)}(b_0, b_1)$, record $\langle cid, \mathcal{P}, b\rangle$, send message $\langle \mathsf{EVAL\text{-}RECEIPT}, sid, cid, cid_0, cid_1, P_i, \mathcal{P}, m\rangle$ to all parties in $\mathcal{P}$ and $\mathcal{S}$, and ignore all further messages of the form $\langle \mathsf{commit}, sid, cid, *, *\rangle$ and $\langle \mathsf{eval}, sid, cid, *, *, *, *\rangle$; otherwise, do nothing.
- **Open phase:** When receiving from P_i a message $\langle \mathsf{open}, sid, cid, P_j\rangle$, if $P_i \in \mathcal{P}$, $P_j \in \mathcal{P}$, and the tuple $\langle cid, \mathcal{P}, b\rangle$ is recorded, then record tuple $\langle \mathsf{openreq}, sid, cid, P_j\rangle$. When a tuple $\langle \mathsf{openreq}, sid, cid, P_i, P_j\rangle$ is recorded for every $P_j \in \mathcal{P}$, then send message $\langle \mathsf{DATA}, sid, cid, b\rangle$ to P_i.

In fact, we only need a "weakened" version of the $\mathcal{F}_{\mathrm{mJGE}}$ functionality for general multi-party computation. The weakened version, denoted by $\mathcal{F}_{\mathrm{wmJGE}}$, has the additional constraint that only XOR and AND operations are allowed in the evaluation phase. It is obvious that since $\{\mathrm{XOR}, \mathrm{AND}\}$ is a complete set of Boolean operations, $\mathcal{F}_{\mathrm{wmJGE}}$ is powerful enough to realize any multi-party functionality.

As in the two-party case, we also need an extension of the $\mathcal{F}_{\mathrm{mECOT}}$ functionality, denoted by $\mathcal{F}_{\mathrm{mECOT}}^4$, that performs $\binom{4}{1}$-oblivious transfer and proves relations among four bits. We only state the following theorem and omit the details.

Theorem 7. *There exists an efficient protocol that securely realizes the* $\mathcal{F}_{\mathrm{mECOT}}^4$ *functionality in the* $\mathcal{F}_{\mathrm{mECOT}}$*-hybrid model against malicious, adaptive adversaries.*

Next, we briefly sketch a protocol UCmJGE that securely realizes $\mathcal{F}_{\mathrm{wmJGE}}$ in the $\mathcal{F}_{\mathrm{mECOT}}^4$-hybrid model. This protocol is essentially a multi-party extension to the UCJGE protocol. In UCmJGE, a bit b is now shared among all participating parties: party P_i has bit b_i such that $\sum_{i=1}^{n} b_i = b \mod 2$. In the following description, we omit some details in the protocol such as the format of the messages and the identifiers of the bits. These details should be clear from the context.

Commit phase: For party P_i to commit to a bit b, it generates random bits $b_1, b_2, ..., b_{n-1} \overset{R}{\leftarrow} \{0,1\}$ and $b_n \leftarrow b \oplus b_1 \oplus \cdots \oplus b_{n-1}$. Then P_i commits to b_i through $\mathcal{F}^4_{\text{mECOT}}$, sends bits b_j to party P_j for all $j \neq i$. Then each P_j commits to b_j through the $\mathcal{F}^4_{\text{mECOT}}$ and opens it to P_i immediately.

Evaluate phase: Assume the two bits to be computed are a and b, and P_i holds bits a_i, b_i as their shares. Naturally we have $a = \sum a_i \mod 2$ and $b = \sum b_i \mod 2$. We assume the result bit is c and each party should hold a share c_i at the end of this phase. We consider two cases according to the operation performed.

XOR: To compute the XOR of bit a and b, each party simply computes $c_i = a_i \oplus b_i$. No messages are needed.

AND: To compute the AND of bit a and b, we follow the approach in [26, 11]. Observe that AND is the multiplication modulo 2, and we have the following equality.

$$\left(\sum_{i=1}^{n} a_i\right) \cdot \left(\sum_{i=1}^{n} b_i\right) = n \cdot \sum_{i=1}^{n} a_i \cdot b_i + \sum_{1 \leq i < j \leq n} (a_i + a_j) \cdot (b_i + b_j) \mod 2$$

(see [26] for the justification of this equality). Therefore, each party P_i can compute $n \cdot a_i \cdot b_i$ by itself, and each pair P_i and P_j can jointly compute $(a_i + a_j) \cdot (b_i + b_j)$ as in the two-party case, by invoking multiple transfer phases of the $\mathcal{F}^4_{\text{mECOT}}$ functionality.

Open phase: To open a bit b, shared as $b = \sum_{i=1}^{n} b_i$, to party P_i, every P_j opens its share b_j through $\mathcal{F}^4_{\text{mECOT}}$. Then P_i sums up all the shares to obtain b.

Abort: In case any party aborts and/or deviates from the protocol, all parties abort the protocol.

Theorem 8. *Protocol* UCmJGE *securely realizes functionality $\mathcal{F}_{\text{mJGE}}$ in the $\mathcal{F}^4_{\text{mECOT}}$-hybrid model against malicious, adaptive adversaries.*

The proof is very similar to that of Theorem 3. Next, for any multi-party functionality $\mathcal{F}$, we construct a protocol $\Pi_{\mathcal{F}}$ that securely realizes $\mathcal{F}$ in the $\mathcal{F}_{\text{mJGE}}$-hybrid model. The construction is almost identical to the two-party case, except that since we assume the circuit computing $\mathcal{F}$ consists of AND and XOR gates, instead of NAND gates, the gate-by-gate evaluation will invoke the $\mathcal{F}_{\text{mJGE}}$ functionality with different encodings of functions. Again, we defer the detailed construction and proof to the full version of this paper.

Theorem 9. *Let $\mathcal{F}$ be a multi-party adaptively well-formed functionality. Then $\Pi_{\mathcal{F}}$ securely realizes $\mathcal{F}$ in the $\mathcal{F}_{\text{mJGE}}$-hybrid model, in the presence of malicious adaptive adversaries.*

Acknowledgements. The authors thank the anonymous reviewers for TCC'04 for the many valuable comments.

References

1. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology–Eurocrypt '97*, pp.480–494, 1997.
2. D. Beaver. Foundations of Secure Interactive Computing. In *Advances in Cryptology – CRYPTO '91*, pp. 377–391, 1991.
3. D. Beaver. Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. In *Journal of Cryptology* 4(2), pp. 75–122, 1991.
4. M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. *Advances in Cryptology—CRYPTO '89*, pp. 547–557, volume 435 of *Lecture Notes in Computer Science*, Springer-Verlag, 1990.
5. D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium* (LNCS 1423), pp. 48–63, 1998.
6. F. Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In *EUROCRYPT 2000*, pp. 431–444, 2000.
7. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 414–430, Springer-Verlag, 1999.
8. R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. In *Journal of Cryptology* 13(1), pp. 143–202, 2000.
9. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pp. 136–145, 2001.
10. R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO 2001* (LNCS 2139), pp. 19–40, 2001.
11. R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally composable two-party computation. In *STOC '02*, pp. 494–503, 2002. Full version in *Cryptology ePrint Archive*, Report 2002/140.
12. R. Canetti and T. Rabin. Universal Composition with Joint State In *Cryptology ePrint Archive*, Report 2002/047, `http://eprint.iacr.org/`, 2002.
13. R. Cramer. Modular Design of Secure yet Practical Cryptographic Protocols. Ph.D. Thesis. CWI and University of Amsterdam, 1997.
14. R. Cramer, I. Damgård, and J. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption In *Advances in Cryptology–EuroCrypt '01*, volume 2045 of *Lecture Notes in Computer Science*, pp. 280–300, Springer-Verlag, 2001.
15. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology–CRYPTO '94* (LNCS 839), pages 174–187, 1994.
16. R. Cramer and V. Shoup. Signature scheme based on the strong RSA assumption. In *ACM Transactions on Information and System Security (ACM TISSEC)* 3(3):161-185, 2000.
17. C. Crépeau. Verifiable disclosure of secrets and applications. In *Eurocrypt '89*, LNCS 434, pp 181 – 191, 1990.
18. C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In *Advances in Cryptology—CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 110–123. Springer-Verlag, 27–31 Aug. 1995.

19. I. Damgård and J. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO 2002* (LNCS 2442), pp. 581–596, 2002. A later version in *Cryptology ePrint Archive*, Report 2001/091. `http://eprint.iacr.org/`, 2001.
20. I. Damgård, and J. Nielsen. Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In *Advances in Cryptology–CRYPTO '03*, 2003.
21. S. Even, O. Goldreich, and A. lempel, A Randomized Protocol for Signing Contracts. In *Communications of the ACM*, 28:637-647 (1985).
22. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology–CRYPTO '97*, pp. 16-30, 1997.
23. J. Garay and P. MacKenzie. Concurrent Oblivious Transfer. In *FOCS 2000*, pp. 314-324, Redondo Beach, CA, November 2000.
24. J. Garay, P. MacKenzie and K. Yang. Strengthening Zero-Knowledge Protocols using Signatures. In *Advances in Cryptology–Eurocrypt 2003*, Warsaw, Poland, LNCS 2656, pp.177-194, 2003. Full version available from *Cryptology ePrint Archive*, Report 2003/037, `http://eprint.iacr.org/2003/037`, 2003.
25. J. Garay, P. MacKenzie and K. Yang. Efficient and Universally Composable Committed Oblivious Transfer and Applications (full paper). To appear in *Cryptology ePrint Archive*.
26. O. Goldreich. Secure Multi-Party Computation (Working Draft, Version 1.2), March 2000. Available from `http://www.wisdom.weizmann.ac.il/~oded/pp.html`.
27. O. Goldreich, S. Micali and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pp. 218–229, 1987.
28. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority, In *Advances in Cryptology-CRYPTO '90*, pp. 77-93, Springer-Verlag, 1991.
29. S. Goldwasser and Y. Lindell. Secure Computation Without Agreement. In 16th *DISC*, Springer-Verlag (LNCS 2508), pages 17-32, 2002.
30. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority, In *Advances in Cryptology–CRYPTO '90*, pp. 77-93, Springer-Verlag, 1991.
31. P. MacKenzie and M. Reiter. Two-Party Generation of DSA Signatures. In *Advances in Cryptology–CRYPTO '01*, pp 137–154, 2001.
32. S. Micali and P. Rogaway. Secure Computation (Abstract). In *Advances in Cryptology – CRYPTO '91*, pp. 392–404, 2001.
33. T. Osamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology–Eurocrypt '98*, pp.380–318, 1998.
34. P. Paillier. Public-key cryptosystems based on composite degree residue classes. In *Advances in Cryptology–Eurocrypt '99*, pp.223–238, 1999.
35. T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO 1991*, pp. 129–140, 1991.
36. B. Pfitzmann, M. Schunter, and M. Waidner. Provably Secure Certified Mail. In *IBM Research Report RZ 3207 (#93253) 02/14/00*, IBM Research Division, Zürich, August 2000.
37. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security (CCS 2000)*, pp. 245–254, 2000.

38. B. Pfitzmann and M. Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE Symposium on Security and Privacy*, pp. 184–200, 2001.
39. P. Rogaway. The round complexity of secure protocols. Ph.D. Thesis, MIT, June 1991.
40. A. Yao. Protocols for Secure Computation. In *FOCS 1982*, pages 160–164, 1982.