Fast and Universally-Composable Oblivious Transfer and Commitment Scheme with Adaptive Security

Megha Byali, Arpita Patra, Divya Ravi and Pratik Sarkar

Indian Institute of Science, India
{megha,arpita,divyar,pratiks}@iisc.ac.in

Abstract. Adaptive security embodies one of the strongest notions of security that allows an adversary to corrupt parties at any point during protocol execution and gain access to its internal state. Since it models real-life situations such as "hacking", efficient adaptively-secure multiparty computation (MPC) protocols are desirable. Such protocols demand primitives such as oblivious transfer (OT) and commitment schemes that are adaptively-secure as building blocks. Efficient realizations of these primitives have been found to be challenging, especially in the no erasure model. We make progress in this direction and provide efficient constructions that are Universally-Composable in the random oracle model.

Oblivious Transfer. We present the first round optimal framework for building adaptively-secure OT in the programmable random oracle (PRO) model, relying upon the framework of Peikert et al. (Crypto 2008). When instantiated with Decisional Diffie Hellman assumption, it incurs a minimal communication overhead of one $\kappa$ bit string and computational overhead of 5 random oracle queries over its static counterpart, where $\kappa$ is the security parameter. This computation overhead translates to 0.02% and 1% in the LAN and WAN setting. Additionally, we obtain a construction of adaptively-secure 1-out-of-N OT by extending the result of Naor et al. (Journal of Cryptology 2005) that transforms $\log N$ copies of 1-out-of-2 OTs to one 1-out-of-N OT in the PRO model. We complete the picture of efficient OT constructions by presenting the first adaptively secure OT Extension, extending the protocol of Asharov et al. (Eurocrypt 2015) for the adaptive setting using PRO. Our OT extension enables us to obtain adaptive OTs at an amortized cost of 3 symmetric key operations and communication of $3\kappa$ bit strings. It incurs a runtime overhead of 2% and 11.95%, in the LAN and WAN setting and almost no communication overhead over the static OT extension protocol. In concrete terms, the cost is 2 $\mu$s and 115 $\mu$s for each OT in LAN and WAN.

Commitment Scheme. We present an adaptively secure commitment scheme in the Global Random Oracle model solely relying on observable random oracle (ORO). Our commitment scheme has a one-time offline setup phase, where a common reference string (CRS) is generated between the parties using an ORO. In the online phase, the parties use the CRS and ORO to generate commitments in a non-interactive fashion. Our construction incurs communication of $4\kappa$ bit strings and computation of 4 exponentiations and 4 random oracle queries for committing to an arbitrary length message. Empirically, it takes around 0.18ms and 0.2 ms for committing to 128 bits and 2048 bits respectively. It finds applications in secure two-party computation (2PC) protocols that adopt offline-online paradigm, where the CRS can be generated in the offline phase and the scheme can be used in the online phase.
Secure multiparty computation (MPC) \cite{Yao82,GMW87} is an area of cryptography that deals with multiple parties who wish to compute a joint function of their private inputs such that the goals of MPC i.e. correctness and privacy of inputs remain intact. Various security notions of MPC are defined in accordance with different types of adversaries. The area of MPC achieving security against a malicious adversary causing static corruptions has been the center of attention in the recent past. Here, static corruption refers to the model in which the adversary corrupts the subset of parties at the outset of the protocol. Most widely known protocols of \cite{Lin13, HKK+14, MR17}, consider malicious security against static corruptions. Although static security is of interest, it is desirable to achieve security against adaptive adversary. Adaptive security is a stronger corruption model that allows the adversary to choose which parties to corrupt during the protocol execution and models real-life situations in a more comprehensive way. For instance, it captures the event of “hacking”, where a hacker can illegally capitalize on the system and corrupt any workstation while protocols are in execution. Adaptive security is further classified based on secure erasures of the memory. An adaptively-secure protocol assuming erasure allows secure erasure of a workstation’s internal memory once it is corrupted by a hacker. \cite{Can01} argued that security relying on erasures often leads to problems and is impractical, especially for real-life systems. It requires an inherent trust assumption on the part of a workstation that it will erase its memory upon being corrupted. Hence, adaptive security without erasures (referred to simply as adaptive-security throughout the paper) is preferable as it precisely models real world “hacking” attacks. However, the current literature of MPC dealing with adaptive adversaries is less explored compared to static security since it turns out to be considerably more challenging. In this paper, we explore the less traveled path of dealing with adaptive adversaries in the Universal Composability (UC) model of \cite{Can01}. We aim at obtaining efficient, adaptively-secure primitives which are instrumental in construction of practical adaptively-secure MPC protocols. We focus on two such primitives in particular: Oblivious Transfer (OT) and Commitment schemes.

In the literature, OT has been regarded as the fundamental primitive \cite{Rab81, BCR86, Kil88, NP05, IPS08}, known to be complete for MPC \cite{Kil88}. Following that, many flavors of OT such as 1-out-of-2 OT \cite{PVW08, CO15}, 1-out-of-N OT \cite{NP05}, k-out-of-N OT \cite{GH08}, have been explored in past. In its most basic form, a 1-out-of-2 OT consists of two parties, sender $S$ and a receiver $R$. $S$ has input messages say $m_0, m_1$ and $R$ has a choice bit $b$. At the end of the protocol, $R$ obtains message $m_b$ corresponding to his choice bit and nothing else. $S$ remains oblivious to the message obtained by $R$. Various constructions of OT (mainly 1-out-of-2) have been proposed providing both static and adaptive security. The protocols of \cite{CO15, PVW08, NP05} deal with malicious adversaries in the static model whereas those of \cite{BCG17, BC16, CKWZ13, GWZ09} address the same in adaptive model. However, the OT protocols that achieve adaptive security lack optimality in terms of efficiency- rounds, computation and communication. We have round optimal and efficient protocols in the literature for the static case. However, designing adaptively-secure protocols has been a challenging task. Infact, there is no known OT protocol in the literature which is round optimal. In this direction, we
present the first round optimal OT protocol which is secure against adaptive adversaries without erasures.

Another interesting direction to consider for OTs is to reduce the number of public key operations. The impossibility result by [IR89] states that its highly unlikely that OTs can be constructed without public key operations. In order to circumvent this limitation, the concept of OT Extension [Bea96, IKNP03, ALSZ13, ALSZ15, KOS15] was introduced and explored. It allows the parties to execute a small number of OTs, called seed OTs, and then extend them to obtain large number of OTs using cheap symmetric key operations. The amortized cost of generating one single OT reduces to a constant number of symmetric key operations. However, there is no known OT extension protocol in the adaptive setting. An effort towards obtaining adaptively secure OT Extension would be of quite interest as it would open the gates towards constructing large number of efficient adaptively secure OTs using small number of seed OTs. Our paper presents one such result for adaptively-secure OT extension.

Commitment Scheme is another fundamental primitive in the MPC literature that draws attention. Informally, we describe a commitment scheme as follows: The sender $S$ commits to a message $m$ in a commitment $c$ and sends $c$ to the receiver $R$ in the commit phase. In the decommit phase, $R$ learns the message $m$, along with some decommitment information, such that $R$ is convinced that indeed $m$ was committed in the commit phase. Besides its involvement in MPC protocols, commitment schemes in the UC model also have implications in key exchange [DG03] and are non-malleable [CF01] in nature. In this work, we explore UC secure commitment protocols which are secure against adaptive adversaries.

1.1 Related Work

The extensive use of fundamental building blocks of MPC has led to substantial work towards attaining efficient primitives. Since our work primarily focuses on universally composable OT and commitment schemes, we outline only the relevant literature below.

**Oblivious Transfer.** The literature of OT is vast and quite diverse in terms of assumptions and security. We highlight few works that are closely related to ours. Firstly, in the standalone model, the works of [NP01, AIR01, HK12, Lin08] are statically-secure against malicious adversary. Secondly, in the UC model, [CLOS02] proposed the first UC secure OT protocol based on general assumptions. Their work includes construction of both static and adaptively UC-secure OTs. Despite being of theoretical interest, [CLOS02] motivated research towards obtaining OTs in the UC model. In the setting of static security, [GMY04] presented a constant round committed bit-OT under Decisional Diffie Hellman (DDH) and RSA assumptions. The work of [JS07] proposed a four round, UC-secure protocol under the Decisional Composite Residue (DCR) assumption. [HK07] provided the first round-optimal protocol that is UC-secure, assuming common reference string (crs). This was followed by the seminal work of Peikert et al. [PVW08], that provided a general framework for round optimal UC-secure OT protocols along with efficient instantiations based on DDH, Quadratic Residue and Learning With Errors (LWE) assumptions in the crs model. We denote the popular DDH based construction of the [PVW08] paper as PVW protocol in rest of the paper.
In the setting of adaptive security, the literature can be divided based on the erasure model. The works of [ABB+13, CKWZ13, BCG17, BC16, BC15] rely on secure erasures of the memory, whereas [GWZ09, CDMW09a, CDMW09b, BDD+17] consider the stronger model of no erasures. [CKWZ13] presented a framework for adaptively-secure OT in the global crs model. They provide instantiations under various assumptions- DLIN, Symmetric External Diffie Hellman (SXDH), DDH and DCR. However, their protocols are not round optimal and achieve adaptive security at the cost of significant overhead in communication and computation compared to the PVW protocol. [CKWZ13] also provided two constructions (Appendix A of [CKWZ13]) of [GWZ09] framework under Decisional Linear (DLIN) assumptions. These instantiations are adaptively-secure with erasures, with computational overhead reduced to constant number of exponentiations. There has been a separate line of work [BCG17, BC16, BC15, ABB+13] based on password-authenticated key exchange (PAKE) and smooth projective hash functions. They require at least 3 rounds of communication, assuming erasures for adaptivity. On the other hand, achieving adaptivity without erasures is a challenging task. [GWZ09] followed the compiler approach to transform the [PVW08] framework into an adaptively-secure OT using adaptively secure commitments. It was further improved by a recent work of [ABP17]. Both protocols incur an overhead of $O(n)$ exponentiations, where $n$ is the size of sender’s input messages. The works of [CDMW09a, CDMW09b] proposed constructions which are of theoretical interest. [CDMW09a] presented an OT construction using non-committing encryption (NCE). On the other hand, [CDMW09b] compiled protocols that are secure against a semi-honest adaptive adversary into one that is secure against a malicious adaptive adversary.

The work of [CO15], the “simplest OT” protocol explored 3-round OT constructions in the PRO model. Although the paper claimed adaptive security, several bugs have been identified [GIR17, BDD+17, HL17] recently in their static security proof thereby rendering the protocol of [CO15] insecure in the UC model. Recently two more works ([HL17], [BDD+17]) have claimed to achieve adaptive security in the same model. However, in Section 3, we give a justification that these protocols are not UC-secure. We note that the authors of [BDD+17] have updated their protocol and their new version presents a 3-round OT framework which can be instantiated under Learning from Parity with Noise, McEliece cryptosystem, QC-MDPC, LWE and Computational Diffie Hellman (CDH) in the PRO model. Their most efficient instantiation (under CDH) incurs twice the amount of communication, while maintaining the same computation cost as ours. Consequently, the problem of attaining an round-optimal, efficient and adaptively-secure OT continued to remain open, which we try to address through our work. Table 1 summarizes the literature on adaptively-secure OT protocols and our result. We do not compare with [GWZ09, CDMW09a, CDMW09b, ABP17] in the table since they require at least $O(n)$ exponentiations, where sender’s input message is of $n$ bits, whereas the other protocols in the table require constant number of exponentiations. Among the PAKE-based schemes we compare with the most efficient works of [ABB+13, BCG17].

Another interesting factor to consider is the model of security that these protocols satisfy. All of the above protocols (except [BDD+17]) are secure in the plain UC-
model [Can01] which allows private crs. There is a stronger model of security, i.e. the Generalized UC model (GUC) [CDPW07], which guarantees stronger compositional guarantees than the plain UC model. In the GUC model, the parties are not allowed to possess private crs and the random oracle is a public object, shared between arbitrary protocols. This prevents the simulator from manipulating the crs during simulation, rendering the simulation procedure more difficult. Interestingly, the recent work of [CDG+18] has shown that the random oracle in the GUC model can satisfy the property of programmability. Such a random oracle is formally known as Global Random Oracle (GRO). Utilizing the programmability feature of the GRO, our protocol and the protocol of [BDD+17] achieve adaptive security. Whereas, the other protocols require a local crs, which is private to one instance of the protocol, thereby hampering composition in the GUC model.

Oblivious Transfer Extension. Next, we consider the problem of OT Extension, which was introduced by the work of [Bea96], followed by the seminal work of [IKNP03]. The paper of [IKNP03] presented an efficient 1-out-of-2 semi-honest OT Extension protocol which was secure against static adversaries. An optimized version of this protocol appeared in [ALSZ13]. The paper of [ALSZ15, KOS15] presented the actively secure versions. The paper of [KK13] gave constructions for 1-out-of-N case, which were made actively secure by [PSS17, OOS17]. However all of these protocols are in the static setting and it was not known whether adaptively-secure OT Extension protocol is possible. Our work answers it in an affirmative way by proving that existing static OT extension [ALSZ13, ALSZ15] schemes satisfy adaptive security under the PRO assumption.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Communication (κ-bit strings / Group elements)</th>
<th>Computation</th>
<th>Setup</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GWZ09] + [FLM11]</td>
<td>83</td>
<td>3</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>[CKWZ13]</td>
<td>59</td>
<td>3</td>
<td>≥14</td>
<td>2</td>
</tr>
<tr>
<td>[CKWZ13]</td>
<td>43</td>
<td>3</td>
<td>≥8</td>
<td>2</td>
</tr>
<tr>
<td>[CKWZ13]</td>
<td>35</td>
<td>4</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>[CKWZ13]</td>
<td>28</td>
<td>4</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>[ABB+13]</td>
<td>15</td>
<td>2</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>[BCG17]</td>
<td>10</td>
<td>4</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>PVW</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>[BDD+17]</td>
<td>15</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Our scheme</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>(by OT Extension)</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>

Notations:
- SKE - symmetric key encryptions, PKE - exponentiations, GUC - Generalized UC
- DDH - Decisional Diffie Hellman, DLIN - Decisional Linear, SXDH - symmetric external Diffie Hellman
- CDH - Computational Diffie Hellman, DCR - Decisional Composite Residuosity, PRO - programmable random oracle
1 The commitment scheme used for instantiation is of [FLM11].
Commitment Scheme. The study of UC secure commitment schemes was initiated by the seminal work of [CF01]. It was followed by the works of [CLOS02, DG03, HM04, Lin11, Fuj16, ABP17] and many more. We highlight some of the notable works in the relevant literature classified based on their round complexity, the security they achieve and the security model they are proven to be secure in.

The contributions of [DG03, Lin11, GIKW14, Fuj16, FLM11, CJS14, ABP17] based on hardness assumptions such as DDH, DLIN and Discrete Log (DLP) are interactive (involve either an interactive commit or decommit phase) in nature. In contrast, the contributions of [CLOS02, CF01, HM04, FLM11, NFT09, ABP17] present non-interactive commitment (NICOM) schemes where both - commit and decommit phases are non-interactive. The offline-online paradigm also forms an interesting flavour of NICOM schemes. These schemes [CDD+15, DDGN14] consist of an input independent setup phase, which is run in the offline phase, while the NICOM is efficiently used in the online phase. The cost of the setup phase can be substantial but it gets amortized over multiple commitments. Our scheme also follows this particular model.

Similar to OT literature, the literature regarding commitment schemes is concentrated mainly around static security [Lin11, BCPV13, CJS14, GIKW14, DDGN14, CDD+15, Fuj16, CDD+16]. Building commitment schemes against adaptive adversaries has been a challenging task, since it involves equivocation of the internal states of the parties in case corruption occurs. There have been few contributions in the past addressing adaptive security. Adaptively-secure schemes can be broadly classified into two categories based on their ability to erase the internal state of the party when corruption occurs. [NFT09, FLM11, BCPV13, Fuj16] proposed adaptively-secure protocols which rely on secure erasure of the party’s internal state whereas the constructions of [CF01, CLOS02, ABB+13, HV15, ABP17, HPV17] achieved the same level of security without erasures. We skim through the protocols in the non-erasure model and compare all the above mentioned works with our protocol in Tab. 2.

[HV15] presented a theoretical construction of an interactive adaptive commitment scheme based on the minimal assumption of trapdoor simulatable public-key encryption. Since our focus is on adaptively-secure NICOM schemes, we elaborate the relevant NICOM results. [CF01, CLOS02, ABB+13, ABP17] provided schemes for bit commitments, communicating at least $O(\kappa^2)$ bits for committing to a $\kappa$ bit string. [HM04] provided the first efficient NICOM for an arbitrary length message in the RO model and involves communication of constant number of bits for commitment. Programmability of RO is used to attain the property of equivocation in [HM04].

The literature can also be classified based on the security model they are proven secure in. All of the above protocols (except [HM04, CJS14, HPV17]) are proven secure in the plain UC-model [Can01]. Recall, that the work of [CDG+18] showed that the GUC model allows observability and programmability from the random oracle. Utilizing the programmability feature of the RO, the protocol of [HM04] achieves adaptive security in the GUC model. Also, the work of [CDG+18] showed that the folklore commitment scheme $F_{RO}(m; r)$ is GUC secure assuming programmability from $F_{RO}$, where $F_{RO}$ is the RO functionality, $m$ is the message and $r$ is the randomness. And it can be trivially shown that it achieves adaptive security too. The work of [CJS14] presents a GUC secure static commitment scheme, relying on the observability prop-
erty of the GRO. [HPV17] presents a theoretical construction of an adaptively secure commitment scheme relying on one way function. They achieve GUC security in the tamper-proof hardware model. From the literature we can observe that, adaptive commitments in the GUC model either asks for programmability from the RO or it incurs a blowup in terms of efficiency. Hence, we can ask the following question:

“Can we obtain an efficient, non-interactive commitment scheme based on the non-programmable random oracle which is adaptively secure in the GUC model?”

Our paper answers it in affirmative by presenting an adaptively-secure NICOM in the offline-online model, relying solely on the observable property of the RO. Table 2 consolidates the comparison of various UC secure commitment schemes alongside our protocol.

Table 2: Comparison among UC secure commitment schemes

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Message Size (bits)</th>
<th>Communication (c-bit strings / Group elements)</th>
<th>Rounds (Commit/Decommit)</th>
<th>Assumptions</th>
<th>Setup</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Lin11]</td>
<td>c</td>
<td>14</td>
<td>1/4</td>
<td>DDH + CRHF</td>
<td>crs</td>
<td>Static</td>
</tr>
<tr>
<td>[Fuj16]</td>
<td>c</td>
<td>10</td>
<td>1/2</td>
<td>DDH + CRHF</td>
<td>crs</td>
<td>Static</td>
</tr>
<tr>
<td>[BCPV13]</td>
<td>c</td>
<td>12</td>
<td>1/3</td>
<td>DDH + CRHF</td>
<td>crs</td>
<td>Static</td>
</tr>
<tr>
<td>[CDD16]</td>
<td>c</td>
<td>10 + o(1)</td>
<td>5/1</td>
<td>OT</td>
<td>crs</td>
<td>Static</td>
</tr>
<tr>
<td>[CDD15]</td>
<td>c</td>
<td>5 + O(1)</td>
<td>5/1</td>
<td>OT</td>
<td>crs</td>
<td>Static</td>
</tr>
<tr>
<td>[CJS14]</td>
<td>poly(c)</td>
<td>7</td>
<td>2/3</td>
<td>DLP, ORO</td>
<td>crs</td>
<td>Static (GUC model)</td>
</tr>
<tr>
<td>[FJM11]</td>
<td>c</td>
<td>21</td>
<td>1/1</td>
<td>DLIN + CRHF</td>
<td>crs</td>
<td>Adaptive with erasures</td>
</tr>
<tr>
<td>[NFT09]</td>
<td>c</td>
<td>7</td>
<td>1/1</td>
<td>DDH + sEUF-OT, (Non-reusable)</td>
<td>crs</td>
<td>Adaptive with erasures</td>
</tr>
<tr>
<td>[YK16]</td>
<td>c</td>
<td>10</td>
<td>3/1</td>
<td>DDH + CRHF</td>
<td>crs</td>
<td>Adaptive with erasures</td>
</tr>
<tr>
<td>[BCPV13]</td>
<td>c</td>
<td>14</td>
<td>3/1</td>
<td>DDH + CRHF</td>
<td>crs</td>
<td>Adaptive with erasures</td>
</tr>
<tr>
<td>[CJS14]</td>
<td>poly(c)</td>
<td>O(n)</td>
<td>1/1</td>
<td>DDH + UOWHF</td>
<td>crs</td>
<td>Adaptive</td>
</tr>
<tr>
<td>[CLOS08]</td>
<td>poly(c)</td>
<td>O(n)</td>
<td>1/1</td>
<td>TDP</td>
<td>crs</td>
<td>Adaptive (GUC model)</td>
</tr>
<tr>
<td>[HM04]</td>
<td>poly(c)</td>
<td>5</td>
<td>1/1</td>
<td>DLP</td>
<td>PRO</td>
<td>Adaptive (GUC model)</td>
</tr>
<tr>
<td>[ABB13]</td>
<td>poly(c)</td>
<td>O(n)</td>
<td>1/1</td>
<td>SXDH</td>
<td>crs</td>
<td>Adaptive</td>
</tr>
<tr>
<td>[ABP17]</td>
<td>poly(c)</td>
<td>O(n)</td>
<td>1/1</td>
<td>DLP</td>
<td>crs</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>poly(c)</td>
<td>O(n)</td>
<td>1/1</td>
<td>DLP</td>
<td>crs</td>
<td>Adaptive (GUC model)</td>
</tr>
</tbody>
</table>

Notations:

DDH - Decisional Diffie Hellman, CRHF - collision resistant hash function, DLP - Discrete Log Problem, DLIN - Decisional Linear,
sEUF-OT - strongly unforgeable one-times signature, TDP - trapdoor permutations, UOWHF - universal one-way hash function, ORO - observable random oracle, crs - circuit, g^x

Note: The protocol of [CJS14] and [HM04] requires a non-interactive trapdoor commitment scheme. It has been instantiated with Pedersen commitment since to the best of our knowledge such a commitment scheme does not exist based on OWF.

1.2 Our Results

In this work, we focus on optimizing the round complexity while attaining adaptive security in an efficient manner, for two widely used primitives: Oblivious Transfer and Commitment Scheme. We also present an adaptively secure well-defined transformation from log N 1-out-of-2 OTs to 1-out-of-N OT, restricting the number of exponen-
tations to $O(\log N)$. Finally, we conclude with detailed implementation results of our primitives. Our contributions are stated below.

**Adaptively Secure 1-out-of-2 Oblivious Transfer.** We construct the first OT framework that is round-optimal, GUC composable and adaptively-secure assuming no erasures. Our construction is motivated by the vital observation of [CO15] that the CRS in OT can be replaced with the PRO. We apply the same observation on the static OT framework of [PVW08]. At the heart of their framework lies the Dual Mode Encryption Scheme (DME) which requires a CRS for its functioning. We generate the CRS of the Dual Mode Encryption (DME) scheme using the PRO. During simulation the CRS can be suitably modified, to equivocate R’s view, by programming the PRO. However, for our scheme it should be possible to generate the CRS of the DME using an RO. Hence, we customize the definition of DME, based on our requirements, to obtain a stronger version, called Samplable DME. Once the CRS has been generated it can be suitably modified, to extract/equivocate R’s view, by programming the PRO. On the other hand, S’s messages are encrypted using another PRO, such that it enables equivocation of S’s view when required. Thus, we replace the CRS in the round-optimal [PVW08] framework with PRO to achieve adaptive security. A similar observation was made by the work of [CJS14] where they tried to generate the CRS using the observability property of the GRO. However, their goal was to obtain one-sided simulatable static OT in the GUC model. Whereas, we aim for adaptive security in the GUC model relying on the programmability feature. In our framework, the DME scheme can be instantiated under the DDH and LWE assumptions. Additionally, when instantiated with DDH assumption, our protocol incurs a computation overhead of 5 random oracle queries and a minimal communication overhead of one $\kappa$-bit string over the static protocol of PVW (the DDH-based instantiation of [PVW08]).

Tab. 1 and Tab. 3 compare our scheme with various other schemes. Empirically our protocol requires 1.676s and 4.3s in LAN and WAN setting, incurring a nominal overhead of 0.02\% and 1\%, in the same settings, over PVW. The protocol of [BDD+17] requires at least the same runtime as ours, since the number of symmetric key operations in [BDD+17] is more than us, although the number of public key operations in both the protocols are same.

### Table 3: Table indicating implementation results for Oblivious Transfer protocol in LAN setting, for $n = 128$ and $\kappa = 128$.  

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Communication (bits)</th>
<th>Total Runtime (s)</th>
<th>Rounds</th>
<th>Assumptions</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVW</td>
<td>768</td>
<td>2.645</td>
<td>2</td>
<td>DDH</td>
<td>Static</td>
</tr>
<tr>
<td>[BDD+17]</td>
<td>1920</td>
<td>$&gt; 2.647$</td>
<td>3</td>
<td>CDH</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>896</td>
<td>2.647</td>
<td>2</td>
<td>DDH</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Our Scheme (from OT extension)</td>
<td>716.7</td>
<td>$20.954 \times 10^{-7}$</td>
<td>3</td>
<td>DDH</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Our Scheme (from OT extension)</td>
<td>702</td>
<td>$1.043 \times 10^{-7}$</td>
<td>5</td>
<td>DDH</td>
<td>Adaptive</td>
</tr>
</tbody>
</table>
Adaptively Secure 1-out-of-N Oblivious Transfer. The work of [NP05] established that $\log N$ copies of 1-out-of-2 OTs can be transformed to obtain one 1-out-of-N OT, which is statically-secure against active adversaries. This transformation implies existence of statically-secure 1-out-of-N OT at the expense of $O(\log N)$ exponentiations. We extend their result to provide a formal proof that the transformation satisfies adaptive security under PRO assumption. At present, one adaptive 1-out-of-N OT protocol [ABB+13, BC15, BC16, BCG17] incurs atleast $O(N)$ exponentiations. Our adaptive transformation brings down the number of exponentiations to $O(\log N)$; thereby matching the efficiency of statically-secure 1-out-of-N OT. Interestingly, it can be shown that for the semi-honest setting the seed OTs can be statically secure, if we consider the simulation of the 1-out-of-2 OTs in a non-blackbox manner. For the active setting, we can show that if the 1-out-of-2 OTs support equivocation of receiver’s view irrespective of equivocation of sender’s view then it is possible to generate adaptively-secure 1-out-of-N OT from our transformation, if the simulation of the 1-out-of-2 OTs is performed in a non-blackbox manner. This implies that we can plug-in statically-secure 1-out-of-2 OTs which satisfy the property of receiver equivocability [GKPS18].

Oblivious Transfer Extension. We provide the first adaptively-secure protocols for OT Extension solely relying on the PRO assumption. In this regard we present two results, one corresponding to semi-honest setting and the other for the active setting. Our first result proves that the semi-honest protocol of [ALSZ13] can be made adaptively-secure. Interestingly, we show that the seed OTs can be statically secure, if we invoke them in a non-blackbox way. We know that for adaptive security, blackbox-usage of the seed OTs is not possible in our construction since it would violate the results of [LZ13, IR89]. [LZ13] proves that the existence of OT extension protocol, secure against semi-honest adaptive adversaries, imply OT protocol secure against static semi-honest adversaries. In that case, blackbox usage of seed OTs establishes that PRO would imply static semi-honest OT, contradicting the result of [IR89] which states that public key operations are necessary for static OT. Our second result proves that the actively secure protocol of [ALSZ15] can be made adaptively-secure against active adversaries. The paper of [ALSZ15] gave two constructions - a 3 round and a 5 round OT extension in their paper. The 3 round OT extension consists of pair-wise consistency check for all the columns. Whereas, in the 5 round protocol, the parties performed a coin-tossing sub-protocol and based on the outcome of the subprotocol, they performed a consistency check on the columns. The 5 round protocol is more efficient than the 3 round one due to reduced number of checks. Similarly, we consider a 3 round and a 5 round variant of our OT extension protocol. Interestingly, we demonstrate that the seed OTs in our case can be replaced with receiver equivocal static OTs which are secure against active adversaries. Our OT Extension protocols preserve the efficiency of the original static protocols, yielding adaptive 1-out-of-2 OTs at an amortized cost of 3 symmetric key operations and 3κ bits communication per OT. Moreover, if we combine the OT Extension protocol with our 1-out-of-N Transformation, then we obtain 1-out-of-N adaptive OTs at an amortized cost of $N + 3\log N + 1$ symmetric key operations per OT. The other adaptive protocols [ABB+13, BC15, BC16, BCG17, BDD+17] require $O(N)$ public key operations instead. Next, we briefly highlight the implementation results regarding the OT extension protocol. The 5 round protocol is more efficient than...
the 3 round one due to reduced number of checks. Empirically, the 3 round protocol has a rough multiplicative overhead of 20 over the 5 round protocol for LAN runtime, while generating $2^{23}$ OTs together. The overhead will decrease for larger number of extended OTs, as the number of checks remain constant, while the number of extended OTs increase. Throughout the paper our implementation results for the OT extension protocol, correspond to the 5 round protocol unless specified otherwise. Empirically our OT extension protocol has an overhead of 2% and 11.95% over [ALSZ15] in the LAN and WAN setting respectively. The amortized (over $1.25 \times 10^5$) cost of generating one adaptive 1-out-of-2 drastically reduces to 2.02 μs and 115.5 μs in LAN and WAN setting for OT output length $n=16$ bits. We also generate 1-out-of-16 OTs from 4 copies of 1-out-of-2 OTs using our transformation. The amortized cost for each 1-out-of-16 OT is 21.3 μs and 145.5 μs in LAN and WAN setting respectively. In Tab. 3 we compare our OT protocols with PVW and [BDD+17].

Commitment Scheme. We construct a NICOM, in the offline-online model, that is secure against an adaptive adversary without erasures. First, we generate a crs for Pedersen commitment using an ORO in the setup phase. The crs is of the form $(g, h)$, where $h = g^x$ for $g, h \in G$ and $x \in \mathbb{Z}_p$. $G$ and $\mathbb{Z}_p$ denote a multiplicative group of prime order $p$ and a prime field of order $p$ respectively. The setup phase is a 4 round protocol where the parties perform a coin tossing protocol and a Zero Knowledge Proof of Knowledge (ZKPoK) protocol, relying on garbled circuits. Once generated, the crs alongwith the ORO can be reused to construct several instances of the NICOM. Under the hood, the NICOM relies on the Pedersen Commitment [Ped91] for equivocation and the ORO for extraction of a corrupted committer’s input. Moreover, ORO permits committing to a message of length $\ell$ while incurring the overhead of committing to a $\kappa$ bit string, where $\ell = \text{poly}(\kappa)$. Compressing the message from $\ell$ to $\kappa$ bits does not break binding since we are in the RO model, where it is hard to find two different messages of arbitrary length s.t. the RO returns the same result upon being queried on those messages. Our protocol involves communication of one $\kappa$ bit string, 3 group elements, computation of 4 exponentiations and 4 random oracle queries to commit to $\ell$ bits. This yields an efficient adaptively secure commitment scheme which is practically motivated for 2PC/MPC protocols based on offline-online paradigm [LR15, RR16]. The setup can be run in the offline phase while the commitment scheme can be conveniently used in the online phase. The ORO in our protocol can be instantiated using a SHA-256 hash function and it can be proven secure in GUC model. This renders our commitment scheme useful in real-life situations where concurrently many protocols are run and share the
same hash function. Table 2 compares our commitment schemes with the recent literature. In terms of empirical costs, our scheme takes approx. 0.18 ms and 0.20 ms to obtain commitment of 128 bits and 2048 bits respectively. In comparison, the scheme of [HM04] takes approx 0.42 ms and 0.49 ms for commitment of 128 and 2048 bits respectively. The protocol of [HM04] requires a non-interactive trapdoor commitment scheme which has been instantiated with Pedersen commitment since to the best of our knowledge such a commitment scheme does not exist based on OWF. The comparison can be viewed in Table 4. Based on the results, we conclude that our scheme is approximately 2 times faster than [HM04] even though we rely on a weaker assumption from the RO.

Table 4: Table indicating implementation results for our Commitment Scheme $\pi_{COM}$ and [HM04] in the LAN setting.

<table>
<thead>
<tr>
<th>Message Size (bits)</th>
<th>Scheme</th>
<th>Runtime of $S$ (ms)</th>
<th>Runtime of $R$ (ms)</th>
<th>Commitment Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>Our Scheme</td>
<td>0.184</td>
<td>0.185</td>
<td>128 + 512</td>
</tr>
<tr>
<td></td>
<td>[HM04]</td>
<td>0.423</td>
<td>0.418</td>
<td>128 + 640</td>
</tr>
<tr>
<td>256</td>
<td>Our Scheme</td>
<td>0.189</td>
<td>0.187</td>
<td>256 + 512</td>
</tr>
<tr>
<td></td>
<td>[HM04]</td>
<td>0.435</td>
<td>0.432</td>
<td>256 + 640</td>
</tr>
<tr>
<td>1024</td>
<td>Our Scheme</td>
<td>0.192</td>
<td>0.191</td>
<td>1024 + 512</td>
</tr>
<tr>
<td></td>
<td>[HM04]</td>
<td>0.455</td>
<td>0.469</td>
<td>1024 + 640</td>
</tr>
<tr>
<td>2048</td>
<td>Our Scheme</td>
<td>0.198</td>
<td>0.202</td>
<td>2048 + 512</td>
</tr>
<tr>
<td></td>
<td>[HM04]</td>
<td>0.487</td>
<td>0.490</td>
<td>2048 + 640</td>
</tr>
</tbody>
</table>

Note: Commitment Size = message size + (commitment & decommitment size)

1.3 Roadmap

The high-level overview of the primitives and notation used is presented in Section 2. In Section 3 we explain the bugs present in the security proofs of concurrent adaptive OT papers[CO15, BDD+17, HL17] in the PRO model. We proceed to the definition of Samplable DME (referred as DME only) in Section 4. Then we present our OT protocol framework in Section 5. The 1-out-of-2 to 1-out-of-N OT transformation is elaborated in Section 6. Section 7 presents the results on OT Extension. Our commitment scheme is covered in Section 8. The implementation details of our protocols have been illustrated in Section 9. Finally, we conclude with prospective future work in Section 10. We provide a summary of UC framework for static and adaptive security in Appendix A for the sake of completeness. We refer to it for better comprehension of notations used in our security proofs. Appendices C and D present instantiations of our DME based on DDH and LWE assumptions.
2 Preliminaries

In this section, we describe the notations used in our protocols and give a high-level overview of the primitives that are used in the paper.

Notations. For the OT and NICOM protocol, we denote the sender by $S$ and receiver by $R$. We denote by $a \leftarrow_R D$ the random sampling of $a$ from a distribution $D$ and the set of elements $\{1, \ldots, n\}$ is represented by $[n]$. Let PRF denote a secure pseudorandom function. A function $\negl(\cdot)$ is said to be negligible, if for every polynomial $p(\cdot)$, there exists a constant $c$, such that for all $n > c$, it holds that $\negl(n) < \frac{1}{p(n)}$. We denote a probabilistic polynomial time algorithm as PPT. We denote the statistical security parameter by $\nu$ and the computational security parameter by $\kappa$. We use $c \approx$ and $s \approx$ to denote computational and statistical indistinguishability, respectively. Let $\mathbb{Z}_p$ denote the field of order $p$, where $p$ is a prime. Let $G$ be the multiplicative group corresponding to $\mathbb{Z}_p^\times$ with generator $g$. For a bit $b \in \{0, 1\}$, we denote $1 - b$ by $\overline{b}$.

Garbled Circuits. The term ‘garbled circuit’ (GC) was coined by Beaver [BMR90] and used extensively only as a technique in secure protocols until they were formalized as a primitive by Bellare et al. [BHR12]. We use notations consistent with [BHR12] (described in Appendix B) for the garbling primitive used to generate CRS for our commitment scheme. Let $GC_k$ denote the $k$th garbled circuit instantiating circuit $C$. We assume that the randomness used for generating circuit $k$ is derived from a $\kappa$-bit random string $\text{seed}_k$ using a PRF. We assume that the fan-in of each gate is 2. We can assume that each AND gate in the circuit has 2 ciphertexts and XOR gates have 0 ciphertexts, using the Half-Gate construction [ZRE15] as the garbling scheme.

Oblivious Transfer. Oblivious transfer (OT) is a protocol between a sender (S) and a receiver (R). In a 1-out-of-2 OT, the sender holds two inputs $a_0, a_1 \in \{0, 1\}^n$ and the receiver holds a choice bit $\sigma$. At the end of the protocol, the receiver obtains $a_{\sigma}$. The sender learns nothing about the choice bit, and the receiver learns nothing about the sender’s other input $a_{\overline{\sigma}}$. The ideal OT functionality $\mathcal{F}_{\text{OT}}$ is recalled below in Figure 1. Similarly a 1-out-of-$N$ OT can be defined as $\mathcal{F}_{\text{N-OT}}$ functionality in Figure 2.

<table>
<thead>
<tr>
<th>$\mathcal{F}_{\text{OT}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose:</strong> On input $(\text{rec}, \text{sid}, \sigma)$ from R where $\sigma \in {0, 1}$; if no message of the form $(\text{rec}, \text{sid}, \sigma)$ has been recorded in the memory, store $(\text{rec}, \text{sid}, \sigma)$ and send $(\text{rec}, \text{sid})$ to S.</td>
</tr>
<tr>
<td><strong>Transfer:</strong> On input $(\text{sen}, \text{sid}, (a_0, a_1))$ from S with $a_0, a_1 \in {0, 1}^n$, if no message of the form $(\text{sen}, \text{sid}, (a_0, a_1))$ is recorded and a message of the form $(\text{rec}, \text{sid}, \sigma)$ is stored, send $(\text{sent}, \text{sid}, a_{\sigma})$ to R and $(\text{sent}, \text{sid})$ to S.</td>
</tr>
</tbody>
</table>
Choose: On input \((\text{rec}, \text{sid}, \sigma)\) from \(R\) where \(\sigma \in \{0, 1\}^\log N\), if no message of the form \((\text{rec}, \text{sid}, \sigma)\) is present in memory, store \((\text{rec}, \text{sid}, \sigma)\) and send \((\text{rec}, \text{sid})\) to \(S\).

Transfer: On input \((\text{sen}, \text{sid}, \{a_j\}_j=1^N)\) from \(S\) with \(a_j \in \{0, 1\}^n\), if no message of the form \((\text{sen}, \text{sid}, \{a_j\}_j=1^N)\) is present in memory and a message of the form \((\text{rec}, \text{sid}, \sigma)\) is stored, send \((\text{sent}, \text{sid}, a_\sigma)\) to \(R\) and \((\text{sent}, \text{sid})\) to \(S\).

**Commitment Schemes.** Commitment schemes allow a party \(S\) to commit to a message \(m\) using randomness \(r\). It keeps the message hidden, while allowing \(S\) to reveal the committed message later. We denote an UC secure commitment to message \(m\) with randomness \(r\) as \(\text{COM}(m; r)\). The ideal commitment functionality \(F_{\text{COM}}\) has been depicted in Fig. 3.

**Random Oracle Model.** A random oracle functionality is parametrized by a domain and a range and it is as \(F_{\text{RO}}\) in Fig. 4. A random oracle query on message \(m\) is denoted by \(F_{\text{RO}}(m)\). The random oracle functionality can be broadly classified \([\text{CDG}^+18]\) into three categories based on its features- plain RO, observable RO and programmable RO. A plain RO returns a random string, from its range, upon being queried on a message \(m\), from its domain. An observable RO inherits the properties of the plain RO but in addition it grants the simulator to observe the queries made, to \(F_{\text{RO}}\), by the adversary. A programmable RO allows the simulator to program \(F_{\text{RO}}(m)\) to return any string from the range, upon being queried on \(m\) for the first time.

### 3 Attack in Concurrent Works on UC-secure Adaptive OT

Concurrent to our work, the works of \([\text{BDD}^+17, \text{HL17}]\) on OT, claim adaptive UC security in the PRO model. The previous version of \([\text{BDD}^+17]\) proposed a general framework for 2-round adaptive OT and provides instantiations under various assumptions such as Learning from Parity with Noise, McEliece cryptosystem, QC-MDPC, Learning With Errors and Computational Diffie Hellman (CDH). We note however that the authors of \([\text{BDD}^+17]\) have fixed their protocol, making it UC secure. \([\text{HL17}]\) proposes a construction of 1-out-of-N OT under the CDH assumption. However, both protocols
Fig. 4: Functionality $\mathcal{F}_{RO}$

$\mathcal{F}_{RO}$ is parameterized by a domain $D$ and range $R$ and it proceeds as follows, running on security parameter $k$:

- $\mathcal{F}_{RO}$ maintains a list $L$ (which is initially empty) of pairs of values $(\hat{m}, \hat{h})$, s.t. $\hat{m} \in D$ and $\hat{h} \in R$.
- Upon receiving a value $(\text{sid}, m)$ (where $m \in D$) perform the following: If there is a pair $(m, \hat{h})$, for some $\hat{h} \in R$, in the list $L$, set $h := \hat{h}$. If there is no such pair, sample $h \leftarrow R$ and store the pair $(m, h)$ in $L$. Once $h$ is set, reply to the activating machine with $(\text{sid}, h)$.

as well as the ‘simplest OT’ construction of [CO15] are prone to a bug when we consider UC-security even against a statically corrupt receiver $R^*$. The attack stems from the late input extraction of a statically-corrupt $R$ as detailed below. The simulator for the case of static corruption of $R$, playing the role of honest $S$, can only extract $R^*$’s input by observing $R^*$’s query to $RO$, made in an attempt to decrypt its chosen message on receiving the last OT message from the sender. This implies that a corrupt $R^*$ can indefinitely delay the input extraction causing composition-related issues.

The delayed input extraction allows us to demonstrate that their constructions do not realise the OT functionality $\mathcal{F}_{OT}$ presented in Fig. 1 where $S$ obtains a notification from the functionality, denoting the end of ideal world execution. Rather, they realise only a weaker version of OT functionality $\mathcal{F}_{wOT}$ as depicted in Fig. 5. In the weaker variant $\mathcal{F}_{wOT}$, $S$ does not obtain any notification from the functionality. Instead its role is limited to sending $(\text{sid}, a_0, a_1)$ to $\mathcal{F}_{wOT}$, after which $S$ halts. However, $\mathcal{F}_{OT}$ is not composable and cannot be used in a bigger protocol to implement the oblivious transfer functionality. Our observation aligns with the work of [LM16], which states (in page 2, last para of Section 1) that the naive OT functionality (Fig. 1 in their paper), same as $\mathcal{F}_{OT}$, is not composable whereas the modified/revised OT functionality (Fig. 3 of their paper), same as our $\mathcal{F}_{wOT}$, can be proven to be composable. The late input extraction problem is referred to as “timing bug” in their paper (page 3, first paragraph). They explain the issue of composability due to timing bug in the naive OT functionality with an example of OT Extension protocol in Section 3. In Section 4, they address the issue by plugging in the revised OT functionality ($\mathcal{F}_{OT}$ in our case). Interestingly, all the currently known UC-secure OT protocols, barring the protocols of [CO15, HL17], implement both $\mathcal{F}_{OT}$ and $\mathcal{F}_{wOT}$ functionalities and hence they are composable. In what follows, we first show that the protocols of [CO15, HL17] do not realise $\mathcal{F}_{OT}$ functionality, but realise only $\mathcal{F}_{wOT}$. Next, we demonstrate the compositional issue of using $\mathcal{F}_{wOT}$ in (yet another example) of 2PC protocol based on garbled circuit (GC) approach.

To show that the constructions that feature delayed input extraction (a.k.a timing bug) do not realise $\mathcal{F}_{OT}$ functionality, we consider an adversarial strategy where $R^*$ does not decrypt the last OT message. In the ideal world, Sim will not be able to extract $R^*$’s input as $R^*$ does not proceed to decrypting its chosen message from the last OT message. Consequently, Sim fails to invoke $\mathcal{F}_{OT}$ functionality with $R^*$’s input and as a result, the $\mathcal{F}_{OT}$ functionality keeps running. This causes a honest $S$ in the ideal
world keep waiting for notification from the $F_{OT}$ functionality in order to terminate. Therefore, while honest $S$ does not halt in the ideal world, it halts in the real world immediately after sending its last message. This difference in the behavior of honest $S$ can be used to distinguish between the two worlds by an environment. With $F^w_{OT}$ functionality, in the scenario mentioned above, an honest $S$ would halt in both the worlds, preserving indistinguishability.

We now illustrate the compositional issue resulted from using $F^w_{OT}$ by means of a GC based 2PC protocol in the OT-hybrid model, where the evaluator $E$ first involves with the constructor in a set of OT functionalities to choose the circuits that will be used for evaluation and respectively for checking and on completion of OTs, the constructor sends the GCs to the evaluator. When $F_{OT}$ was used, a simulator against a corrupt evaluator $E^*$, extracts the inputs of $E^*$ from the OT and either constructs a fake/simulated GC or a real GC based on the extracted input of $E^*$. However in $F^w_{OT}$-hybrid model, the simulator for a corrupt evaluator $E^*$ (playing the role of $R^*$) cannot exact the $E^*$’s input bits to OT and hence cannot substitute certain (evaluation to be specific) GCs with simulated ones (without getting caught with high probability). This difference would enable the environment $Z$ to distinguish between both worlds based on $E^*$’s view.

4 Samplable Dual-Mode Encryption

The seminal paper of [PVW08] introduced the primitive of dual mode encryption (DME). It works like a regular public key encryption scheme, alongside a notion of encryption branches. The key generation algorithm, takes a branch $\sigma \in \{0, 1\}$ as an input, alongside the crs, to generate the public and secret key pair $(pk, sk)$. Messages encrypted using $pk$, on branch $\sigma$, can be decrypted using $sk$, whereas the messages encrypted on the other branch remains hidden under certain conditions, described next. The encryption scheme can be further initialized in either of the two modes - messy or decryption mode, based on the setup phase. The setup phase is invoked with the mode and it returns $(crs, t)$ to the invoking party, where $t$ is the trapdoor for the crs. If the mode is initialized to messy, then the message encrypted on branch $1 - \sigma$ remains statistically hidden. Also, it is possible to extract the branch value $1 - \sigma$ (and $\sigma$ can be computed) given $pk$ and $t$. Whereas, if the scheme is set to decryption mode, then it is possible to generate secret keys $sk_0$ and $sk_1$ which decrypts ciphertexts on branches 0 and 1 respectively. Formally, a dual mode encryption scheme is defined as follows:
- \((\text{crs}, t) \leftarrow \text{SetupMessy}(1^n)\): It is a randomized algorithm that takes as input the security parameter \(\kappa\) and outputs the \(\text{crs}\) and the trapdoor information \(t\), for the messy mode. It enables the invocation of \FindMessy\ algorithm.

- \((\text{crs}, t) \leftarrow \text{SetupDec}(1^n)\): It is a randomized algorithm that takes as input the security parameter \(\kappa\) and outputs the \(\text{crs}\) and the trapdoor information \(t\), for the decryption mode. It enables the invocation of \DecKeyGen\ algorithm.

- \((\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{crs}, \sigma)\): It is a randomized algorithm that takes as input the \(\text{crs}\) and the branch \(\sigma\) and returns the public/encryption and secret/decryption key pair \((\text{pk}, \text{sk})\) for branch \(\sigma\).

- \((y, r) \leftarrow \text{Enc}(\text{pk}, \sigma, m)\): It is a randomized algorithms that takes as input \(\text{pk}\), the branch \(\sigma\) and the message \(m \in \{0, 1\}^\kappa\). It returns the ciphertext \(y\), encrypted on branch \(\sigma\) and the randomness \(r\), used for the encryption.

- \(m \leftarrow \text{Dec}(\text{sk}, y)\): It is a deterministic algorithm that takes in input \(\text{sk}\) and \(y\) and it returns \(m\) only if \(\text{sk}\) and \(y\) correspond to the same branch.

- \(b \leftarrow \text{FindMessy}(\text{pk}, t)\): It returns the branch value \(b \in \{0, 1\}\) given \(\text{pk}\) and the trapdoor \(t\), when the mode is set to messy. The message encrypted on this branch is statistically hidden.

- \((\text{pk}, \text{sk}_0, \text{sk}_1) \leftarrow \text{DecKeyGen}(t)\): It is a randomized algorithm that generates a key pair \((\text{pk}, \text{sk}_0, \text{sk}_1)\) when it is invoked with the trapdoor \(t\) as input and the mode is set to decryption. The secret key \(\text{sk}_b, b \in \{0, 1\}\), enables decryption of ciphertexts on branch \(b\).

The above defined scheme satisfies correctness and four security properties, as mentioned in the [PVW08] paper. In addition, the dual mode encryption scheme must satisfy another property for our OT protocol. It must be possible to generate a \(\text{crs}\) for the messy mode from a random oracle query with overwhelming probability. We outline the correctness and security properties as follows:

- **Correctness**: For every mode, for all \((\text{crs}, t) \leftarrow \text{SetupMessy}(1^n) / \text{SetupDec}(1^n)\), for each \(\sigma \in \{0, 1\}\), for each \((\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{crs}, \sigma)\), and for all \(m \in \{0, 1\}^\kappa\), the following holds \(\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, \sigma, m)) = m\).

- **Property 1** (Indistinguishability of modes): The \(\text{crs}\) generated from either modes are indistinguishable, i.e., for all \((\text{crs}, t) \leftarrow \text{SetupMessy}(1^n)\) and for all \((\text{crs}', t') \leftarrow \text{SetupDec}(1^n)\), the following holds \(\text{crs} \approx \text{crs}'\).

- **Property 2** (Indistinguishability of Branches in Messy mode): In messy mode, the public key does not leak the branch value to a computational adversary i.e., for all \(\text{crs}\) generated by \text{SetupMessy}, the following condition holds true- \(\text{KeyGen}(\text{crs}, 0) \approx \text{KeyGen}(\text{crs}, 1)\).

- **Property 3** (Messy Branch Identification): For each \(\text{crs}\) generated by \text{SetupMessy} and for each \(\text{pk}\) returned by \text{KeyGen}(\text{crs}, \sigma), \text{FindMessy}(\text{pk}, t)\) returns the messy branch \(b\), s.t. \(b = 1 - \sigma\). Moreover, any message encrypted on branch \(b\) is statistically hidden, i.e. for every \(m_0, m_1 \in \{0, 1\}^\kappa\), \(\text{Enc}(\text{pk}, b, m_0) \approx \text{Enc}(\text{pk}, b, m_1)\).

- **Property 4** (Dual Decryptable Branches in decryption mode): For each \((\text{crs}, t) \leftarrow \text{SetupDec}(1^n)\) and for each \((\text{pk}, \text{sk}_0, \text{sk}_1) \leftarrow \text{DecKeyGen}(t)\), the following three conditions hold for all \(m, m_0, m_1 \in \{0, 1\}^\kappa\):
i. \((pk, sk_0) \overset{\$}{\approx} \text{KeyGen}(\text{crs}, 0), (pk, sk_1) \overset{\$}{\approx} \text{KeyGen}(\text{crs}, 1)\) and \((pk, sk_0) \overset{\$}{\approx} (pk, sk_1)\).
ii. \(\text{Dec}(sk, \text{Enc}(pk, b, m)) = m\) for all \(b \in \{0, 1\}\).
iii. \(\text{Enc}(pk, b, m_0) \overset{\$}{\approx} \text{Enc}(pk, b, m_1)\), for all \(b \in \{0, 1\}\).

- **Property 5** (Samplable crs in Messy mode): for all \(c \in \{0, 1\}^\kappa\), \((\text{crs}, t) \leftarrow \text{SetupMessy}(1^\kappa)\), the random oracle query \(F_{RO}(c)\) is identically distributed to the crs of messy mode except with negligible probability i.e., \(F_{RO}(c) \overset{\inf}{\approx} \text{crs}\).

The above DME scheme can be instantiated under the DDH and the LWE assumption, which has been provided in Appendices C and D respectively. The paper of [PVW08] provides proof for correctness and Properties 1-4 of DME instantiations under the two respective assumptions. We further demonstrate, in Appendices C and D, that they satisfy Property 5 too, thereby yielding a samplable DME.

5 Framework for Adaptive Oblivious Transfer

In this section, we present our round-optimal framework for adaptive OT given a DME scheme, random oracles \(F_{RO1}\) and \(F_{RO2}\). We first present a brief overview of our protocol and then we present our proof. \(R\) generates the crs by sampling a random string \(c\) and invoking the random oracle on \(c\). \(R\) obtains a valid crs, in messy mode, except with negligible probability. He invokes the KeyGen with the crs and his choice bit \(\sigma\) to obtain a key pair \((pk, sk)\) which would allow decryption on branch \(\sigma\) using \(sk\). \(R\) sends \(c\) and \(pk\) to \(S\). Property 2 ensures that \(pk\) does not leak about \(\sigma\). \(S\) generates the crs using \(c\) and encrypts two random pads on both branches using \(pk\). Finally, \(S\) encrypts his messages using RO queries on the pads. \(R\) can decrypt the pad corresponding to branch \(\sigma\), due to correctness of the DME scheme, and thus obtain the message corresponding to choice bit \(\sigma\). The framework has been presented as protocol \(\pi_{OT}\) in Fig. 6.

5.1 Static Security

To make the proof of adaptive security more comprehensible, we first prove that \(\pi_{OT}\) realizes the ideal functionality \(F_{OT}\) (Fig. 1) in presence of static adversaries. This is extended to adaptive security in Section 5.2.

In order to prove static security, we describe a simulator \(\text{Sim}\) who behaves as the ideal world adversary and generates a view of \(Z\) which is indistinguishable from the view generated by the real world adversary \(\text{Adv}\) in the real world. It does so by invoking \(F_{OT}\), on behalf of the adversary in ideal world, and running a copy of \(\text{Adv}\) internally, in the head. We denote this internal adversary as \(\text{Adv}_{\text{int}}\). \(\text{Sim}\) simulates the role of the honest parties and the environment to \(\text{Adv}_{\text{int}}\) in the internal execution. Whenever \(\text{Adv}\) corrupts a party in the real world, \(\text{Adv}_{\text{int}}\) also corrupts that party in the internal execution and \(\text{Sim}\) corrupts that party in the ideal world. At the end of the protocol \(\text{Adv}_{\text{int}}\) forwards its view to \(\text{Sim}\) who forwards it to \(Z\) as its ideal world view. We refer to Appendix A for clarity of \(\text{Adv}_{\text{int}}\) notation and details of static security in the UC model [Can01].

For static security, we prove Theorem 1 by considering the four exhaustive corruption cases namely: (1) Both \(S\) and \(R\) are honest (2) \(S^*\) is corrupt while \(R\) is honest (3) \(S\) is
Choose:
- $R$ samples $c \leftarrow \{0, 1\}^\kappa$.
- $R$ generates $\text{crs} \leftarrow F_{\text{RO1}}(\text{sid} || c)$.
- $R$ computes $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{crs}, \sigma)$.
- $R$ sends $(c, \text{pk})$ to $S$.

Transfer:
- $S$ generates $\text{crs} \leftarrow F_{\text{RO1}}(\text{sid} || c)$.
- $S$ samples $r_0, r_1 \leftarrow \{0, 1\}^\kappa$.
- $S$ computes $s_b \leftarrow \text{Enc}(\text{pk}, b, r_b)$, for $b \in \{0, 1\}$.
- $S$ sets $y_0 = F_{\text{RO2}}(\text{sid} || r_0) \oplus a_0$ and $y_1 = F_{\text{RO2}}(\text{sid} || r_1) \oplus a_1$.
- $S$ sends $(s_0, y_0), (s_1, y_1)$ to $R$.

Local Computation by $R$:
- Computes $r_\sigma = \text{Dec}($sk$,$ $s_\sigma$) and outputs $a_\sigma = y_\sigma \oplus F_{\text{RO2}}(\text{sid} || r_\sigma)$.

honest while $R^*$ is corrupt (4) Both $S^*$ and $R^*$ are corrupt. In each of the above cases we describe a simulator $\text{Sim}$ and show that the real world view of $Z$ is indistinguishable from its ideal world view.

We first give a brief intuition of the security proof. Revisiting the proof of security for a static adversary [PVW08], note that $\text{Sim}$ sets the $\text{crs}$ in accordance with which party is corrupt, to enable input extraction of $\text{Adv}_{\text{Int}}$ in the internal execution. More specifically, while $\text{crs}$ of the internal execution is set to messy mode when $R$ is corrupted, it is set to decryption mode when $S$ is corrupted. $\text{Sim}$ can perform this by invoking $\text{SetupMessy}$ (or $\text{SetupDec}$) to obtain $(\text{crs}, t)$ and then program the random oracle to return the same $\text{crs}$. In the former case, $\text{FindMessy}$ can be suitably invoked to extract $\sigma$. In the latter case, $\text{Sim}$ can invoke $\text{DecKeyGen}$ to obtain secret keys on both branches and unlock both messages using the secret keys. The simulation of our protocol for static security is similar to that of PVW.

We are ready to present the formal proof of Theorem. 1. We design an ideal world adversary $\text{Sim}$ who creates an ideal world view $\text{Ideal}_{F, \text{Sim}, Z}(1^n, z)$ of $Z$ which is indistinguishable from the real world view $\text{Real}_{F, \text{Adv}, Z}(1^n, z)$ of $Z$.

**Theorem 1.** If $\text{DME}$ is a samplable dual mode encryption scheme then protocol $\pi_{\text{OT}}$ securely realizes the $F_{\text{OT}}$ functionality against static active adversaries in $(F_{\text{RO1}}, F_{\text{RO2}})$-hybrid model.

**The Simulator:** We describe the simulator $\text{Sim}$ for each possible case of corruption.

**Case 1. $S$ and $R$ are honest:** In this case $\text{Sim}$ acts on behalf of both parties in the internal execution. At the end, $\text{Adv}_{\text{Int}}$ generates his view without corrupting any party and sends it to $\text{Sim}$ who forwards it to $Z$. 

---

**Fig. 6: Adaptively-Secure Oblivious Transfer Protocol**

<table>
<thead>
<tr>
<th>$\pi_{\text{OT}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionalities:</strong> Random oracles $F_{\text{RO1}} : {0, 1}^{2\kappa} \rightarrow {0, 1}^*$ and $F_{\text{RO2}} : {0, 1}^{2\kappa} \rightarrow {0, 1}^\ell$ respectively.</td>
</tr>
<tr>
<td><strong>Private Inputs:</strong> $S$ has input messages $(a_0, a_1)$ and $R$ has an input bit $\sigma$.</td>
</tr>
</tbody>
</table>
(i) Simulating the crs and R’s message: Sim obtains (crs, t) ← SetupDec. Sim samples c ← \{0, 1\}^s and programs \(F_{RO1}(\text{sid}||c)\) to return crs. Sim invokes DecKeyGen(t) to obtain (pk, sk_0, sk_1) and sends (c, pk) as the first OT message to S on behalf of R in the internal execution.

(ii) Simulating S’s message: Sim sets (s_0, s_1) as per the protocol and (y_0, y_1) are set randomly. Sim sends the simulated message on behalf of S.

(iii) Simulating R’s computation: Sim completes the simulation on behalf of R in the internal world.

Case 2. S* is corrupted and R is honest: In this case, Sim acts on behalf of R in the internal execution. At the end, Adv_{Int} generates the view of S* and sends it to Sim who forwards it to Z.

(i) Simulating the crs and R’s message: Same as Case 1.

(ii) Adv_{Int} plays the role of S* and computes the sender message in the internal world. Adv_{Int} sends the second OT message to R in the internal world.

(iii) Simulating R’s computation: Sim decrypts r_0 and r_1 using secret keys sk_0 and sk_1 and obtains a_0 and a_1 from y_0 and y_1. Sim invokes \(F_{OT}\) with (a_0, a_1) and completes the simulation on behalf of R in the internal world.

Case 3. S is honest and R* is corrupted: In this case, Sim acts on behalf of S in the internal execution. At the end, Adv_{Int} generates the view of R* and sends it to Sim who forwards it to Z.

(i) Simulating the crs: Adv_{Int} plays the role of R* and computes the receiver message in the internal world. Whenever Adv_{Int} queries \(F_{RO1}(\text{sid}||c)\), Sim invokes SetupMessy to obtain (crs, t) and programs \(F_{RO1}(\text{sid}||c)\) to return crs. Sim stores the tuple in a list Q as (sid, c, crs, t). If a query is repeated then Sim returns the crs corresponding to the entry in Q indexed by the sid and c values. Adv_{Int} sends the first OT message to S in the internal world.

(ii) Simulating S’s message: Sim extracts the the choice bit, i.e. \(\sigma\), of R* by invoking FindMessy(pk, t). Sim sends \(\sigma\) to \(F_{OT}\) on behalf of R in the ideal world, obtains \(a_\sigma\) and constructs \((s_\sigma, y_\sigma)\) honestly. On the other hand, \(s_\sigma\) is set honestly and \(y_\sigma\) is set randomly. Finally, Sim computes the sender’s message and sends it to R* in the internal world.

(iii) Adv_{Int} computes on behalf of R* and completes the protocol in the internal world.

Case 4. Both S* and R* are corrupted: This is a trivial case of corruption. Sim invokes Adv_{Int}, who simulates messages of both parties and generates the view internally. At the end of execution, Adv_{Int} sends the generated view to Sim who forwards it to Z.

Indistinguishability: Here we show that the ideal world view \(\text{REAL}_{\mathcal{F}, \text{Adv}, Z}(1^s, z)\) of \(Z\) is indistinguishable from the real world view \(\text{REAL}_{\mathcal{F}, \text{Adv}, Z}(1^s, z)\) of \(Z\). We denote \(\text{REAL}_{\mathcal{F}, \text{Adv}, Z}(1^s, z)\) as hybrid HYB_{R}. The ideal world view of \(Z\) varies based on the case of corruption. For case D (\(D \in [4]\)), we denote the ideal world view as HYB_{J,d}. We prove that HYB_{R} is indistinguishable from the corresponding ideal world view for each of the four exhaustive cases of corruption.
Case 1. S and R are honest: We prove that HYB_R and the ideal world view i.e HYB_I are indistinguishable through a series of intermediate hybrids.

- HYB_1: We consider a hybrid HYB_1 which is same as HYB_R except that here, the crs is generated using SetupDec. Indistinguishability follows from Property 1 of DME, i.e. indistinguishability of the two modes, and random sampling of c. A distinguisher for the hybrids can be used to break Property 1 or guess the exact value of c, both of which happen with negligible probability.

- HYB_2: We consider a hybrid HYB_2 which is same as HYB_1 except that Sim generates pk using DecKeyGen. Indistinguishability follows statistically from Property 4 of DME, i.e. the branch remains statistically hidden when the DME is set to decryption mode.

- HYB_3: We consider a hybrid HYB_3 similar to HYB_2 except that in HYB_3 Sim constructs s_0 and s_1 using r_0 and r_1 whereas y_0 and y_1 are formed using different random pads, r'_0 and r'_1. Indistinguishability between the hybrids follows from Property 4 of DME, i.e. indistinguishability of ciphertexts in decryption mode.

- HYB_I: We consider the ideal world hybrid HYB_I similar to HYB_3 except that in HYB_I, Sim sets y_0 and y_1 randomly. Adv_Int can distinguish between the hybrids only if the values r'_0 or r'_1 are guessed precisely and queried to the random oracle. This event occurs with negligible probability in the random oracle model and as a result indistinguishability between the hybrids follows.

Case 2. S* is corrupted and R is honest: We prove that HYB_R and the ideal world view i.e HYB_I are indistinguishable through an intermediate hybrid.

- HYB_1: We consider hybrid HYB_1 which is same as HYB_1 in previous case. Indistinguishability between HYB_R and HYB_1 follows (similar to the previous case) from Property 1 of DME.

- HYB_I: We consider a hybrid HYB_I similar to HYB_1 except that in HYB_I, Sim decrypts r_0 and r_1 and obtains a_0 and a_1 respectively using secret keys on both branches. Indistinguishability follows from Property 4 of DME, which ensures that Sim can decrypt messages on both branches using the secret keys in the decryption mode.

Case 3. S is honest and R* is corrupted: We prove that HYB_R and the ideal world view i.e HYB_I are indistinguishable through a series of intermediate hybrids.

- HYB_1: We consider a hybrid HYB_1 which is same as HYB_R except here the crs is generated using SetupMessy. Indistinguishability follows from Property 5 and 1, i.e. the crs in the messy mode can be sampled using the random oracle and the crs in the messy mode is indistinguishable from the crs in decryption mode. Another possible way of distinguishing is if the distinguisher can guess the value of the crs without querying c to the random oracle, but that happens with negligible probability, due to the random oracle assumption.

- HYB_2: We consider a hybrid HYB_2 which is same as HYB_1 except that, here, Sim extracts the value of σ, by invoking FindMessy(crs, t), and sends it to FOT and obtains a_σ. Indistinguishability follows from Property 3 of DME.
HYB$_3$ : We consider a hybrid HYB$_3$ similar to HYB$_2$ except that in HYB$_3$, Sim constructs $\sigma_\tau$ using $r_\tau$, whereas $y_\tau$ is formed using different random pad, $r'_\tau$. Indistinguishability between the hybrids follows from the fact that $r'_\tau$ remains statistically hidden, due to Property 3, in the messy mode.

HYB$_{I,3}$ : Finally we consider our ideal world hybrid HYB$_{I,3}$ where $y_\sigma$ is set randomly. Indistinguishability follows from the random oracle assumption since the distinguisher has to guess $r'_\sigma$ to distinguish between the two hybrids and this happens with negligible probability.

Case 4. Both $S^*$ and $R^*$ are corrupted: In this case HYB$_R$ and HYB$_{I,4}$ are generated by Adv and Adv$_{Int}$ after being in control of both honest parties in the real world and internal world respectively. As a result, the two views are identical.

5.2 Adaptive Security

Building upon the proof of static security in the previous section, we now prove that $\pi_{OT}$ securely implements $F_{OT}$ in the presence of adaptive adversaries. We refer to Appendix A for details about the security model. We give a brief overview of the proof and then we present it formally. Following the lines of the static proof, Sim programs the crs of the internal execution to be in messy mode, when the receiver is corrupt in the first round, or in the decryption mode otherwise, to enable extraction of $R$’s input or $S$’s input respectively.

In addition, Sim has to equivocate the view of $R$ (resp. $S$), in the internal execution, when $R$ (resp. $S$) gets corrupted adaptively by Adv$_{Int}$. The proof demands equivocation only when $R$ gets corrupted after sending the first OT message and/or $S$ gets corrupted after sending the second OT message. This is done to ensure that the ideal world views (messages and internal state) of the simulated honest parties (in the internal execution) are consistent with the real world views (messages and internal state) of the actual honest parties else $Z$ can distinguish between the two views. In the first scenario, when $R$ is corrupted after the first OT message is sent, the mode is set to decryption, and Sim can extract secrets keys for both branches, corresponding to $pk$, by invoking DecKeyGen. When $R$ gets corrupted and Sim obtains $\sigma$ as $R$’s input, Sim can provide $sk_\sigma$ as its secret key on branch $\sigma$. Equivocation is successful due to Property 4ii of DME. In the second case when $S$ is corrupted after the second OT message is sent, the random values sent corresponding to $(y_0, y_1)$ by Sim have to be made consistent with sender’s actual input $(a_0, a_1)$. For this, Sim exploits the programmability of $F_{RO2}$ to enforce that $(y_0, y_1)$ decrypts to $(a_0, a_1)$. We are ready to present the proof of Theorem 2.

Theorem 2. If DME is a samplable dual mode encryption scheme then protocol $\pi_{OT}$ securely realizes $F_{OT}$ functionality against adaptive active adversaries in $(F_{RO1}, F_{RO2})$-hybrid model.

Proof. We describe the simulator corresponding to the protocol $\pi_{OT}$, for each possible case of adaptive corruption.
The Simulator: The simulator $\text{Sim}$ that generates the ideal world view, is initialized with input values from $Z$ based on which party is corrupted to facilitate simulation.

Outset of the Protocol: Whenever $\text{Adv}_{\text{Int}}$ queries $F_{\text{RO1}}(\text{sid}||c)$, $\text{Sim}$ invokes the SetupMessy algorithm to obtain $(\text{crs}, t)$ and programs $F_{\text{RO1}}(\text{sid}||c)$ to return $\text{crs}$. $\text{Sim}$ stores the tuple in a list $Q$ as $(\text{sid}, c, \text{crs}, t)$. If a query is repeated then $\text{Sim}$ returns the $\text{crs}$ corresponding to the entry in $Q$ indexed by the sid and $c$ values.

$R$ is honest in the first round: $\text{Sim}$ computes $R$’s message similar to case 1(i) ($R$’s message for $(S, R)$ case) of the static proof. The $\text{crs}$ is set in the decryption mode by programming $F_{\text{RO1}}(\text{sid}||c)$. $\text{Sim}$ invokes $\text{DecKeyGen}$ to obtain $(pk, sk_0, sk_1)$ and he sends $c$ and $pk$ as the first OT message to $\text{Adv}_{\text{Int}}$ on behalf of $R$ in the internal execution.

- $S$ is honest in the second round: $\text{Sim}$ acts on behalf of $S$ in the internal execution.
  - Case 1(A). $R$ is honest in the first round, $S$ is honest in the second round, $R^*$ is corrupted after second OT message: $\text{Sim}$ obtains $\sigma$ and $a_\sigma$ in the ideal world. $\text{Sim}$ equivocates $pk$ by setting $sk_\sigma$ as the secret key on branch $\sigma$. Additionally, $\text{Sim}$ equivocates $y_\sigma$ s.t. $a_\sigma$ can be obtained from it. For this, $\text{Sim}$ programs $F_{\text{RO2}}(\text{sid}||r_\sigma) = a_\sigma \oplus y_\sigma$. At the end of the protocol, $\text{Adv}_{\text{Int}}$ outputs $\bot$ and sends its internal state to $\text{Sim}$ who forwards it to $Z$.

  Post Execution. In case of post execution corruption of $S^*$, $\text{Sim}$ obtains $(a_0, a_1)$ and needs to provide the internal randomness of $S^*$ s.t. $y_0$ and $y_1$ open to $a_0$ and $a_1$. We note that $y_\sigma$ was previously equivocated. Equivocation of $y_\sigma$ is performed by programming $F_{\text{RO2}}(\text{sid}||r_\sigma) = a_\sigma \oplus y_\sigma$. $\text{Sim}$ sends the internal state of $S^*$ to $Z$ who halts with an output.

- Case 1(B). $R$ is honest in the first round, $S$ is honest in the second round, $R$ is honest after second OT message: In this case, $\text{Sim}$ acts on behalf of both parties throughout the protocol in the internal execution. At the end of the protocol, $\text{Adv}_{\text{Int}}$ outputs his random tape as its internal state to $\text{Sim}$ who forwards it to $Z$.

  Post Execution. In case of post execution corruption of $R^*$ and $S^*$, $\text{Sim}$ obtains $(\sigma, a_\sigma)$ and $(a_0, a_1)$, and has to provide the internal randomness of $R^*$ and $S^*$ to $Z$. Equivocation of both views is similar to the previous case (Case 1(A) of adaptive simulation) where the view of $R^*$ is equivocated first and then the view of $S^*$ is equivocated. $\text{Sim}$ sends the equivocated views of $R^*$ and $S^*$ to $Z$, who halts with an output.

- Case 2. $R$ is honest in the first round, $S^*$ is corrupted in the second round: $\text{Sim}$ receives the second message on behalf of R from $S^*$ in the internal execution. Simulation is performed as in Case 2(iii) ($R$’s computation for $(S^*, R)$ case) of static proof. $\text{Adv}_{\text{Int}}$ outputs $\bot$ at the end of the protocol and sends its internal state to $\text{Sim}$ who forwards it to $Z$.

  Post Execution. In case of post execution corruption of $R^*$, $\text{Sim}$ obtains $\sigma$ and $a_\sigma$ and he proceeds like the simulator for case 1(A) of adaptive simulation.

$R^*$ is corrupted in the first round: Whenever $\text{Adv}_{\text{Int}}$ queries $F_{\text{RO1}}(\text{sid}||c)$, $\text{Sim}$ invokes the SetupMessy algorithm to obtain $\text{crs}$ and programs $F_{\text{RO1}}(\text{sid}||c)$ to return
crs. Sim stores the tuple in a list $Q$ as $(sid, c, crs, t)$. If a query is repeated then Sim returns the crs corresponding to entry in $Q$ indexed by the sid and $c$ values. Adv$_{int}$ generates the first OT message which is sent to $S$ in the internal execution. This is similar to Case 3(i) ($R$’s message for $(S, R^*)$ case) in static proof.

- **Case 3. $R^*$ is corrupted in the first round, $S$ is honest in the second round:** Sim receives the first OT message on behalf of $S$ from Adv$_{int}$ controlling $R^*$ in the internal execution. Sim continues simulation as in Case 3(ii) ($S$’s message for $(S, R^*)$ case) of static proof.

  **Post Execution.** In case of post execution corruption of $S^*$, Sim obtains $(a_0, a_1)$ and needs to equivocate $y_\sigma$ such that it opens to $a_\sigma$. Sim performs this by programming $F_{RO^2}(sid||r_\sigma) = y_\sigma \oplus a_\sigma$. The internal state of $S^*$ is revealed to $Z$ who halts with an output.

- **Case 4. $R^*$ is corrupted in the first round, $S^*$ is corrupted in the second round:** This is a trivial case since both parties are corrupted by the adversary. The parties are controlled by Adv/Sim/Adv$_{int}$ in the real/ideal/internal world. Adv$_{int}$ generates the second OT message similar to Case 4 ($(S^*, R^*)$ case) of static proof. At the end of execution, Adv$_{int}$ outputs a special symbol $\bot$ on behalf of the corrupted parties and hands over the internal state to Sim who in turn forwards it to $Z$.

  **Post Execution.** There is no post execution corruption since both parties are corrupted and $Z$ halts with an output computed from the internal state of Adv$_{int}$.

**Indistinguishability:** Here we show that the ideal world view $\text{IDEAL}_{F, \text{Sim}, Z}(1^n, z)$ of $Z$ is indistinguishable from the real world view $\text{REAL}_{F, \text{Adv}, Z}(1^n, z)$ of $Z$. We denote $\text{REAL}_{F, \text{Adv}, Z}(1^n, z)$ as hybrid $\text{HYB}_{R}$ and show that in each simulation case $\text{HYB}_{R}$ is indistinguishable from the ideal world by relying on the static indistinguishability proof.

**Case 1(A). $R$ is honest in the first round, $S$ is honest in the second round, $R^*$ is corrupted after second OT message:** Simulation of both OT messages follows along the same direction as Case 1 of static proof. When $R^*$ gets corrupted after second OT message Sim obtains $\sigma$ and he can provide $sk_\sigma$ as the secret key for branch $\sigma$. Equivocation is successful due to Property 4ii of DME, i.e. $\sigma$ is statistically hidden in $pk$ when the crs is generated in decryption mode. In case of post execution corruption, Sim successfully equivocates $y_\sigma$, s.t. it unlocks $a_\sigma$ by programming $F_{RO^2}(sid||r_\sigma)$. Adv can query $r_\sigma$ to $F_{RO^2}$ only with negligible probability and hence equivocation is successful. This follows from Property 4iii of DME, i.e. indistinguishability of ciphertexts in the decryption mode.

**Case 1(B). $R$ is honest in the first round, $S^*$ is corrupted in the second round, $R$ is honest after second OT message:** In this case we can consider that the post execution corruption occurs in two phases - first $R^*$ gets corrupted and only after that $S^*$ gets corrupted. Then it becomes identical to the previous case (case 1(A) of adaptive proof) and indistinguishability follows from the previous argument.

**Case 2. $R$ is honest in the first round, $S^*$ is corrupted in the second round:** The first message is computed according to Case 1(i) of static proof, which is identical to
Case 2(i) of static proof. The rest of the simulation proceeds as Case 2 of static proof. Indistinguishability follows from the indistinguishability of HYB_{I,2} from HYB_{R} in the static proof. In case of post execution corruption, equivocation of R’s view is possible since the crs is in decryption mode and Sim has secret keys for both branches. sk_{\sigma} can be provided as the secret key for branch \sigma. Indistinguishability follows from Property 4ii of DME.

**Case 3. R^* is corrupted in the first round, S is honest in the second round:** Indistinguishability follows from the indistinguishability of HYB_{I,3} from HYB_{R} in the static proof. In case of post execution corruption equivocation can fail if during the simulation Adv_{int} unlocks r_{\sigma} from s_{\sigma} and queries it to FRQ2. However, this occurs with negligible probability due to Property 3 of DME, i.e. ciphertext indistinguishability in messy mode, and the random oracle assumption.

**Case 4. R^* is corrupted in the first round, S^* is corrupted in the second round:** This is identical to case 4 (both (S^*, R^*) are corrupted) in static proof. Hence, indistinguishability of simulation for this case follows from indistinguishability of HYB_{I,4} from HYB_{R} in the static proof.

### 5.3 Concrete Instantiations and Optimized Protocol based on DDH

We provide concrete instantiations of our protocol by instantiating the DME scheme under the DDH and LWE assumptions in Appendix C and D respectively. Furthermore, we can optimize our protocol, while considering the DDH-based instantiation of the DME. In Figure 7 we present our optimized adaptive OT protocol \( \pi^{\text{DDH}}_{\text{OT}} \) based on the DDH assumption. It differs from the original framework since protocol \( \pi^{\text{DDH}}_{\text{OT}} \) does not send separate messages \((s_0, s_1)\) for encrypting the random pads. Rather, the random pads and the messages are encrypted in the same ciphertext \((w_0, w_1)\). This protocol incurs a computation cost of 11 exponentiations and 5 random oracle queries and it has an optimal round complexity of 2. It requires sending a \(\kappa\) bit string - \(c\), two \(\ell\) bit strings - \((u_0, u_1)\); and 4 group elements - \((g, h)\). Interestingly, it is the first round-optimal adaptively-secure OT protocol and it has an overhead of 5 random oracle queries and \(\kappa\)-bit string communication overhead over the static protocol of [PVW08]. The security of our protocol follows from the security of the OT framework and the DME instantiation presented in Appendix C. Hence we avoid presenting it and summarize it in Theorem 3.

**Theorem 3.** If the Decisional Diffie Hellman problem is hard in group \( \mathbb{G} \) then protocol \( \pi^{\text{DDH}}_{\text{OT}} \) securely realizes the \( F_{\text{OT}} \) functionality against adaptive active adversaries in \((F_{\text{RO1}}, F_{\text{RO2}})\)-hybrid model.

### 5.4 Static Version of our Protocol

We can optimize our adaptive \( \pi_{\text{OT}} \) protocol to obtain an efficient static OT protocol, denoted as \( \pi'_{\text{OT}} \). It is similar to the [PVW08] framework, except here we generate the crs.
Fig. 7: Optimized Adaptively-Secure Oblivious Transfer Protocol from DDH

\[ \pi^\text{DDH}_{\text{RO1}} \]

- **Functionalities**: Random oracles \( F_{\text{RO1}} : \{0,1\}^{2\kappa} \rightarrow 4G \) and \( F_{\text{RO2}} : \{0,1\}^\kappa \times G \rightarrow \{0,1\}^\ell \).
- **Private Inputs**: \( S \) has \( \ell \)-bit input messages \((a_0, a_1)\) and \( R \) has an input bit \( \sigma \).

**Choose:**
- \( R \) samples \( c \leftarrow_R \{0,1\}^\kappa \).
- \( R \) generates \( \text{crs} = (g_0, g_1, h_0, h_1) \), s.t. \( (g_0, g_1, h_0, h_1) \leftarrow F_{\text{RO1}}(\text{sid} || c) \).
- \( R \) sends \( \{c, (g, h)\} \) to \( S \).

**Transfer:**
- \( S \) generates the \( \text{crs} = (g_0, g_1, h_0, h_1) \), s.t. \( (g_0, g_1, h_0, h_1) \leftarrow F_{\text{RO1}}(\text{sid} || c) \).
- \( S \) samples \( r_0, r_1, s_0, s_1 \leftarrow_R Z_p \).
- \( S \) computes \( w_0 = g_0^r h_0^s \) and \( w_1 = g_1^{r+1} h_1^s \).
- \( S \) sets \( w = F_{\text{RO2}}(\text{sid} || g^r h^s) \oplus a_0 \) and \( w_1 = F_{\text{RO2}}(\text{sid} || g^{r+1} h^{s+1}) \oplus a_1 \).
- \( S \) sends \( \{(w_0, w_1), (w_1, w_1)\} \) to \( R \).

**Local Computation by \( R \):**
- Computes \( a_\sigma = w_\sigma \oplus F_{\text{RO2}}(\text{sid} || a_\sigma^r) \).

using a PRO. We can obtain \( \pi'_\text{OT} \) from \( \pi_\text{OT} \) by removing the random oracle invocation \( F_{\text{RO2}} \). \( F_{\text{RO2}} \) is not required since static corruption does not demand equivocation, of sender’s message, by the Sim. The security of our protocol is summarized in Theorem 4 and the proof is similar to the static security proof of \( \pi_\text{OT} \).

**Theorem 4.** If DME is a samplable dual mode encryption scheme then protocol \( \pi_\text{OT} \) securely realizes the \( F_{\text{RO1}} \) functionality against static active adversaries in \( F_{\text{RO1}} \)-hybrid model.

Interestingly, \( \pi'_\text{OT} \) satisfies the property of receiver equivocability, which states that the view of the receiver can be equivocated if the receiver gets adaptively corrupted during/after the protocol execution. We analyze this property as two exhaustive subcases as follows: (1) \( R \) gets corrupted before the first OT message is sent (2) \( R \) gets corrupted after the first OT message is sent. For the first case, the simulator has not sent any message on behalf of \( R \) and hence equivocation is not required. In this case the \( \text{crs} \) is set to messy mode. For the second case, the simulator sets the \( \text{crs} \) in the decryption mode which enables him to equivocate \( R \)'s view on obtaining \( \sigma \) after \( R \) gets corrupted.

6 Adaptively Secure 1-out-of-N Oblivious Transfer.

The work of [NP05] implements \( F_{\text{N-OT}} \), against static adversaries in the \( F_{\text{OT}} \)-hybrid model by relying on pseudorandom functions. \( S \) has \( N \) strings \((a_1, a_2, \ldots, a_N)\) as input and \( R \) has a \( \log N \) bit input choice string \( \sigma \). \( S \) samples \( 2 \log N \) random pads \( p_0^i, p_1^i \leftarrow \{0,1\}^\kappa \) for \( i \in [\log N] \). The two parties invoke \( \log N \) copies of \( F_{\text{OT}} \) with \( S \)'s input as \( (p_0^i, p_1^i) \) and \( R \)'s input as \( \sigma_i \) respectively. The \( i \)-th invocation of \( F_{\text{OT}} \) outputs \( p_{\sigma_i}^i \) to \( R \), i.e...
the pad corresponding to his $\sigma_i$ choice bit. Finally, $S$ encrypts $a_j$ as $w_j$ using the pads corresponding to the bit representation of $j$, $j \in [N]$. The message $a_j$ is encrypted as follows:

$$w_j = a_j \oplus \bigoplus_{i=1}^{\log N} \text{PRF}_{p_i}^j(j),$$

where $c = j$, i.e. the $i$th bit of string $j$. $R$ obtains $(w_0, w_1, \ldots, w_N)$ and decrypts $w_\sigma$ for which he possesses all the $\log N$ pads. For other $w$ values, $R$ lacks at least one random pad and security follows by applying PRF on that secret random pad. The transformation communicates $\log N$ ciphertexts, and requires $\log N$ invocations of $F_{\text{OT}}$ and $N \log N$ evaluations of a PRF. It guarantees security against a statically corrupted active adversary.

We show in Fig. 8 that the same transformation can be made adaptively-secure (by protocol $\pi_{N\text{-OT}}$) by implementing the underlying $F_{\text{OT}}$ functionality in an adaptive secure manner. In addition, we replace PRF with a programmable random oracle $F_{\text{RO}}$ and modify the formation of $w_j$ as follows, where $j_i$ denotes the $i$th bit of $j$.

$$w_j = a_j \oplus F_{\text{RO}}(\text{sid}||j,p_1^1,||p_2^2,||p_3^{\log N}) = a_j \oplus F_{\text{RO}}(\text{sid}||j,v_j),$$

where $v_j = (p_1^1,||p_2^2,||\ldots||p_{\log N}^j)$. This reduces the $N \log N$ PRF evaluations to $\log N$ random oracle queries. Later, in this section we further demonstrate that the underlying $F_{\text{OT}}$ need not be full adaptively secure. Static OTs with security against adaptive receivers, i.e. receiver equivocal OTs, suffice for the transformation. Hence, we obtain adaptively-secure 1-out-of-N from $\log N$ 1-out-of-2 receiver equivocal OTs.

### 6.1 Security

The security of the protocol is proven by constructing a simulator $\text{Sim}$ for $\pi_{N\text{-OT}}$. We first present the static proof and then discuss the adaptive proof. For a statically corrupted $S^*$, $\text{Sim}$ can extract the pads by invoking the simulator for $F_{\text{OT}}$. $\text{Sim}$ forms $\{v_j\}_{j \in [N]}$ and decrypts $\{w_j\}_{j \in [N]}$ to unlock all the input messages of $S^*$, i.e. $\{a_j\}_{j \in [N]}$, and completes the simulation by invoking $F_{N\text{-OT}}$ with the input messages. Indistinguishability follows from the $F_{\text{OT}}$ hybrid model. For a statically corrupted $R^*$, $\text{Sim}$ can extract the choice bit $\sigma_i$ by invoking the simulator for $i$th copy of $F_{\text{OT}}$. $\text{Sim}$ reconstructs $\sigma$ and invokes $F_{N\text{-OT}}$ with $\sigma$ to obtain $a_\sigma$. $\text{Sim}$ concludes the simulation by setting $w_\sigma$ correctly while $\{w_j\}_{j \notin [N]/\sigma}$ are set as random strings. $R^*$ obtains all the random pads (corresponding to $\sigma$) from $\log N$ invocations of $F_{\text{OT}}$ and constructs $v_\sigma$ to unlock $a_\sigma$. The other $\{a_j\}_{j \notin [N]/\sigma}$ values remain hidden since $R^*$ lacks at least one random pad for $\{v_j\}_{j \notin [N]/\sigma}$, and hence $F_{\text{RO}}(j,v_j)$ will be indistinguishable from a random string, except with negligible probability, due to the random oracle assumption.

In order to prove adaptivity, we need the property of equivocation. The simulated messages have to be equivocated appropriately so that they are indistinguishable from the real world messages of honest parties. $\text{Sim}$ can equivocate the simulated message (consisting of only $F_{\text{OT}}$ messages) of $R$ by invoking the adaptive simulator for $F_{\text{OT}}$. The simulated message of $S$ consists of the $F_{\text{OT}}$ messages and the simulated ciphertexts.
The $\mathcal{F}_{\text{OT}}$ messages can be trivially equivocated by invoking the adaptive simulator for $\mathcal{F}_{\text{OT}}$. The simulated ciphertexts can be equivocated by programming $\mathcal{F}_{\text{RO}}(\text{sid}||j, v_j) = w_j \oplus a_j$, for $j \in [N]/\sigma$. Adversary can query $\mathcal{F}_{\text{RO}}(\text{sid}||j, v_j)$ with negligible probability since he lacks at least one pad in $v_j$ and hence simulator can program $\mathcal{F}_{\text{RO}}(\text{sid}||j, v_j)$ successfully to equivocate correctly. The proof of indistinguishability is similar to the one for statically corrupted $\mathcal{R}^*$. 

![Adaptively-Secure 1-out-of-N Oblivious Transfer Protocol](image)

**Fig. 8: Adaptively-Secure 1-out-of-N Oblivious Transfer Protocol**

<table>
<thead>
<tr>
<th>$\pi_{N}$-OT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionalities:</strong> Random oracle $\mathcal{F}<em>{\text{RO}} : {0, 1}^{\kappa + \log N(1+\kappa)} \rightarrow {0, 1}^n$. $\mathcal{F}</em>{\text{OT}}$ denotes a 1-out-of-2 OT functionality.</td>
</tr>
<tr>
<td><strong>Private Inputs:</strong> $\mathcal{S}$ has input messages $(a_j)_{j=1}^N$ and $\mathcal{R}$ has an input choice string $\sigma$, where $</td>
</tr>
</tbody>
</table>

**Choose:**
- $\mathcal{R}$ invokes $\mathcal{F}_{\text{OT}}$ functionality $\log N$ times. He invokes the $i$th copy of $\mathcal{F}_{\text{OT}}$ with input $(\text{rec}, i, \sigma_i)$ for $i \in [\log N]$. 

**Transfer:**
- $\mathcal{S}$ samples $2 \log N$ random pads $(p^0_i, p^1_i)$, where $p^0_i, p^1_i \in \{0, 1\}^\kappa$. 
- $\mathcal{S}$ invokes the $i$th copy of $\mathcal{F}_{\text{OT}}$ with input $(\text{sen}, i, (p^0_i, p^1_i))$. 
- $\mathcal{S}$ encrypts his input messages as $w_j = a_j \oplus \mathcal{F}_{\text{RO}}(\text{sid}||j, p^0_j, ||p^2_j, ||...||p^{\log N}_j)$, for $j \in [N]$. 

**Local Computation by $\mathcal{R}$:**
- Computes $a_\sigma$ as $a_\sigma = w_\sigma \oplus \mathcal{F}_{\text{RO}}(\text{sid}||\sigma, p^{0}_\sigma, ||p^2_\sigma, ||...||p^{\log N}_\sigma)$.

### 6.2 Efficiency

Our protocol invokes $\mathcal{F}_{\text{OT}}$ functionality $\log N$ times and queries $\mathcal{F}_{\text{RO}}$ for $N$ times. It incurs communication of $\log N$ copies of $\mathcal{F}_{\text{OT}}$ and $N$ ciphertexts of size $n$ bits.

### 6.3 Instantiating $\mathcal{F}_{\text{OT}}$ using Static Receiver Equivocal OT

We observe that our transformation continues to be adaptively-secure, despite replacing the adaptively-secure 1-out-of-2 OTs in the above result with statically-secure OTs, that support equivocation of receiver’s view irrespective of equivocation of sender’s view. When $\mathcal{S}^*$ or $\mathcal{R}^*$ is corrupted, inputs are extracted by invoking the simulator for the actively-secure static OT protocol. For adaptive security, we need equivocation of $\mathcal{S}$’s and $\mathcal{R}$’s views if corruption occurs during the course of execution or at the end of protocol. In the transformation, $\mathcal{S}$’s view consists of the OT messages and ciphertexts, i.e. $\{w_j\}_{j \in [N]}$. A simulator, playing the role of $\mathcal{S}$ can trivially simulate the OT messages by running the honest $\mathcal{S}$ algorithm with random pads, which are independent of $\mathcal{S}$’s inputs. The ciphertexts are equivocated by programming $\mathcal{F}_{\text{RO}}(\text{sid}||j, v_j)$ (see Section 6.1). On the other hand, $\mathcal{R}$’s view consists only of the OT messages which can be
trivially equivocated by relying on the receiver equivocal property of the OT. Our $\pi_\text{OT}'$ protocol (Section 5.4) is a statically-secure protocol satisfying receiver equivocal property; hence we can plug in our protocol to obtain adaptively-secure 1-out-of-$N$ OTs. The security of $\pi_{N,\text{OT}}$ is summarized in Theorem 5.

**Theorem 5.** If $F_{\text{RO}}$ is a programmable random oracle and $F_{\text{OT}}$ is a static actively secure OT functionality satisfying receiver equivocation, then $\pi_{N,\text{OT}}$ UC-securely realizes the $F_{N,\text{OT}}$ functionality in the $F_{\text{OT}}$ hybrid model against adaptive (without erasures) active adversaries.

## 7 Adaptively Secure Oblivious Transfer Extension

In this section we present our semi-honest and actively secure OT Extension protocols which are secure against adaptive adversaries, without assuming erasures.

### 7.1 Adaptively Secure OT Extension against Semi-Honest Adversaries

We prove that the semi-honest OT Extension protocol of [ALSZ13] (denoted as ALSZ13 hereby) can be made adaptively secure using PRO. We initiate our proof by briefly recalling the protocol of ALSZ13. In ALSZ13, $S$ has $m$ pairs of messages - $\{x_{j,0}, x_{j,1}\}$ for $j \in [m]$, and $R$ has $m$ selection bits, denoted as vector $r = (r_1, \cdots , r_m)$. $R$ samples two random strings $(k_0^j, k_1^j)$ and $S$ samples a random string $s_i$, $i \in [k]$. $R$ and $S$ performs the seed OT phase by invoking $F_{\text{OT}}$ on inputs $(k_0^j, k_1^j)$ and $s_i$ as sender and receiver respectively. $S$ obtains $k_0^i$ from $F_{\text{OT}}$. $R$ computes matrices $B$ and $E$ s.t. $B^i = G(k_0^i)$ and $E^i = r$, where $G$ is a PRG. $R$ sends a correction matrix $D$ to $S$, s.t. $D^i = B^i \oplus G(k_1^i) \oplus E^i$. $S$ forms the matrix $Q$ using $G(k_1^i)$, $s_i$ and $D$. The $j$th row of $Q$ is used to encrypt the $j$th message pair $\{x_{j,0}, x_{j,1}\}$ using pads $H(j, Q_j)$ and $H(j, Q_j \oplus s_r)$ respectively, where $H$ is a correlation robust function [IKNP03]. $S$ sends the ciphertexts to $R$, who decrypts and obtains $x_{j,r_j}$ using $H(j, B_j)$. The ALSZ13 is similar to the protocol presented in Fig. 9, except $F_{\text{RO1}}$ and $F_{\text{RO2}}$ will be replaced by $G$ and $H$. Security against a statically corrupt $S$ follows from the sender privacy of $F_{\text{OT}}$, which ensures that one of $R$’s input (acting as sender in $F_{\text{OT}}$) to $F_{\text{OT}}$ remains hidden from $S$. On the other hand, a statically corrupt $R$ can break honest sender privacy if he obtains both $x_{j,0}$ and $x_{j,1}$ for some $j$. However, this happens with negligible probability since $R$ has to either guess the value of $s_i$ or obtain a collision in the random oracle query, s.t. $H(j, B_j) = H(j, B_j \oplus s)$. We refer to [ALSZ13] for the full proof.

While considering adaptive security, the messages of $S$ and $R$ require equivocation. These messages can be classified into three exhaustive cases. Below, we give a high level overview for equivocating each case:

- $F_{\text{OT}}$ messages: It can be observed that $S$ and $R$ invoke $F_{\text{OT}}$ on random inputs - $s_i$ and $(k_0^i, k_1^i)$ respectively, independent of their actual inputs. This can be leveraged to reduce the assumption on $F_{\text{OT}}$. More specifically, $F_{\text{OT}}$ can be secure only against static adversaries, since the simulator can simulate the view of an honest sender (resp. receiver) by running the honest sender (resp. receiver) algorithm on random inputs. When sender (resp. receiver) gets corrupted, the simulator can produce the
Fig. 9: Adaptive OT Extension Protocol secure against passive adversaries

**Protocol \( \pi^p_{ALSZ} \) for obtaining \((\ell)\)-OT\(_p^m\) from \((\ell)\)-OT\(_m\)**

- **Input of S:** \( m \) pairs \( \{x_{j,0}, x_{j,1}\} \) for \( j \in [m] \).
- **Input of R:** \( m \) selection bit vectors \( \mathbf{r} = (r_1, \ldots, r_m) \) such that each \( r_j \in \{0, 1\} \).
- **Functionalities:** Random Oracles \( \mathcal{F}_{RO1} : \{0, 1\}^{2\kappa} \rightarrow \{0, 1\}^{\kappa} \) and \( \mathcal{F}_{RO2} : \{0, 1\}^{\kappa} \times [m] \times \{0, 1\}^* \rightarrow \{0, 1\}^{\ell} \). \( \mathcal{F}_{OT} \) denotes a 1-out-of-2 OT functionality.

**Notations:**

- \( \mathcal{S} \) chooses \( \kappa \) bit strings: \( \mathcal{S} \) inputs \( \kappa \) pairs of seeds \((\mathcal{S}_0, \mathcal{S}_1)\) each of length \( \kappa \).
- \( \mathcal{R} \) chooses \( \kappa \) pairs of seeds \((\mathcal{R}_0, \mathcal{R}_1)\) each of length \( \kappa \).
- \( \mathcal{R} \) and \( \mathcal{S} \) invocations of the OT functionality for \( \kappa \) times with inputs \((\mathcal{R}_c, \mathcal{S}_0, \mathcal{S}_1)\) and \((\mathcal{R}_c, \mathcal{S}_0, (\mathcal{S}_0^0, \mathcal{S}_0^1))\) respectively. \( \mathcal{S} \) receives \( \{\mathcal{S}_0^i\}\) as output.

**Seed OT Phase:**

1. \( \mathcal{S} \) chooses \( s \leftarrow \{0, 1\}^\kappa \) at random.
2. \( \mathcal{S} \) computes the first message of \( \pi_{OT} \) as \( \pi_{OT}^1(s) \).
3. \( \mathcal{R} \) chooses \( \kappa \) pairs of seeds \((\mathcal{R}_0, \mathcal{R}_1)\) each of length \( \kappa \).
4. \( \mathcal{S} \) and \( \mathcal{R} \) invokes the \( \mathcal{F}_{OT} \) functionality for \( \kappa \) times with inputs \((\mathcal{R}_c, \mathcal{S}_0, \mathcal{S}_1)\) and \((\mathcal{R}_c, \mathcal{S}_0, (\mathcal{S}_0^0, \mathcal{S}_0^1))\) respectively. \( \mathcal{S} \) receives \( \{\mathcal{S}_0^i\} \) as output.

**OT Extension Phase:**

1. \( \mathcal{R} \) forms three \( m \times \kappa \) matrices \( \mathcal{B}, \mathcal{E}, \) and \( \mathcal{C} \) in the following way and sends them to \( \mathcal{S} \):
   - Set \( \mathcal{B} = \mathcal{F}_{RO1}(\mathcal{S}_0^i) \).
   - Set \( \mathcal{E} = (s_i \oplus \mathcal{E}_i) \). Clearly, \( \mathcal{E} = \mathcal{R} \).
   - Set \( \mathcal{D} = \mathcal{B} \oplus \mathcal{F}_{RO1}(\mathcal{S}_0^i) \).
2. On receiving \( \mathcal{D} \), \( \mathcal{S} \) forms a \( m \times \kappa \) matrix \( \mathcal{Q} \) with the \( j \)th column of \( \mathcal{Q} \) set to \( \mathcal{Q}_j =\)
   - \( s_i \oplus \mathcal{D} \). Clearly, \( \mathcal{Q} = (B \oplus (s \oplus \mathcal{E}))(\mathcal{E} \oplus (s \oplus \mathcal{Q})) \).
3. For every \( j \in [m] \), \( \mathcal{S} \) computes \( \mathcal{y}_{j,0} = x_{j,0} \oplus \mathcal{F}_{RO2}(\mathcal{S}_0) \mathcal{Q}_i \) and \( \mathcal{y}_{j,1} = x_{j,1} \oplus \mathcal{F}_{RO2}(\mathcal{S}_0) \mathcal{Q}_i \mathcal{S}_0^i \). \( \mathcal{S} \) sends \( \{\mathcal{y}_{j,0}, \mathcal{y}_{j,1}\} \) to \( \mathcal{R} \).
4. For every \( j \in [m] \), \( \mathcal{R} \) recovers \( \mathcal{z}_j = \mathcal{F}_{RO2}(\mathcal{S}_0) \mathcal{Q}_i \) \( \mathcal{R} \) outputs \( \{\mathcal{z}_j\} \in [m] \).

The view, generated using random inputs, is indistinguishable from the honest sender’s (resp. receiver’s) view.

- \( \mathcal{R} \)’s messages: It consists of matrix \( \mathcal{D} \) which depends on \( \mathcal{R} \)’s input \( \mathcal{R} \). When \( \mathcal{R} \) gets corrupted, the simulator has to equivocate \( \mathcal{R} \)’s message, i.e. matrix \( \mathcal{D} \), based on \( \mathcal{R} \)’s input. This is ensured by replacing \( \mathcal{G} \) with a PRO \( \mathcal{F}_{RO1} \).
- \( \mathcal{S} \)’s messages: It consists of messages - \((\mathcal{y}_{j,0}, \mathcal{y}_{j,1})\), which are dependent on \( \mathcal{S} \’s \) inputs \((x_{j,0}, x_{j,1})\). When \( \mathcal{S} \) gets corrupted, the simulator has to equivocate \( \mathcal{S} \’s \) message based on \( \mathcal{S} \’s \) inputs. This is ensured by replacing \( \mathcal{H} \) with a PRO \( \mathcal{F}_{RO2} \).

The explanation for the last two subcases will be clear from the next few paragraphs, where we elaborate on the equivocation cases. We consider equivocation of \( \mathcal{S} \) and \( \mathcal{R} \) in two separate cases. Our passively secure adaptive OT extension protocol \( \pi^p_{ALSZ} \) is presented in Fig. 9 and the security proof has been summarized in Theorem 6.

**Theorem 6.** The protocol \( \pi^p_{ALSZ} \) UC-securely realizes the Oblivious Transfer Extension functionality in the \((\mathcal{F}_{OT}, \mathcal{F}_{RO1}, \mathcal{F}_{RO2})\) hybrid model against adaptive (without erasures) passive adversaries.

- Equivocation of \( \mathcal{S} \’s \) view: When \( \mathcal{S} \) is honest, \( \sim \) plays the role of \( \mathcal{S} \) in the internal world. The view of \( \mathcal{S} \) consists of his input \( \{x_{j,0}, x_{j,1}\} \), random vector \( s \), matrix
Q and the ciphertexts \((y_{j,0}, y_{j,1})\). The s vector and Q matrix is independent of S’s inputs and can be simulated correctly without equivocation. The ciphertexts are simulated by relying on the programmability property of \(\mathcal{F}_{RO2}\). We first consider the case where both parties are honest. The formal simulation for S is presented as follows. Sim invokes the \(\mathcal{F}_{OT}\) simulator with random input \(s_i\) to simulate the ith seed-OT. Sim generates Q following honest sender algorithm. In the OT Extension phase, Sim sends random values for \(y_{j,b}\), where \(b \in \{0, 1\}\). In case of post execution corruption of S, Sim obtains the inputs of S. Sim provides s and Q as part of the view, and this is indistinguishable from the real world view. In addition, Sim opens \(y_{j,b}\) to \(x_{j,b}\) by programming \(\mathcal{F}_{RO2}\) as follows:

\[
\mathcal{F}_{RO2}(j, Q_j) = y_{j,0} \oplus x_{j,0}, \\
\mathcal{F}_{RO2}(j, Q_j \oplus s) = y_{j,1} \oplus x_{j,1}.
\]

Equivocation is successful due to the randomness of s and random oracle assumption. An adversary \(\text{Adv}_{\text{int}}\) can prevent Sim from programming \(\mathcal{F}_{RO2}\) if he queries \(Q_j\) or \(Q_j \oplus s\) to \(\mathcal{F}_{RO2}\). This happens with negligible probability as Q and s are randomly chosen. This proves that real and ideal world are indistinguishable. Now, we consider the case where \(\text{Adv}_{\text{int}}\) corrupts R*. Sim obtains the choice vector r. Sim invokes the OT Extension functionality with r and obtains \(\{x_{j,r_j}\}_{j \in [m]}\). Sim computes \(y_{j,r_j}\) honestly like an honest sender, s.t. \(y_{j,r_j}\) opens to \(x_{j,r_j}\). In case of post execution corruption, Sim obtains \((x_{j,0}, x_{j,1})\) and he programs \(\mathcal{F}_{RO2}(j, Q_j \oplus (s \odot \tau_j))\), s.t. \(y_{j,\tau_j}\) opens to \(x_{j,\tau_j}\). To prevent equivocation, \(\text{Adv}_{\text{int}}\) has to guess the value of s and query \(Q_j \oplus (s \odot \tau_j) = B_j \oplus s\) to \(\mathcal{F}_{RO2}\) which happens with negligible probability, proving indistinguishability of the two worlds.

- **Equivocation of R’s view:** When R is honest, Sim plays the role of R in the internal world. The view of R consists of his inputs, the seeds \((k^0_i, k^1_i)\) and the matrices, B, E and D. The seeds and B matrix can be simulated without any equivocation since they are independent of R’s input. The other two matrices are simulated by programming \(\mathcal{F}_{RO1}\), since they depend on R’s input. We first consider the case where both parties are uncorrupted. The formal simulation is presented as follows. Sim randomly generates \((k^0_i, k^1_i)\) and simulates ith \(\mathcal{F}_{OT}\) by invoking \(\mathcal{F}_{OT}\) with inputs \((k^0_i, k^1_i)\) like an honest R. Sim sets D matrix randomly and the B and E matrices are constructed later as they are not required for the simulation. Sim sends D to S and ends simulation as both parties are honest. The adversary \(\text{Adv}_{\text{int}}\) (without corrupting S) observes R’s message, which consists only of D matrix. Based on that, he cannot distinguish between real and ideal world since in ideal world D is a random matrix, whereas in real world, it is padded with \(\mathcal{F}_{RO1}(\text{sid}, k^0_i)\) and \(\mathcal{F}_{RO1}(\text{sid}, k^1_i)\), where \(k^0_i\) and \(k^1_i\) remains hidden from \(\text{Adv}_{\text{int}}\). Thus, indistinguishability follows from the random oracle assumption. In case of post execution corruption of R, Sim obtains r and he programs \(\mathcal{F}_{RO}(\text{sid}||k^0_i)\) and \(\mathcal{F}_{RO}(\text{sid}||k^1_i)\) s.t. the following holds true:

\[
\mathcal{F}_{RO1}(k^0_i) \oplus \mathcal{F}_{RO1}(k^1_i) = D^t \oplus r. \tag{1}
\]

The B matrix is constructed after \(\mathcal{F}_{RO1}\) has been programmed according to Eq. 1. Equivocation is successful since \(\text{Adv}_{\text{int}}\) does not obtain \(k^0_i\) and \(k^1_i\) from \(\mathcal{F}_{OT}\) and
he has to guess it, which occurs with negligible probability, in order to prevent equivocation. If $\text{Adv}_{\text{int}}$ corrupts $S^*$, then $\text{Sim}$ obtains $s$ from $\mathcal{F}_{\text{OT}}$ simulator (since $S^*$ is the OT receiver) and he sets $\mathcal{F}_{\text{RO1}}(k_i^s)$ randomly concluding the simulation. In case of post execution corruption of $R^*$, $\text{Sim}$ obtains $r$ and constructs $E$ matrix s.t. $E^i = r$. $\text{Sim}$ equivocates $D$ matrix by programming $\mathcal{F}_{\text{RO1}}(k_i^s)$ as follows:

$$\mathcal{F}_{\text{RO1}}(k_i^s) = \mathcal{F}_{\text{RO1}}(k_i^s) \oplus D^i \oplus r.$$  \hspace{1cm} (2)

The $B$ matrix is constructed after $\mathcal{F}_{\text{RO1}}$ has been programmed according to Eq. 2. Equivocation is successful since $\text{Adv}_{\text{int}}$ does not obtain $k_i^s$ from $\mathcal{F}_{\text{OT}}$ and he has to guess it, in order to prevent equivocation. However, it occurs with negligible probability and hence the two worlds are indistinguishable.

### 7.2 Adaptively Secure OT Extension against Active Adversaries

The ALSZ13 protocol is actively secure against $S^*$ and passively secure against $R^*$, assuming seed OTs to be actively secure. An active $R^*$ can send a bad $D$ matrix such that $Q$ is malformed and it leaks the $s$ vector. We refer to [ALSZ15] (denoted as ALSZ henceforth) for the exact attack. To address this attack, the ALSZ15 protocol introduces column wise consistency check for every pair of columns. Moreover, for active security we require that the simulator $\text{Sim}$ can extract a corrupted party’s input in the protocol-

In order to extract corrupted parties’ inputs and simulate the consistency checks properly, we observe that the $\mathcal{F}_{\text{OT}}$ messages are random in the OT extension protocol of ALSZ13 and ALSZ15. Hence, we can modify the protocol s.t. the parties invoke $\mathcal{F}_{\text{RO}}$ (Fig. 11) instead of $\mathcal{F}_{\text{OT}}$. $\mathcal{F}_{\text{RO}}$ is an OT functionality where $R$ has an input whereas $S$ does not have any input. The functionality returns random messages to $S$ and one of the random messages to $R$ based on his input. Formally stating, $\mathcal{F}_{\text{RO}}$ takes choice bit $\sigma$ as input from receiver of $\mathcal{F}_{\text{RO}}, i.e., R_{\text{RO}},$ and generates two random pads $(a_0, a_1)$ for sender of $\mathcal{F}_{\text{RO}}, i.e., S_{\text{RO}}.$ It sends $a_\sigma$ to $R_{\text{RO}}$ and $(a_0, a_1)$ to $S_{\text{RO}}.$ On a high level, when $\mathcal{F}_{\text{RO}}$ is invoked in our OT Extension protocol the input of $S$ (playing as $R_{\text{OT}}$) to the $\mathcal{F}_{\text{RO}}$ can be extracted by invoking the static simulator for $\mathcal{F}_{\text{RO}}$. The seeds for $R$ (playing the role of $S_{\text{OT}}$) are obtained from $\mathcal{F}_{\text{RO}}$. For equivocation of $R$’s view, $\text{Sim}$ runs the honest $S_{\text{OT}}$ algorithm to obtain random seeds. The consistency checks are simulated by programming $\mathcal{F}_{\text{RO1}}$ and $\mathcal{F}_{\text{RO2}}$. For equivocation of $S$’s view, $\text{Sim}$ invokes the $\mathcal{F}_{\text{RO}}$ simulator for $R_{\text{OT}}$ with a random $s$ vector. Indistinguishability follows since the value of $s$, sampled by honest $S$, is random in real world as well. The ciphertexts $(y_{j,0}, y_{j,1})$ are equivocated by programming $\mathcal{F}_{\text{RO3}}$. Invoking $\mathcal{F}_{\text{RO}}$ has a significant advantage - $\mathcal{F}_{\text{RO}}$ can be adaptively implemented by a static receiver equivocal OT (Sec. 5.4), since $S$ does not have any message, whereas implementing $\mathcal{F}_{\text{OT}}$ in an adaptive way is expensive.

Now we formally describe our protocol $\pi_{\text{ALSZ}}^n$, which has been presented in Fig. 10. Our protocol is same as ALSZ15, except here the PRG, hash and correlation robust
function invocations of ALSZ15 are replaced by PRO invocations. Our protocol has a seed OT phase similar to the $\pi_{ALSZ}^p$ protocol. The OT extension phase is divided into two parts, based on $R$’s and $S$’s role. $R$’s role consist of phase I of OT extension, followed by the consistency checks and finally phase II of OT extension consists of $S$’s role in the OT extension. Similar to ALSZ15, our protocol also contains column wise consistency check for every pair of columns. It has been recently identified by [ALSZ15] team that the checks leak $\kappa^2$ bits of randomness, on $R$’s end. In order to counter that, they add $\kappa^2$ bits of randomness by adding $\kappa$ columns and $\kappa$ rows, as dummy choice bits $\tau$. As a result the new choice bit string is $r' = r||\tau$. The checks ensure that the same $r'$ has been used in all the columns of $D$, i.e. $E_i = r'$ for all $i \in [k]$, where $k = 2\kappa$. Our static proof of security follows from the security proof of [ALSZ15] and we refer to their paper for the static proof against an active adversary. Adaptive security for our protocol can be proved against active adversaries in the PROM model, provided the underlying seed OTs are adaptive actively secure implementation of $F_{RO3}$ (Fig. 11) functionality. Security of $\pi_{ALSZ}^a$ has been summarized in Theorem 7.

**Theorem 7.** If $F_{RO1}$, $F_{RO2}$ and $F_{RO3}$ are programmable random oracles then $\pi_{ALSZ}^a$ UC-securely realizes the Oblivious Transfer Extension functionality in the $F_{RO3}$ hybrid model against adaptive (without erasures) active adversaries.

For proving adaptive security, the views of $S$ and $R$ require equivocation. Equivocation of $S$’s view follows from the equivocation of $S$’s view in ALSZ13 protocol. Equivocation of $R$’s view is similar to that of ALSZ13 but in addition it requires simulating the consistency checks correctly.

- Equivocation of $S$’s view: $Sim$ simulates the seed-OT by invoking the $F_{RO3}$ functionality with a random $s$ to obtain $\{k_i^0\}$, for $i \in [k]$. If both parties are honest then the simulation proceeds similar to the simulation for “Equivocation of $S$’s view” (passive case) in Sec. 7.1. The consistency checks can be trivially simulated since $S$’s role is limited to verification of the checks. If $Adv_{\text{int}}$ corrupts $R'$ then $Sim$ extracts the choice vector $r'$. More specifically, $Sim$ extracts $R'$’s input, $(k_i^0, k_i^1)$, by invoking the $F_{RO3}$ simulator and computes $r' = E_i = r||\tau$ from $D_i$ using the seeds. $Sim$ invokes the OT Extension functionality with $r$ and obtains $\{x_{j,r_j}\}_{j \in [m+n]}$. $Sim$ programs $F_{RO3}(\text{sid}|j, Q_j \oplus (s \circ r_j))$ s.t. $y_{j,r_j}$ opens to $x_{j,r_j}$. In case of post execution corruption, $Sim$ obtains $(x_{j,0}, x_{j,1})$ and he programs $F_{RO3}(\text{sid}|j, Q_j \oplus (s \circ r_j))$, s.t. $y_{j,r_j}$ opens to $x_{j,r_j}$. To prevent equivocation $Adv_{\text{int}}$ has to guess the value of $s$ and query $Q_j \oplus (s \circ r_j) = B_j \oplus s$ to $F_{RO3}$ which happens with negligible probability.

- Equivocation of $R$’s view: The view of $R$ consists of his inputs, the seeds $(k_i^0, k_i^1)$ and the matrices, $B$, $E$ and $D$. When $R$ is honest, $Sim$ plays the role of $R$ in the internal world where $Sim$ obtains $(k_i^0, k_i^1)$ from $F_{RO3}$. If both parties are honest then simulation proceeds similar to the simulation for “Equivocation of $R$’s view” (passive case) in Sec. 7.1. The consistency checks are simulated by setting $h_{a,b}^{\alpha,\beta}$ values randomly, for all $a, b \in \{0, 1\}$ and all pairs $\alpha, \beta \subseteq [k^2]$. In case of post execution corruption, $Sim$ obtains $r$, and then based on the equivocated $B$, $E$ and $D$ matrices, $Sim$ can equivocate the $h_{a,b}^{\alpha,\beta}$ values correctly by programming $F_{RO2}$.
OT Extension Phase II:

Seed OT Phase:

Check Phase:

– Security Parameter: $k = 2\kappa$.

– Functionalities: $\mathcal{F}_{\text{RO1}} : \{0, 1\}^{m+\kappa} \rightarrow \{0, 1\}^{m+\kappa}$, $\mathcal{F}_{\text{RO2}} : \{0, 1\}^m \times \{0, 1\}^k \rightarrow \{0, 1\}^k$ respectively. $\mathcal{F}_{\text{OT}}$ denotes OT functionality.

– Notations: In the protocol $j \in [m + \kappa]$ and $i \in [k]$, unless specified otherwise.

Seed OT Phase:

1. $S$ samples $s \in \{0, 1\}^k$ and invokes $\mathcal{F}_{\text{RO1}}$ with message $(\text{rec}, (\kappa, s_i))$ for $k$ times to obtain $\{k^a_i\}$ seeds.

2. $R$ invokes $\mathcal{F}_{\text{RO1}}$ with message $(\text{sen}, (\text{sid}, (\text{transfer}, \kappa)))$ for $k$ times to obtain $\{(k^a_0, k^1_0)\}$ seeds.

OT Extension Phase I:

1. $R$ forms three $(m + \kappa) \times k$ matrices $B$, $E$ and $C$ in the following way and sends $D$ to $S$:
   - Sets $B^i = \mathcal{F}_{\text{RO1}}(\text{sid}||k^a_i)$.
   - Samples $\tau \leftarrow \{0, 1\}^k$ and sets $r' = r || \tau$.
   - Sets $E^i = (r'_j || \ldots || r'_j)$. Clearly, $E^i = r'$.
   - Set $D^i = B^i \oplus \mathcal{F}_{\text{RO1}}(\text{sid}||k^a_i) \oplus E^i$.

2. On receiving $D$, $S$ forms $(m + \kappa) \times k$ matrix $Q$ with the $j$th column of $Q$ set as $Q_j^i = \{s_i \oplus D^j\lbrack \mathcal{F}_{\text{RO1}}(k^a_i)\rbrack \}$. Clearly, (i) $Q^i = (B^i \oplus (s \circ E^i))$ and (ii) $Q_i = (B_j \oplus (s \circ r_j))$.

Check Phase:

1. For every pair $\alpha, \beta \subseteq [k^2]$ and $a, b \in \{0, 1\}$, $R$ defines the following four values:

   $h_{\alpha, \beta}^{a, b} = \mathcal{F}_{\text{RO2}}(\text{sid}||\mathcal{F}_{\text{RO1}}(\text{sid}||k^a_\alpha) \oplus \mathcal{F}_{\text{RO1}}(\text{sid}||k^b_\beta))$

   $R$ sends each $h_{\alpha, \beta}^{a, b}$ to $S$.

2. For every pair $\alpha, \beta \subseteq [k^2]$, $S$ knows $s_\alpha$, $s_\beta$, $k^a_\alpha$, $k^b_\beta$, $D^a$ and $D^b$. $S$ checks that:
   (a) $h_{\alpha, \beta}^{a, b} = \mathcal{F}_{\text{RO2}}(\text{sid}||\mathcal{F}_{\text{RO1}}(\text{sid}||k^a_\alpha) \oplus \mathcal{F}_{\text{RO1}}(\text{sid}||k^b_\beta))$.
   (b) $h_{\alpha, \beta}^{a, b} = \mathcal{F}_{\text{RO2}}(\text{sid}||\mathcal{F}_{\text{RO1}}(\text{sid}||k^a_\alpha) \oplus \mathcal{F}_{\text{RO1}}(\text{sid}||k^b_\beta) \oplus D^a \oplus D^b) = \mathcal{F}_{\text{RO2}}(\text{sid}||\mathcal{F}_{\text{RO1}}(\text{sid}||k^a_\alpha) \oplus \mathcal{F}_{\text{RO1}}(\text{sid}||k^b_\beta) \oplus r^a \oplus r^b)$, where $r^a$, $r^b$ denotes $r$ used in $D^a$, $D^b$ respectively.
   (c) $D^a \neq D^b$.

   In case one of these checks fails, $S$ aborts and outputs $\perp$.

OT Extension Phase II:

1. For every $j \in [m]$, $S$ computes $y_{j,0} = x_{j,0} \oplus \mathcal{F}_{\text{RO3}}(\text{sid}||j, Q_j)$ and $y_{j,1} = x_{j,1} \oplus \mathcal{F}_{\text{RO3}}(\text{sid}||j, Q_j \oplus s)$. $S$ sends $\{y_{j,0}, y_{j,1}\}_{j \in [m]}$ to $R$.

2. For every $j \in [m]$, $R$ recovers $z_j = y_{j, r_j} \oplus \mathcal{F}_{\text{RO3}}(\text{sid}||j, B_j)$. $R$ outputs $\{z_j\}_{j \in [m]}$. 

Fig. 10: Adaptive OT Extension Protocol secure against active adversaries
Fig. 11: The ideal functionality $F_{\text{ROT}}$ for Random Oblivious Transfer

$F_{\text{ROT}}$

**Initiate:** On input $(\text{rec}, \text{sid}, (\ell, \sigma))$ from $R$; if no message of the form $(\text{sid}, (\ell, \sigma))$ is present in memory, store $(\text{sid}, (\ell, \sigma))$. Send $(\text{rec}, \text{sid})$ to $S$.

**Transfer:** On input $(\text{sen}, \text{sid}, (\text{transfer}, \ell))$ from $S$, if no message of the form $(\text{sid}, (\ell, \sigma))$ is present in memory, then abort. Else sample $a_0, a_1 \leftarrow_R \{0, 1\}^\ell$. Send $(\text{sent}, \text{sid}, a_0)$ to $R$ and $(\text{sent}, \text{sid}, (a_0, a_1))$ to $S$.

**Corruption:** If $\text{Adv}$ corrupts $S^*$ then receive $(a_0, a_1)$ from $\text{Adv}$ and continue execution as above using these values. If $\text{Adv}$ corrupts $R^*$ the usual execution continues.

$\text{Adv}_{\text{int}}$ lacks $k_0^\alpha$ and $k_1^\beta$, hence he cannot query the required argument to $F_{\text{RO2}}$ s.t. it would prevent equivocation of $h_{\alpha, \beta}^{a,b}$. Thus, the consistency checks can be correctly simulated. If $\text{Adv}_{\text{int}}$ corrupts $S^*$ then $\text{Sim}$ extracts $s$ vector from $F_{\text{ROT}}$ simulator. $S$ computes $F_{\text{RO1}}(\text{sid}||k^{\alpha}_s)$ and simulates the $B$, $E$ and $D$ matrices according to the simulation strategy in Section 7.1. For simulating the consistency checks the $h_{\alpha, \beta}^{a,b}$ values are randomly set, for all $a,b \in \{0,1\}$. For every pair $\alpha, \beta \subseteq [k^2]$, $\text{Sim}$ programs $F_{\text{RO2}}$ as follows:

$$F_{\text{RO2}}(\text{sid}||F_{\text{RO1}}(\text{sid}||k^{\alpha}_s) \oplus F_{\text{RO1}}(\text{sid}||k^{\beta}_s)) = h_{\alpha, \beta}^{s, s}$$

$$F_{\text{RO2}}(\text{sid}||F_{\text{RO1}}(\text{sid}||k^{\alpha}_s) \oplus F_{\text{RO1}}(\text{sid}||k^{\beta}_s) \oplus D^\alpha \oplus D^\beta) = h_{\alpha, \beta}^{s, s, \ell, \sigma} \quad (3)$$

If $F_{\text{RO2}}$ is queried on other values by the adversary $\text{Adv}_{\text{int}}$ then $\text{Sim}$ answers it with a random value. When $R^*$ gets corrupted, $\text{Sim}$ programs $F_{\text{RO1}}(\text{sid}||k^\ell_s)$ according to Eq. 1. Furthermore, he programs $F_{\text{RO2}}$ as follows:

$$F_{\text{RO2}}(\text{sid}||F_{\text{RO1}}(\text{sid}||k^{\alpha}_s) \oplus F_{\text{RO1}}(\text{sid}||k^{\beta}_s)) = h_{\alpha, \beta}^{s, s}$$

$$F_{\text{RO2}}(\text{sid}||F_{\text{RO1}}(\text{sid}||k^{\alpha}_s) \oplus F_{\text{RO1}}(\text{sid}||k^{\beta}_s) \oplus D^\alpha \oplus D^\beta) = h_{\alpha, \beta}^{s, s, \ell, \sigma} \quad (4)$$

$\text{Adv}_{\text{int}}$ corrupting $S^*$ does not obtain $k^\ell_s$ and $k^\beta_s$ except with negligible probability as we are in the $F_{\text{ROT}}$ hybrid model. It allows $\text{Sim}$ to correctly program $F_{\text{RO2}}$ involving $k^\ell_s$ and $k^\beta_s$, according to Eq. 4, permitting equivocation of $R$’s view and completing the simulation for the consistency checks.

**Optimized OT Extension:** Our 3 round OT extension protocol requires consistency check for each pair of columns, incurring an overhead of $\mathcal{O}(k^2)$ checks. It can be optimized to $\mathcal{O}(k)$ checks by following the 5 round protocol of [ALSZ15]. In the optimized protocol, the parties perform a 3 round coin-tossing subprotocol, after “OT Extension Phase I”, where the first round of the subprotocol can be run in parallel with the last round of “OT Extension Phase I”. The subprotocol returns a set of $\mathcal{O}(k)$ pairs of column indices. $R$ performs consistency check on those $\mathcal{O}(k)$ pairs of columns and sends the $h_{\alpha, \beta}^{a,b}$ to $S$ in parallel with the last round of the coin-tossing protocol. $S$ verifies the outcome of the coin-tossing protocol and the consistency checks. After successful
verification, $S$ executes “OT Extension Phase II”. The work of [ALSZ15] proved static security for this protocol. Adaptive security follows immediately if we perform the consistency checks using PRO ($\mathcal{F}_{RO1}$ and $\mathcal{F}_{RO2}$), similar to the 3 round protocol. $S$ will lack $k^\Sigma_{\alpha}$ and $k^\Sigma_{\beta}$ and Sim can equivocate $R$’s view according to Eq. 3. For empirical results, we implement the 5 round protocol and compare it with the state-of-the-art OT extension protocols.

7.3 Efficiency and Implications

Our protocols $\pi^p_{ALSZ}$ and $\pi^a_{ALSZ}$ requires one extra round, compared to the seed OTs, and it matches with the round complexity of the state-of-the-art static OT extension protocols [ALSZ13, KOS15, PSS17]. They also preserve the efficiency of the semi-honest and actively secure protocols of [ALSZ13] and [ALSZ15] respectively. In concrete terms, the amortized cost for generating $m$ copies of both semi-honest and active adaptively secure 1-out-of-2 OTs is $3\kappa$ bits communication and 3 symmetric key operations per OT. The other costs - seed OTs, PRG expansion and consistency checks, are independent of $m$. We present a few implications resulting from our OT Extension protocols:

- Our semi-honest OT Extension protocol requires static semi-honest OT protocol for the seed-OTs in a non-blackbox fashion, where we explicitly modified the simulation strategy for the static semi-honest OT. Thus, static semi-honest OT implies adaptive semi-honest OT Extension under the PROM assumption in a non-blackbox manner. Our theorem is summarized in Thm. 8. [LZ11] states that adaptive semi-honest oblivious transfer cannot be constructed from static semi-honest oblivious transfer in a black-box manner and we demonstrate that its possible in a non-blackbox way.

- If the random OT is instantiated using our receiver equivocal static OT protocol then our actively secure OT Extension protocols requires 3 and 5 rounds, thereby efficiently generating large number of OTs at an amortized cost of $3\kappa$ bits communication and 3 symmetric key operations per OT. However, the state-of-the-art adaptively secure 1-out-of-2 OT protocols [BCG17] that have a round complexity of 3 rounds, assume erasures, incur 27 exponentiations and communication of $10 \kappa$ bits. Thus, we drastically reduce the cost of generating adaptive OTs.

- We can efficiently generate $m \log N$ copies of 1-out-of-2 OTs using our OT Extension protocol (following the previous approach) and then apply our adaptive transformation (Sec. 6) to obtain $m$ copies of 1-out-of-$N$ OTs. The amortized cost for each 1-out-of-$N$ OT is $N + 3 \log N + 1$ symmetric key operations. Whereas the state-of-the-art adaptively secure 1-out-of-$N$ OT protocols [BCG17, BDD+17] require at least $O(N)$ exponentiations.

We have implemented our adaptively-secure variant of [ALSZ15] and presented the results in Sec. 9. For implementation we have considered the optimized variant where $O(\kappa)$ are performed.
Theorem 8. If there exists an Oblivious Transfer protocol that is secure in the presence of static semi-honest adversaries, then there exists an Oblivious Transfer Extension protocol from $\kappa$ to $\text{poly}(\kappa)$ that is secure in the presence of adaptive semi-honest adversaries, in a non-blackbox way, under the programmable random oracle assumption.

8 Non-Interactive UC-Secure Commitment Scheme

In this section we present our adaptively-secure NICOM scheme $\text{COM}$, implemented by protocol $\pi_{\text{COM}}$. The protocol $\pi_{\text{COM}}$ (described in Fig. 12) is universally composable and securely realizes the functionality $F_{\text{COM}}$ (described in Fig. 3) in the $\text{crs}$ model under ORO assumption and Discrete Log (over a group $G$) assumption. Later in this section, we demonstrate a protocol $\pi_{\text{CRS}}$ to generate an instance of the Discrete Log problem, as the $\text{crs}$, in 4 rounds. Once the $\text{crs}$ is generated, it can be used for subsequent instances of $\text{COM}$. Our $\text{crs}$ generation algorithm is independent of the parties’ inputs and hence it is adaptively-secure, rendering the whole protocol adaptively-secure under the ORO assumption. Our protocol is also secure in the Global Random Oracle model [CJS14] since we generate the local $\text{crs}$ within the protocol, using $\pi_{\text{CRS}}$.

8.1 Protocol Overview

We build upon the commitment scheme of Pedersen [Ped91], that relies on hardness of Discrete Log problem. The Pedersen commitment inherently supports equivocation as the message is statistically hidden in their case. However, for UC security the simulator, acting on behalf of $R$, has to extract the message, of a corrupted $S^*$, from the commitment. Our first approach was to apply an observable RO, i.e. $F_{\text{RO1}}: \{0,1\}^{\text{poly}(\kappa)} \rightarrow \mathbb{Z}_p$, on the message being committed, i.e. $F_{\text{RO1}}(\text{sid}||m)$, and then commit the response of the ORO query in the Pedersen commitment. This would allow the simulator to observe the queries and obtain candidate message values. However, the simulator cannot uniquely identify the message committed. It is necessary to extract the randomness, say $r_1$, used in the commit phase so that the simulator can match the (message, randomness) with the commitment value. We achieve this by enforcing $S$ to bind to $r_1$ using a second ORO, i.e. $F_{\text{RO2}} : \{0,1\}^\kappa \times \mathbb{Z}_p \rightarrow \{0,1\}^\kappa$. $S$ commits to $r_1$ by means of the query $F_{\text{RO2}}(\text{sid}||r_1)$. The hardness of the Discrete Log Problem ensures that a corrupted $S^*$ is unable construct more than one such (message, randomness) pair that matches the Pedersen commitment.

However, the above technique demands binding to $r_1$ using RO which in turn prevents equivocation by simulator, acting on behalf of $S$. In order to restore the equivocal property, $S$ is required to pad $F_{\text{RO2}}(\text{sid}||r_1)$ with fresh randomness $r_2$. This allows the simulator to equivocate the commitments by equivocating the first part of the commitment $c_1$ (Pedersen commitment on the message) separately, fixing $r_1$ to a new value. Now, the second part of the commitment $c_2$ can be equivocated by using $r_1$ and setting $r_2$ accordingly.
Our proof is in the Proof. and Adv Log Problem is hard in multiplicative group

If Theorem 9.

proving theorem 9.

active adversaries and securely realizes the functionality

We show that our non-interactive commitment scheme C

8.2 Static Security

We show that our non-interactive commitment scheme COM is secure against static active adversaries and securely realizes the functionality \( \mathcal{F}_{\text{COM}} \) in the UC model by proving theorem 9.

**Theorem 9.** If \( \mathcal{F}_{\text{ROI}} \) and \( \mathcal{F}_{\text{RO2}} \) are observable random oracles and solving the Discrete Log Problem is hard in multiplicative group \( G \), then \( \pi_{\text{COM}} \) UC-securely realizes the \( \mathcal{F}_{\text{COM}} \) functionality in the \( \text{crs} \) model against static active adversaries.

**Proof.** Our proof is in the \( \text{crs} \) model where it is assumed that Sim knows the trapdoor \( x \) s.t. \( h = g^x \) for the \( \text{crs} - (g; h) \). The proof proceeds in two cases - first, where Adv corrupts \( R^* \) and second, where the Adv corrupts \( S^* \).

\( R^* \) is corrupted: Adv corrupts \( R^* \) in the real world, Sim corrupts \( R^* \) in the ideal world and \( \text{Adv}_{\text{int}} \) corrupts \( R^* \) in the internal execution. During the commit phase, Sim commits to a random message \( m' \) using \( (r'_1, r'_2) \) as randomness, thereby computing \( \text{COM} \), similar to an honest sender and sends \( \text{COM}(m'; r'_1, r'_2) \) to \( \text{Adv}_{\text{int}} \). Sim further invokes \( \mathcal{F}_{\text{COM}} \) on behalf of \( R^* \) to obtain the message (RECEIPT, sid, S, R). In the decommit phase, Sim invokes \( \mathcal{F}_{\text{COM}} \) to obtain the message (DECOMMIT, sid, m). On obtaining the committed message \( m \), Sim provides randomness \( (r_1, r_2) \) s.t. \( \text{COM} \) decommits to \( m \). The randomness \( (r_1, r_2) \) is computed by Sim as follows:

- Let \( a = \mathcal{F}_{\text{ROI}}(\text{sid}||m) \) and \( a' = \mathcal{F}_{\text{ROI}}(\text{sid}||m') \).
- The trapdoor \( x \) is known to Sim. Sim generates \( (r_1, r_2) \) s.t the values of \( (c_1, c_2) \) remain unchanged while the commitment is being equivocated. Sim does so by
solving equations 5 and 6:

\[ g^a h'^r \mod p = g^{a'} h'^r \mod p \]
\[ \implies g^{a + r_1 x} \mod p = g^{a'} + r_1 x \mod p \]
\[ \implies a + r_1 x = a' + r_1 x \]
\[ \implies r_1 = (a' - a + r_1 x)^{-1} \quad (5) \]

\[ \mathcal{F}_{RO2}(\text{sid}|r_1) \oplus r_2 = \mathcal{F}_{RO2}(\text{sid}|r'_1) \oplus r'_2 \]
\[ \implies r_2 = \mathcal{F}_{RO2}(\text{sid}|r'_1) \oplus r'_2 + \mathcal{F}_{RO2}(\text{sid}|r_1) \quad (6) \]

Sim provides \((m, r_1, r_2)\), as opening to the commitment \(\text{COM}\), to \(\text{Adv}_{\text{int}}\).

At the end of the protocol, \(\text{Adv}_{\text{int}}\) sends its view to Sim. Sim forwards the view to \(Z\) who halts with an output.

**Indistinguishability**: We show that the real world view of \(Z\) is indistinguishable from the ideal world view by showing that the following two hybrids are statistically indistinguishable.

- **HYB\(_R\)**: Real world execution of the protocol.
- **HYB\(_I\)**: Same as HYB\(_R\), except that, Sim commits to a random message \(m'\) using \((r'_1, r'_2)\) as randomness and opens to \(m\) in the decommit phase using different randomness \((r_1, r_2)\). HYB\(_I\) corresponds to the ideal world view of \(Z\). It follows from Eq. 5 and 6 that the committed message remains statistically hidden in \(\text{COM}\). Hence, \(\forall m, m', r'_1, r'_2\), Sim can always find a consistent pair of randomness \((r_1, r_2)\) s.t the commitment opens to \(m\), provided Sim knows the trapdoor value \(x\). This proves statistical indistinguishability of the two worlds.

**\(S^*\) is corrupted**: \(\text{Adv}\) corrupts \(S^*\) in the real world. Sim corrupts \(S^*\) in the ideal world and \(\text{Adv}_{\text{int}}\) corrupts \(S^*\) in the internal execution. Sim emulates the role of an honest \(R\) against \(\text{Adv}_{\text{int}}\) in the internal execution. Sim plays the role of \(S^*\) in \(\mathcal{F}_{\text{COM}}\). During the commit phase, Sim obtains the commitment \(\text{COM}(m)\) from \(\text{Adv}_{\text{int}}\) in the internal execution. He observes the random oracle queries (both \(\mathcal{F}_{RO1}\) and \(\mathcal{F}_{RO2}\)) made by \(\text{Adv}_{\text{int}}\) and tries to extract the committed message \(m\). Sim aborts if it fails to extract the message. Let us assume that \(\text{Adv}_{\text{int}}\) makes \(s\) random oracle queries during the commit phase and Sim records them as \((q_1, q_2, \ldots, q_s)\). We denote a \((q_i, q_j)\) pair as valid, if \(q_i\) was queried to \(\mathcal{F}_{RO1}\), \(q_j\) was queried to \(\mathcal{F}_{RO2}\) and the following holds:

\[ g^{\mathcal{F}_{RO1}(\text{sid}|q_i) h^{q_j}} \mod p = c_1 \quad (7) \]

where \(\text{COM}(m) = (c_1, c_2)\) is received from \(\text{Adv}_{\text{int}}\) by Sim. Sim runs over all possible pairs of \((q_i, q_j)\) to find the valid pair(s). Based on the number of valid pair(s) discovered, Sim performs the following:

- If there does not exist any valid pair then Sim samples \(m' \leftarrow_R \mathbb{G}\) and sends \((\text{COMMIT}, \text{sid}, m')\) to \(\mathcal{F}_{\text{COM}}\).
– If there exists a unique valid \((q_i, q_j)\) pair then Sim sends \((\text{COMMIT}, \text{sid}, q_i)\) to \(F_{\text{COM}}\).
– If there exists more than one valid pair then Sim samples \(m' \leftarrow_R \mathcal{G}\) and sends \((\text{COMMIT}, \text{sid}, m')\) to \(F_{\text{COM}}\).

In the decommitment phase, \(\text{Adv}_{\text{Int}}\) sends \((m, r_1, r_2)\) to Sim. Sim verifies the commitment and aborts if verification fails in the internal execution. Whereas, Sim aborts in the ideal world as well as in \(F_{\text{COM}}\) if the following holds:

– **Case 1:** If there was no valid pair.
– **Case 2:** If there was one valid \((q_i, q_j)\) pair, and \(q_i \neq m\).
– **Case 3:** If there exists more than one valid pair.

If none of the above conditions hold, then Sim has an unique valid \((q_i, q_j)\) pair, s.t. \(q_i = m\) and \(q_j = r_1\). He sends \((\text{DECOMMIT}, \text{sid})\) to \(F_{\text{COM}}\) to complete simulation of \(F_{\text{COM}}\). At the end of the protocol, \(\text{Adv}_{\text{Int}}\) sends its view to Sim. Sim forwards the view to \(Z\) who halts with an output.

**Indistinguishability:** We show that the real world view of \(Z\) is indistinguishable from the ideal world view by showing that the following two hybrids are computationally indistinguishable.

– **HYB\(_R\)**: Real world execution of the protocol.
– **HYB\(_I\)**: It represents the ideal world execution of the protocol \(Z\) can distinguish between the two hybrids (or worlds) if Sim aborts in ideal world and \(F_{\text{COM}}\), while the honest R completes the protocol in real world. This occurs when the decommitment to COM provided by \(\text{Adv}_{\text{Int}}\) verifies correctly but Sim fails to extract the underlying committed message in the internal execution. This event has been captured as an union of three exhaustive cases presented in the simulation. We will show that each case occurs with negligible probability:

- **Case 1:** This case shows that \(\text{Adv}_{\text{Int}}\) obtained \(F_{\text{RO1}}(\text{sid}||m)\) and \(F_{\text{RO2}}(\text{sid}||r_1)\) without querying \(m\) or \(r_1\) to the random oracle during the commit phase. The random oracle assumption ensures that the query results would be random. Hence \(\text{Adv}_{\text{Int}}\) can guess \(F_{\text{RO1}}(\text{sid}||m)\) (or \(F_{\text{RO2}}(\text{sid}||r_1)\)) without querying \(m\) (or \(r_1\)) with negligible probability.
- **Case 2:** This case demonstrates that either \(\text{Adv}_{\text{Int}}\) obtained \(F_{\text{RO1}}(\text{sid}||m)\) and \(F_{\text{RO2}}(\text{sid}||r_1)\) without querying \(m\) or \(r_1\) to the random oracle during the commit phase, or the \(\text{Adv}_{\text{Int}}\) possesses two valid pairs \((q_i, q_j)\) and \((m, r)\). The first event occurs with negligible probability as we are in the RO model. And the occurrence of the second event implies that the DLP problem can be solved by using \(\text{Adv}_{\text{Int}}\) as a blackbox. The adversary \(\text{Adv}_{\text{DLP}}\) will obtain a challenge instance \((g, h) = (g, g^x)\) from the challenger \(\text{Chall}_{\text{DLP}}\) of the DLP game. \(\text{Adv}_{\text{DLP}}\) will invoke the \(\text{Adv}_{\text{Int}}\) while simulating the role of an honest R in the commitment scheme with \(\text{crs} = (g, h)\). \(\text{Adv}_{\text{DLP}}\) will observe the queries made by \(\text{Adv}_{\text{Int}}\) to obtain a valid pair \((q_i, q_j)\). He will receive the commitment \(\text{COM}\)
We briefly discuss the proof in this section.

**Case 3:** $g$ valid pair in Eq 7) as $(it is not possible for both $F_{\text{RO1}}(\text{sid}||q_i) + q_j x = F_{\text{RO1}}(\text{sid}||m) + r x$)

$$g^{F_{\text{RO1}}(\text{sid}||q_i)} h^{q_j} = g^{F_{\text{RO1}}(\text{sid}||m)} h^{r} \mod p$$

$$\Rightarrow g^{F_{\text{RO1}}(\text{sid}||q_i) + q_j x} = g^{F_{\text{RO1}}(\text{sid}||m) + r x} \mod p$$

$$\Rightarrow F_{\text{RO1}}(\text{sid}||q_i) + q_j x = F_{\text{RO1}}(\text{sid}||m) + r x$$

$$\Rightarrow x = (F_{\text{RO1}}(\text{sid}||q_i) - F_{\text{RO1}}(\text{sid}||m))(r - q_j)^{-1} \quad (8)$$

However, we assume that the DLP problem is hard in group $G$ and hence this case occurs with negligible probability.

- **Case 3:** This case indicates that $\text{Adv}_{\text{int}}$ obtains two or more valid pairs. This again implies that either the DLP problem has been solved by $\text{Adv}_{\text{int}}$ or $\text{Adv}_{\text{int}}$ found a collision in the random oracle queries. Let us denote two such valid pairs as $(q_i, q_j)$ and $(q_i', q_j')$. We will further split this case into two more sub-cases for analysis based on the equality of $F_{\text{RO1}}(\text{sid}||q_i)$ and $F_{\text{RO1}}(\text{sid}||q_j)$ values.

  i. $q_i \neq q_i', F_{\text{RO1}}(\text{sid}||q_i) = F_{\text{RO1}}(\text{sid}||q_j'), q_j = q_j'$; This indicates that $\text{Adv}_{\text{int}}$ found a collision in the random oracle queries as $q_i \neq q_i'$. However, this event occurs with negligible probability as we are in the random oracle model.

  ii. $q_i \neq q_i', F_{\text{RO1}}(\text{sid}||q_i) \neq F_{\text{RO1}}(\text{sid}||q_j'), q_j \neq q_j'$; In this case, $\text{Adv}_{\text{int}}$ can be used as blackbox by $\text{Adv}_{\text{DLP}}$ to solve the DLP problem. $\text{Adv}_{\text{DLP}}$ will invoke the $\text{Adv}_{\text{int}}$ while simulating the role of an honest $R$ in the commitment scheme with $\text{crs} = (g, h)$. $\text{Adv}_{\text{DLP}}$ will observe the queries made by $\text{Adv}_{\text{int}}$ to obtain two valid pairs $(q_i, q_j)$ and $(q_i', q_j')$ s.t. $q_i \neq q_i'$ and $q_j \neq q_j'$. $\text{Adv}_{\text{DLP}}$ solves the DLP problem by finding $x$ as follows:

$$g^{F_{\text{RO1}}(\text{sid}||q_i)} h^{q_j} = g^{F_{\text{RO1}}(\text{sid}||q_j')} h^{q_j'} \mod p$$

$$\Rightarrow g^{F_{\text{RO1}}(\text{sid}||q_i) + q_j x} = g^{F_{\text{RO1}}(\text{sid}||q_j') + q_j' x} \mod p$$

$$\Rightarrow F_{\text{RO1}}(\text{sid}||q_i) + q_j x = F_{\text{RO1}}(\text{sid}||q_j') + q_j' x$$

$$\Rightarrow x = (F_{\text{RO1}}(\text{sid}||q_i) - F_{\text{RO1}}(\text{sid}||q_j'))(q_j' - q_j)^{-1} \quad (9)$$

However, we assume that the DLP problem is hard in group $G$ and hence this case occurs with negligible probability.

The other two cases involve $F_{\text{RO1}}(\text{sid}||q_i) = F_{\text{RO1}}(\text{sid}||q_i'), q_j \neq q_j'$ and $F_{\text{RO1}}(\text{sid}||q_i) \neq F_{\text{RO1}}(\text{sid}||q_j'), q_j = q_j'$ for $q_i \neq q_i'$. However, in these cases, it is not possible for both $(q_i, q_j), (q_i', q_j')$ to be valid pairs (refer condition for valid pair in Eq 7) as $g^{F_{\text{RO1}}(\text{sid}||q_i)} h^{q_j} \neq g^{F_{\text{RO1}}(\text{sid}||q_i')} h^{q_j'}$. There maybe almost one valid pair, for which the analysis follows from Case 1 and 2.

### 8.3 Adaptive Security

Interestingly, our commitment scheme $\pi_{\text{COM}}$ (Fig 12) also satisfies the stronger security notion of adaptivity under the observable random oracle assumption in the crs setup.

We briefly discuss the proof in this section.
Theorem 10. If $F_{RO1}$ and $F_{RO2}$ are observable random oracles and solving the Discrete Log Problem is hard in multiplicative group $G$, then protocol $\pi_{COM}$ UC-securely realizes the $F_{COM}$ functionality in the crs model against adaptive active adversaries (without erasures).

Proof. To prove adaptive security, we require $\text{Sim}$ to equivocate the views of $R$ and $S$ appropriately on adaptive corruption in addition to static security. We divide our simulation into cases based on the party being corrupted:

$R^*$ is corrupted: $R$ does not have any private input or input randomness, and so the role of $R$ in the protocol is restricted to verifying the commitments upon obtaining the message ($m$) and randomness ($r_1$ and $r_2$) during the decommitment phase. When $\text{Adv}_{\text{Int}}$ corrupts $R^*$ at any stage, i.e. commit phase, decommitment phase or post execution of the protocol, $\text{Sim}$ returns a random tape as the internal randomness of $R^*$.

$S^*$ is corrupted: $\text{Sim}$ closely imitates the role of the simulator for static corruption when $S^*$ gets corrupted adaptively. If $\text{Adv}_{\text{Int}}$ corrupts $S^*$ in the beginning of the protocol or before the commitment is sent, $\text{Sim}$ returns a random tape as the internal randomness of $S^*$. If $\text{Adv}_{\text{Int}}$ corrupts $S^*$ after the commitment is sent, i.e. in decommitment phase or post execution, then $\text{Sim}$ needs to equivocate. $\text{Sim}$ initially commits to a dummy message $m'$ and upon corruption of $S^*$, $\text{Sim}$ obtains message $m$ and successfully equivocates the commitment to open to $m$ using randomness $(r_1, r_2)$ (computed as described in the proof of Theorem 9). $Z$ cannot distinguish between the commitment to $m'$ and commitment to the actual value due to the statistical hiding property of the scheme. Equivocation follows from the equivocal property as proven for the static case.

8.4 Generation of crs using Observable Random Oracle

For our protocol $\pi_{COM}$, the involved parties require a crs of the form $(g, h)$, where $h = g^x$. Also, the simulator should have the knowledge of the trapdoor $x$ in order to perform simulation. The crs can be trivially generated if we assume a PRO. The parties can generate the crs as $F_{RO}(\text{sid}||\text{"com"})$. $\text{Sim}$ samples $x$ and programs the RO to return $(g, h)$ s.t. $h = g^x$. This preserves adaptive security of $\pi_{COM}$ when the crs generation algorithm is included as part of $\pi_{COM}$. However we are interested in generating the crs without relying on the programmability of the RO. This can be achieved by executing a 2PC protocol $\pi_{CRS}$ (Fig. 13) relying solely on the observability property of the RO. Once the crs is generated it can be reused for subsequent commitments between the parties. By generating the crs as part of the protocol it can be concluded that our commitment scheme is adaptively secure in the Global Random Oracle model.

Intuition: Our $\pi_{CRS}$ protocol proceeds in two phases - coin tossing and zero knowledge proof of knowledge (ZKPoK). The parties perform coin tossing to generate random shares $h_S$ and $h_R$. These shares are then used to obtain $h = h_S.h_R$. Once the coin tossing is performed, they engage in ZKPoK in order to prove the knowledge of trapdoors to their respective shares, i.e. $h_S$ and $h_R$ respectively. The ZKPoK enables the simulator of $\pi_{CRS}$ to extract the corrupted party’s share in order to obtain the trapdoor
Fig. 13: Generating crs = (g, h) for πCOM using \( \mathcal{F}_{\text{RO}} \)

\[ \pi_{\text{crs}} \]

- **Public Inputs:** The generator of group \( G \) is \( g \).
- **Functionality:** Random oracle \( \mathcal{F}_{\text{RO}} : \{0, 1\}^{\text{poly}(\kappa)} \rightarrow \{0, 1\}^{\kappa} \).
- **Private Inputs:** The parties do not have any input.

**Coin Tossing:**
- Round 1:
  - \( S \) samples \( x_S \leftarrow R \mathbb{Z}_p \), computes \( h_S = g^{x_S} \mod p \) and sends \( \mathcal{F}_{\text{RO}}(\text{sid}||h_S) \) to \( R \).
  - \( R \) samples \( x_R \leftarrow R \mathbb{Z}_p \), computes \( h_R = g^{x_R} \mod p \) and sends \( \mathcal{F}_{\text{RO}}(\text{sid}||h_R) \) to \( S \).
- Round 2:
  - \( S \) sends \( h_S \) to \( R \).
  - \( R \) sends \( h_R \) to \( S \).
- Computation:
  - \( S \) verifies \( \mathcal{F}_{\text{RO}}(\text{sid}||h_R) \) and computes \( h = h_S \cdot h_R \mod p \), else aborts.
  - \( R \) verifies \( \mathcal{F}_{\text{RO}}(\text{sid}||h_S) \) and computes \( h = h_S \cdot h_R \mod p \), else aborts.

**Zero Knowledge Proof of Knowledge:**

\( S \) and \( R \) perform the following steps in parallel with their roles interchanged.
- Round 1:
  - \( R \) samples a challenge string \( c \leftarrow R \{0, 1\}^{\nu} \) and sends \( \mathcal{F}_{\text{RO}}(\text{sid}||c) \) to \( S \).
- Round 2:
  - \( S \) computes \( \nu \) garbled circuits, by sampling \( \text{seed}_i \leftarrow R \{0, 1\}^{\nu} \) and \((\text{GC}_i, e_i, d_i) \leftarrow \text{Gb}(\text{PRG(} \text{seed}_i \text{), } C)\) where \( C \) computes \( g^{x_i} \) for \( i \in [\nu] \).
  - \( S \) sends \( \mathcal{F}_{\text{RO}}(\text{sid}||\text{seed}_i), \mathcal{F}_{\text{RO}}(\text{sid}||\text{GC}_i) \) and \( d_i \) to \( R \).
- Round 3:
  - \( R \) reveals \( c \) to \( S \). \( S \) verifies \( \mathcal{F}_{\text{RO}}(\text{sid}||c) \) and aborts if verification fails.
- Round 4: (Let \( c_i \) denote \( i^{\text{th}} \) bit of \( c \))
  - If \( c_i = 0 \), then \( \text{GC}_i \) is a check circuit and \( S \) sends \( \text{seed}_i \) to \( R \).
  - If \( c_i = 1 \), then \( \text{GC}_i \) is an evaluation circuit and \( S \) sends \( X = \text{En}(x_S, e_i) \) to \( R \).
- Computation:
  - \( R \) verifies the check circuit \( \text{GC}_i \) as \( \text{Ve}(\mathcal{C}, \text{GC}_i, e_i) \), if \( c_i = 0 \), else he aborts.
  - \( R \) computes \( y = \text{De}(\text{Ev}(\text{GC}_i, X)) \), if \( c_i = 1 \), and aborts if \( y \neq h_S \).
  - \( R \) stores \((g, h)\) as the crs.

to \((g, h)\). Our coin tossing protocol requires 2 rounds and ZKPoK consumes 4 rounds. However, the first 2 rounds of the ZKPoK can be parallelized with the coin tossing protocol, thus yielding a 4 round protocol for crs generation. The coin tossing is performed using the random oracle and the ZKPoK is performed by plugging a simplified ZK version of [HV16] which uses garbled circuits and random oracle only. Note that we need a ZKPoK protocol which can be implemented using an ORO and without relying on any other setup assumption. This rules out the possibilities of using efficient ZKPoK protocol of [JKO13] since it uses an OT, which would further require other setup assumptions. However, the interactive ZKPoK variants of “MPC-in-the-head” protocols [GMO16, AHIV17] also suffices for our purpose.
Static Security: We prove that $\pi_{\text{CRS}}$ generates $\text{crs}$ in the presence of static active adversaries. Since our protocol is symmetric, we consider the case where $S^*$ is corrupted and simulator $\text{Sim}$ plays the role of honest $R$. $\text{Sim}$ behaves like an honest $R$ throughout the protocol and extracts the trapdoor $x_S$ from the ZKPoK sent by $S^*$. $\text{Sim}$ plays the role of an evaluator in the ZKPoK whereas $S^*$ plays the role of a constructor. $\text{Sim}$ extracts the seeds, for all circuits sent by $S^*$, by observing the queries made to $\mathcal{F}_{\text{RO}}$ and matching them with $\mathcal{F}_{\text{RO}}(\text{sid}||\text{seed}_i)$. $\text{Sim}$ generates the encoding information of all evaluation circuits from the seeds. Finally, $\text{Sim}$ extracts the input witness $x_S$ of $S^*$, by matching the input wire labels, of the evaluation circuit, with the encoding information, of the evaluation circuit. On the other hand, $\text{Sim}$ can simulate the ZKPoK, on behalf of constructor, by behaving like an honest $R$ since he knows the trapdoor $x_R$ sampled by him. A corrupt $S^*$ cannot extract any information about $x_R$ due to the privacy of the garbling scheme.

Adaptive Security: We prove that protocol $\pi_{\text{CRS}}$ is adaptively-secure by observing that the parties do not have any private inputs at the outset of the protocol. The simulator can simulate by running honest sender/receiver algorithms as required. Upon corruption of a party, the simulator can reveal its random tape for the corresponding party. The simulated honest party’s view will be indistinguishable from the real honest party’s view since in both cases the protocol transcripts are honestly generated using a random tape. This proves that our protocol $\pi_{\text{CRS}}$ UC-securely generates the $\text{crs}$ required for our commitment scheme.

8.5 Final Commitment Scheme $\pi = \pi_{\text{CRS}} + \pi_{\text{COM}}$

We can combine $\pi_{\text{CRS}}$ and $\pi_{\text{COM}}$ to obtain an UC-secure protocol $\pi$ which implements $\mathcal{F}_{\text{COM}}$ functionality in the presence of adaptive adversaries solely relying on ORO. The security has been summarized in Theorem 11. It can be proven secure in the GRO of [CJS14] since it does not require a local $\text{crs}$.

Theorem 11. If Garble = (Gb, En, Ev, De, Ve) is a private, verifiable garbling scheme and solving the Discrete Log Problem is hard in multiplicative group $\mathbb{G}$, if $\pi_{\text{COM}}$ implements $\mathcal{F}_{\text{COM}}$ functionality against active adaptive adversaries in the $\text{crs}$, ORO-model and $\pi_{\text{CRS}}$ generates the $\text{crs}$ against active adaptive adversaries in the ORO model, then $\pi = \pi_{\text{CRS}} + \pi_{\text{COM}}$ implements $\mathcal{F}_{\text{COM}}$ functionality against active adaptive adversaries (without erasures) in the ORO-model.

8.6 Efficiency of $\pi$

We analyze the efficiency of $\pi$ by analyzing the efficiency of $\pi_{\text{CRS}}$ and $\pi_{\text{COM}}$ separately. The $\pi_{\text{CRS}}$ protocol requires 4 rounds and has a computation cost of 2 exponentiations, $8\nu + 8$ oracle queries, construction and evaluation of $2\nu$ circuits. The communication cost is $2\nu$ circuits + $(8 + 4\nu + 2\kappa)$ strings of $\kappa$ bits. However, it is a one-time cost which would get amortized when multiple commitments are performed using the same $\text{crs}$. Next, we analyze the efficiency of $\pi_{\text{COM}}$. The length of our commitment is one group element and one $\kappa$ bit string, independent of the message length. Decommitment incurs
communication of two group elements. The computation is also minimal, incurring one random oracle query on $|m|$ bits, one oracle query on a $\kappa$ bits string and two exponentiations on sender’s side for committing. Decommitment incurs similar computation overhead on the receiver’s end. It is non-interactive in both commitment and decommitment phase. This yields an efficient adaptively secure commitment scheme which is practically motivated for offline-online 2PC/MPC protocols [HKK+14, LR14, LR15, RR16]. The $\pi_{\text{CRS}}$ protocol can be run in the offline phase while the commitment scheme can be conveniently used in the online phase.

9 Implementation Details

In this section, we support our theoretical claims with empirical details. First, we talk about the hardware configuration of our system and then we compare our results with other relevant protocols in the literature. We specifically compare against the state-of-the-art statically secure protocols to establish the fact that we achieve upgrade in security with minimal overhead.

Hardware Details: We have tested the code in a standard LAN network and a simulated WAN setting. Our machine has 8GB RAM and an Intel Core i5-4690 CPU with 3.5 GHz processor speed. For WAN simulation, we used the tc tool of Linux and introduced a round trip delay of 100 milliseconds into the network, with a bandwidth of 20Mbps.

Software Details: To establish OT results, first we compare the performance of our adaptively secure OT with the statically secure PVW OT, both under DDH assumption. Table 5 shows the results for LAN and WAN setting for a message size of 128 bits. Second, for OT Extension, we build upon the OT extension code made available by Encrypto group on github [OTC]. It contains the OT extension implementations of KK13, NNOB and ALSZ in C++, using the Miracl library for elliptic curve arithmetic. Additionally, we have received the code from the authors of [PSS17] and used it in our comparison for 1-out-of-N OT extension. The random oracle in our protocol has been implemented by the use of standard AES-128 and SHA-256 hash function. We require 190, 342, 256 and 190 seed OTs for ALSZ, NNOB, PSS and our protocol respectively to obtain the necessary security. Tables 6 and 7 give a detailed view of cost involved in attaining 1-out-of-2 OT Extension and 1-out-of-N (1-out-of-16) OT Extension respectively in both LAN and WAN setting over various values of $m$ which is the number of OTs generated. For our adaptively secure commitment scheme, we use SHA-256 as a standard replacement to Observable Random Oracle and tabulate efficiency values over various message lengths as depicted in 4. For all implementation purposes, we set the computational security parameter to be $\kappa = 128$ and the statistical security parameter to be $\nu = 40$.

Comparison: Tables 5, 6 and 7 compares the performance of our adaptively secure protocols against the existing state-of-the-art static protocols. Our adaptively secure OT protocol has a mere overhead of 0.02% and 1.0% over the PVW protocol in the LAN and WAN setting respectively. This overhead is nominal compared to the security
Table 5: Table indicating implementation results for Oblivious Transfer protocol in the LAN setting for $\kappa=128$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Msg Size (bits)</th>
<th>Setting</th>
<th>Runtime of S (s)</th>
<th>Runtime of R (s)</th>
<th>Comm. (bits) of S (s)</th>
<th>Comm. (bits) of R (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVW</td>
<td>128</td>
<td>LAN</td>
<td>1.675</td>
<td>0.970</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>(Static, DDH based)</td>
<td></td>
<td>WAN</td>
<td>4.321</td>
<td>1.681</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our Scheme</td>
<td>128</td>
<td>LAN</td>
<td>1.676</td>
<td>0.971</td>
<td>512</td>
<td>384</td>
</tr>
<tr>
<td>(Adaptive, DDH-based)</td>
<td></td>
<td>WAN</td>
<td>4.329</td>
<td>1.690</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

achieved. We can in fact consider this to be an upgrade in security and ideally suited for real world applications, at the cost of static OT. From Fig. 14 we can see that our protocol has minimal overhead over the static 1-out-of-2 OT extension protocols. We incur an overhead of 2% and 11.95% over ALSZ, 8% and almost no overhead over NNOB in the LAN and WAN setting respectively in terms of computation. Communication wise, we incur 2.4% overhead over ALSZ and none over NNOB. For generating 1-out-of-16 OT, we incur an overhead of 0.6% and 0.6% over ALSZ, 2% and 7% NNOB, 16% and 24% over PSS in LAN and WAN runtime. In concrete terms, each extended 1-out-of-2 adaptive OT costs around $0.104-2 \mu s$ based on the number of OTs generated together.

The implementation of KOS [KOS15] is not available in the public domain. ALSZ has 24% overhead over [IKNP03] and KOS has 5% over [IKNP03] in LAN setting. This translates to 18% overhead of ALSZ over KOS. Based on that we can compare accordingly with KOS, by comparing with ALSZ. Similarly each extended 1-out-of-16 adaptive OT costs around $21 \mu s$. Hence, our OT extension protocols generates practically efficient OTs which can find useful applications in real-life protocols that demand adaptive security. We compare our commitment scheme with the state-of-the-art adaptive protocol of [HM04]. Our scheme requires around 0.20 ms for committing to a string of length 2048 bits, whereas they require around 0.48 ms. This implies our scheme is approx. 2 times faster compared to [HM04]. Tab. 4 compares our protocol with [HM04] for various message sizes and demonstrates our efficiency gain.

Table 6: Performance Comparison of our adaptive OT extension with the state-of-the-art static OT extensions for generating 1-out-of-2 OTs. The value of $m$ denotes the number of OTs generated and $\kappa = 128$.

<table>
<thead>
<tr>
<th>$m$ (10^5)</th>
<th>ALSZ</th>
<th>NNOB</th>
<th>Ours</th>
<th>ALSZ</th>
<th>NNOB</th>
<th>Ours</th>
<th>ALSZ</th>
<th>NNOB</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25</td>
<td>0.234</td>
<td>0.127</td>
<td>0.253</td>
<td>14.35</td>
<td>14.75</td>
<td>14.469</td>
<td>11.00</td>
<td>13.36</td>
<td>11.024</td>
</tr>
<tr>
<td>2.5</td>
<td>0.350</td>
<td>0.270</td>
<td>0.369</td>
<td>26.44</td>
<td>27.11</td>
<td>26.569</td>
<td>21.96</td>
<td>26.70</td>
<td>21.984</td>
</tr>
<tr>
<td>5</td>
<td>0.590</td>
<td>0.562</td>
<td>0.610</td>
<td>51.02</td>
<td>52.43</td>
<td>51.14</td>
<td>43.90</td>
<td>53.38</td>
<td>43.924</td>
</tr>
<tr>
<td>12.5</td>
<td>1.237</td>
<td>1.329</td>
<td>1.259</td>
<td>123.9</td>
<td>127.43</td>
<td>124.024</td>
<td>109.74</td>
<td>133.45</td>
<td>109.764</td>
</tr>
</tbody>
</table>
Table 7: Performance Comparison of our adaptive OT extension with the state-of-the-art static OT extensions for generating 1-out-of-16 OTs. The value of $m$ denotes the number of OTs generated and the value of the security parameter is $\kappa = 128$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>ALSZ</th>
<th>NNOB</th>
<th>PSS</th>
<th>Ours</th>
<th>ALSZ</th>
<th>NNOB</th>
<th>PSS</th>
<th>Ours</th>
<th>ALSZ</th>
<th>NNOB</th>
<th>PSS</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1.25 \times 10^5$</td>
<td>02.65</td>
<td>02.61</td>
<td>2.30</td>
<td>2.668</td>
<td>18.10</td>
<td>16.90</td>
<td>14.62</td>
<td>18.218</td>
<td>16.67</td>
<td>21.60</td>
<td>4.77</td>
<td>16.693</td>
</tr>
<tr>
<td>$2.5 \times 10^5$</td>
<td>05.26</td>
<td>05.05</td>
<td>4.50</td>
<td>5.287</td>
<td>33.80</td>
<td>31.40</td>
<td>26.45</td>
<td>34.019</td>
<td>33.33</td>
<td>43.21</td>
<td>9.54</td>
<td>33.353</td>
</tr>
<tr>
<td>$5 \times 10^5$</td>
<td>10.04</td>
<td>10.10</td>
<td>9.0</td>
<td>10.062</td>
<td>65.00</td>
<td>60.40</td>
<td>50.75</td>
<td>65.119</td>
<td>66.62</td>
<td>86.39</td>
<td>19.08</td>
<td>66.643</td>
</tr>
<tr>
<td>$1.25 \times 10^6$</td>
<td>24.81</td>
<td>24.84</td>
<td>22.50</td>
<td>24.836</td>
<td>158.60</td>
<td>143.20</td>
<td>121.94</td>
<td>159.76</td>
<td>166.54</td>
<td>215.95</td>
<td>47.70</td>
<td>166.563</td>
</tr>
</tbody>
</table>

Fig. 14: Performance Comparison of OT extensions for producing 1-out-of-2 OTs on 16 bits. The x-axis denotes number of OTs (order $10^5$) and y-axis denotes runtime for graphs (a); (b) and communication (MB) for graph (c).

10 Conclusion

In this paper, we presented UC-secure, efficient round-optimal protocols for cryptographic primitives, secure against adaptive adversaries aided with asymptotic and empirical costs. Our OT and OT extension schemes rely solely on programmability of random oracle as a setup assumption, while our commitment scheme relies on the observability of random oracle. Our OT and commitment scheme are secure in the GRO model, making it the first adaptive OT and commitment scheme in the strong GUC model and thereby realistic for applications. We have validated our claims with robust implementation results supported by tables; graphs depicting the cost incurred for computation and communication in various settings. These results highlight the prospective practicality of our protocols in the adaptive setting. Finally, we leave it as an open problem to obtain a fully simulatable adaptive OT in the GRO model and an OT extension based on assumptions weaker than random oracle.
References


A Universal Composability Model

We prove security of our protocol in the standard Universal Composability (UC) framework of Canetti [Can01], with static and adaptive corruptions. And we conclude this
section with the definition of $F$-hybrid model, which is instrumental for security proofs in the UC model.

A.1 Static Security in the UC Model

In this model, the real world execution of protocol $\pi$ is carried out between the honest parties $P_1$ and $P_2$ and an adversary $\text{Adv}$, in the presence of an external entity called the environment $Z$. All the parties are PPT Turing machines and $Z$ has an auxiliary information $z$. At the outset of the protocol the environment initiates the parties with inputs and provides some initial information to $\text{Adv}$. $Z$ is allowed to interact with $\text{Adv}$ throughout the protocol. At the outset of the protocol, $\text{Adv}$ may or may not corrupt a party. Upon corruption of a party, $\text{Adv}$ gets access to the internal state and input of that party. From now on the party will behave according to $\text{Adv}$'s instructions (since we are in the malicious model). At the end of the protocol, the honest parties send their output to $Z$ while $\text{Adv}$ outputs $\bot$ on behalf of the corrupted parties and its internal state to $Z$. We denote the view of $Z$ as $\text{REAL}_{F, \text{Adv}, Z}(1^\kappa, z)$.

In the ideal world we consider the honest parties $P_1$ and $P_2$, a PPT adversary $\text{Sim}$, $Z$ and the functionality $F$. $\text{Sim}$ has a random tape $r$ and security parameter $\kappa$. He simulates the role of $\text{Adv}$ in the ideal world and whenever $\text{Adv}$ corrupts a party in the real world $\text{Sim}$ corrupts that party in the ideal world and gets access to its internal state. $\text{Sim}$ invokes the algorithm of $\text{Adv}$, in his head, in another internal protocol execution where $\text{Sim}$ simulates the view of the honest parties to $\text{Adv}$. We will denote this internal copy of $\text{Adv}$ as $\text{Adv}_{\text{int}}$. Based on the reply of $\text{Adv}_{\text{int}}$ in the internal execution, $\text{Sim}$ behaves accordingly in the ideal world execution. He extracts the inputs of the corrupted parties in the internal execution and invokes $F$ in the ideal world with those inputs to obtain the output. In the internal execution he simulates the protocol in such a way that $\text{Adv}_{\text{int}}$ obtains that output. At the end of the protocol, $\text{Adv}_{\text{int}}$ forwards his view to $\text{Sim}$ who forwards it to $Z$. We denote the view of $Z$ as $\text{IDEAL}_{F, \text{Sim}, Z}(1^\kappa, z)$. We say that a protocol $\pi$ UC-securely implements a functionality $F$ in the presence of static adversaries if the real world and ideal world views are indistinguishable.

**Definition 1.** Let $\pi$ be a protocol for computing a functionality $F$. We say that $\pi$ UC-securely computes the two party protocol functionality $F$ in the presence of static adversaries if for every PPT adaptive real-world adversary $\text{Adv}$ and every environment $Z$, there exists a PPT ideal-world adversary $\text{Sim}$, such that:

$$\text{REAL}_{F, \text{Adv}, Z}(1^\kappa, z) \approx \text{IDEAL}_{F, \text{Sim}, Z}(1^\kappa, z)$$

A.2 Adaptive Security in the UC Model

In the adaptive setting, $Z$ can ask the real world adversary $\text{Adv}$ to corrupt an honest party during the real world execution of the protocol or after the execution completes. During the execution, $\text{Adv}$ can observe the public transcript of the protocol and based on that he can adaptively corrupt an honest party. Once a party gets corrupted, $\text{Adv}$ gets access to the input and private randomness of the party, thus controlling the party from thereon. In case of post execution corruption, $\text{Adv}$ observes the output and the
transcript of the protocol, and then he corrupts the honest party to get access to the input and private randomness of the party. After post execution corruption occurs, Adv forwards its view to Z. Based on that, Z constructs its real world view, which we denote as $\text{REAL}_{F,\text{Adv},Z}(\kappa, z)$.

Similarly, in the ideal world Z can ask the ideal world adversary Sim to corrupt an honest party during the ideal world execution of the protocol or after the execution completes. When Z instructs Sim to corrupt an honest party in the ideal world, Sim obtains the input of the honest party, in the ideal world, and he instructs the internal world adversary AdvInt to corrupt the corresponding honest party in the internal world. Recall that Sim simulates the honest parties in the internal execution. When AdvInt corrupts an honest party in the internal world, Sim has to produce a private randomness for the simulated honest party such that it matches with the input of the honest party and the simulated transcript produced by Sim, in the internal world, on behalf of the honest party. Sim provides this matching randomness and the input of the simulated honest party to AdvInt in the internal world. In case of post execution corruption of an honest party, Sim obtains the honest party’s input in the ideal world and produces the matching randomness (corresponding to the simulated transcript) in a similar fashion to AdvInt in the internal world. After post execution corruption occurs, Adv forwards its view to Sim, who forwards it to Z. Based on that, Z constructs its ideal world view, which we denote as $\text{IDEAL}_{F,\text{Sim},Z}(\kappa, z)$. We say that a protocol $\pi$ UC-securely implements a functionality $F$ in the presence of adaptive adversaries if the real world and ideal world views are indistinguishable.

**Definition 2.** Let $\pi$ be a protocol for computing a functionality $F$. We say that $\pi$ UC-securely computes the two party protocol functionality $F$ in the presence of adaptive adversaries if for every PPT adaptive real-world adversary Adv and every environment $Z$, there exists a PPT ideal-world adversary Sim, such that:

$$\text{REAL}_{F,\text{Adv},Z}(\kappa, z) \approx \text{IDEAL}_{F,\text{Sim},Z}(\kappa, z)$$

**A.3 The $F$-hybrid model.**

In order to construct our protocols, we utilize other secure two-party protocols as subprotocols. The standard way of doing this is to work in a “hybrid model” where both the parties interact with each other (as in the real model) in the outer protocol and use ideal functionality calls (as in the ideal world) for the subprotocols. The UC composition theorem states that if a protocol $\rho$ UC-securely implements a functionality $F$, then any execution of $\rho$ in a bigger protocol can be replaced with ideal calls to the functionality $F$. Specifically, while constructing a protocol $\pi$ that uses $\rho$ as subprotocol, for securely computing some functionality $F$, the parties can run $\pi$ and invoke $F$. The execution of $\pi$ that invokes $F$, for each execution of $\rho$, is called the $F$-hybrid execution of $\pi$ and is denoted as $\pi^F$. The hybrid ensemble $\text{HYB}_{\pi^F,\text{Adv},Z}(\kappa, z)$ describes Z’s output after interacting with Adv and the parties running protocol $\pi^F$. Whereas, the execution of $\pi$ that considers execution of $\rho$ is denoted as $\pi^\rho$. The hybrid ensemble $\text{HYB}_{\pi^\rho,\text{Adv},Z}(\kappa, z)$ describes Z’s output after interacting with Adv and the parties running protocol $\pi^\rho$. By UC security, the two hybrids $\text{HYB}_{\pi^F,\text{Adv},Z}(\kappa, z)$ and
HYB_{π^*, AOV, Z}(1^n, z) are indistinguishable. This permits replacing executions of ρ, in π, with ideal calls to F functionality; thereby allowing π to execute in the F-hybrid model. It simplifies the security proof of πF as it can be performed in the F-hybrid model, instead of proving security of ρ within the proof of πρ.

B Garbled Circuits

Bellare et al [BHR12] gave an abstraction of garbling schemes for circuits and formalized several notions of security. Using the language of [BHR12] for circuits; the circuit itself is a directed acyclic graph, where each gate g is indexed by its outgoing wire, and its left and right incoming wires A(g) and B(g) are numbered such that g > B(g) > A(g). Also, a circuit output wire can not be an input wire to any gate. We denote the number of input wires, gates and output wires using n, q and m respectively in a circuit C.

At a high-level, a garbling scheme consists of the following algorithms: Gb takes a circuit as input and outputs a garbled circuit, encoding information, and decoding information. En takes an input x and encoding information and outputs a garbled input X. Ev takes a garbled circuit and garbled input X and outputs a garbled output Y. Finally, De takes a garbled output Y and decoding information and outputs a plain circuit-output (or an error, ⊥).

In [JKO13], there is an additional verification algorithm in the garbling scheme which when accepts a given (GC, e) signifies that the GC is correct, and that the garbled output corresponding to any clear output can be extracted. Formally, a garbling scheme is defined by a tuple of functions Garble = (Gb, En, Ev, De, Ve), described as follows:

- Garble algorithm Gb (1^n, C): A randomized algorithm which takes as input the security parameter and a circuit C: \{0,1\}^n \rightarrow \{0,1\}^m and outputs a tuple of strings (GC, e, d), where GC is the garbled circuit, e denotes the input-wire labels, and d denotes the decoding information.
- Encode algorithm En (x, e): a deterministic algorithm that outputs the garbled input X corresponding to input x.
- Evaluation algorithm Ev (GC, X): A deterministic algorithm which evaluates garbled circuit GC on garbled input X, and outputs a garbled output Y.
- Decode algorithm De (Y, d): A deterministic algorithm which outputs the plaintext output corresponding to Y or ⊥ signifying an error if the garbled output Y is invalid.
- Verify algorithm Ve (C, GC, e): A deterministic algorithm which takes as input a circuit C: \{0,1\}^n \rightarrow \{0,1\}^m, a garbled circuit (possibly malicious) GC, encoding information e, and outputs d when GC is a valid garbling of C, and ⊥ otherwise.

A garbling scheme may satisfy several properties such as correctness, privacy, obliviousness, authenticity and verifiability. We review some of these notions below. The definitions for correctness and authenticity are standard: correctness enforces that a correctly garbled circuit, when evaluated, outputs the correct output of the underlying circuit; authenticity enforces that the evaluator can only learn the output label that corresponds to the value of the function. Verifiability [JKO13] allows one to check that
the garbling of a circuit indeed implements the specified plaintext circuit \( C \). Given that verification succeeds for a candidate \((C, GC, e)\), the garbled output corresponding to a given clear output can be extracted. We provide definitions of correctness, privacy and verifiability as we need it for our \( \pi_{CRS} \) protocol.

**Definition 3.** (Correctness) A garbling scheme \( \text{Garble} \) is correct if for all input lengths \( n \leq \text{poly}(\kappa) \), circuits \( C : \{0,1\}^n \rightarrow \{0,1\}^m \) and inputs \( x \in \{0,1\}^n \), the following probability is negligible in \( \kappa \):

\[
\Pr \left( \text{De}(\text{Ev}(GC, En(e, x)), d) \neq C(x) : (GC, e, d) \leftarrow \text{Gb}(1^\kappa, C) \right) .
\]

**Definition 4.** (Privacy) A garbling scheme \( \text{Garble} \) is private if for all input lengths \( n \leq \text{poly}(\kappa) \), circuits \( C : \{0,1\}^n \rightarrow \{0,1\}^m \), there exists a PPT simulator \( \text{Sim} \) such that for all inputs \( x \in \{0,1\}^n \), for all probabilistic polynomial-time adversaries \( \text{Adv} \), the following two distributions are computationally indistinguishable:

\[
- \text{REAL}(C, x) : \text{run} (GC, e, d) \leftarrow \text{Gb}(1^\kappa, C), \text{and output} \ (GC, En(x, e), d).
- \text{IDEAL-Sim}(C, C(x)) : \text{output} (GC', X, d') \leftarrow \text{Sim}(1^\kappa, C, C(x)).
\]

**Definition 5.** (Verifiability) A garbling scheme \( \text{Garble} \) is verifiable if for all input lengths \( n \leq \text{poly}(\kappa) \), circuits \( C : \{0,1\}^n \rightarrow \{0,1\}^m \), inputs \( x \in \{0,1\}^n \), and PPT adversaries \( \text{Adv} \), the following probability is negligible in \( \kappa \):

\[
\Pr \left( \text{De}(\text{Ev}(GC, En(x, e)), d) \neq C(x) : (GC, e, d) \leftarrow \text{Adv}(1^\kappa, C) \right) V_e (C, GC, e) = d \neq \bot .
\]

We are interested in a class of garbling schemes referred to as projective in [BHR12]. When garbling a circuit \( C : \{0,1\}^n \mapsto \{0,1\}^m \), a projective garbling scheme produces encoding information of the form \( e = (K_i^0, K_i^1)_{i \in [n]} \), and the encoded input \( X \) corresponding to \( x = (x_i)_{i \in [n]} \) can be interpreted as \( X = En(x, e) = (K_i^x)_{i \in [n]} \).

### C Realization of OT Framework Based on DDH

In this section we present an instantiation of the DME (from [PVW08]) based on the DDH assumption:

- \((\text{crs}, t) \leftarrow \text{SetupMessy}(1^\kappa)\) : In messy mode, the \( \text{crs} \) is a non-DDH tuple and it is generated as follows: Sample \( g_0, g_1 \leftarrow R \mathbb{G}, x, y \leftarrow R \mathbb{Z}_p \) and initialize \( h_0 = g_0^x, h_1 = g_1^x \). Set \( \text{crs} = (g_0, g_1, h_0, h_1) \) and the trapdoor \( t \) is set to \( (x, y) \).

- \((\text{crs}, t) \leftarrow \text{SetupDec}(1^\kappa)\) : In decryption mode, the \( \text{crs} \) is a DDH tuple and it is generated as follows: Sample \( g_0 \leftarrow R \mathbb{G}, x, y \leftarrow R \mathbb{Z}_p \) and initialize \( g_1 = g_0^y, h_0 = g_0^y, h_1 = g_1^y \). Set \( \text{crs} = (g_0, g_1, h_0, h_1) \) and the trapdoor \( t \) is set to \( (x, y) \).

- \((\text{pk}, \text{sk}) \leftarrow \text{KeyGen(\text{crs}, \sigma)}\) Given \( \sigma \in \{0,1\} \) and \( \text{crs} = (g_0, g_1, h_0, h_1) \), set \( \text{pk} = (g, h) = (g_0^\sigma, h_0^\sigma) \), where \( \alpha \leftarrow R \mathbb{Z}_p \). The secret key \( \text{sk} \) is set to \( \alpha \).

- \( y \leftarrow \text{Enc(pk, b, m)}\) : Given \( \text{pk} = (g, h) \), \( m \) and \( b \in \{0,1\} \), \( m \) is encrypted as follows: Sample \( s, r \leftarrow R \mathbb{Z}_p \) and set \( u = g_0^s h_0^r, v = g^s h^r \). The ciphertext is \( y = (u, v, m) \) and the corresponding randomness is \( (s, r) \).
− $m \leftarrow \text{Dec}(\text{sk}, y)$: Given $\text{sk} = \alpha$ and ciphertext $y = (c_0, c_1)$, the corresponding plaintext $s$ obtained as $m = c_1/c_0\alpha$.
− $b \leftarrow \text{FindMessy}(\text{pk}, t)$: Given $\text{pk} = (g, h)$ and $t = (x, y)$, if $h = g^x$ then output that branch $b = 0$ is messy else output that branch $b = 1$ is messy.
− $(\text{pk}, \text{sk}_0, \text{sk}_1) \leftarrow \text{DecKeyGen}(t)$: Given $\text{crs} = (g_0, g_1, h_0, h_1)$ in decryption mode and $t = (x, y)$ generate public and secret key pairs as follows: Sample $r_0 \leftarrow R \mathbb{Z}_p$ and compute $r_1 = r_0/y$. Set $\text{pk} = (g, h) = (g_0^r, h_0^r)$, $\text{sk}_0 = r_0$ and $\text{sk}_1 = r_1$.

The above DDH-based instantiation correctly implements a DME scheme and it satisfies Properties 1-5 (Section 4). Proof of Property 1-4 follows from the paper of [PVW08]. Next, we show that it also satisfies Property 5, i.e. the $\text{crs}$ in the messy mode can be sampled using a random oracle. The $\text{crs}$ for the messy mode is required to be a non-DDH tuple. A random oracle query result of size $4|\mathbb{G}|$ bits returns a DDH tuple, with probability:

$$\frac{|\mathbb{Z}_p|^3}{|\mathbb{Z}_p|^4} = \frac{1}{|\mathbb{Z}_p|}$$

With $1 - \frac{1}{|\mathbb{Z}_p|}$ probability the tuple will be non-DDH and hence a valid $\text{crs}$ for messy mode will be generated, satisfying Property 5.

### D Instantiation of DME based on LWE

In this section we provide an instantiation of our DME based on LWE. Before describing the DME instantiation we recall the definition of an LWE encryption scheme from [GPV08], which will be instrumental in the instantiation. The encryption scheme is a collection of following three algorithms:

− LWESetup: Choose a matrix $A \leftarrow \mathbb{Z}_q^{m \times \kappa}$ uniformly at random.
− LWEKeyGen: Choose a secret decryption key $s \leftarrow \mathbb{Z}_q^{\kappa}$ uniformly at random. The public key is the vector $\text{pk} = A^Ts + x \in \mathbb{Z}_q^m$, where $x = (x_1, \ldots, x_m)$ and each $x_i$ is chosen independently from the error distribution $\chi$ for $i \in [m]$.
− LWEEnc($p, b$): To encrypt a bit $b \in \{0, 1\}$, choose a vector $e \in \mathbb{Z}^m$ uniformly at random and set the ciphertext as $(u, c) = (Ae, p^Te + b[\lfloor q/2 \rfloor] \in \mathbb{Z}_q^{m\epsilon+1}$.
− LWE.Dec($s, (u, c)$): Compute $b' = c - s^T u \in \mathbb{Z}$: Output 0 if $b'$ is closer to 0 than to $[\lfloor q/2 \rfloor \mod q$, otherwise output 1.

The above encryption scheme satisfies the notion of IND-CPA security if we assume that the LWE problem is hard for parameters $q = O(\kappa^3), m = O(\kappa \log \kappa)$ (Lemma. 1). The proof appears in the work of [GPV08] and we refer to their paper for more details.

**Lemma 1.** The cryptosystem above is CPA-secure, assuming that LWE is hard for parameters $q, m$.

Next, we will use the encryption scheme in our DME instantiation. In our instantiation we need an additional algorithm, IsMessy, besides the usual algorithms of DME. IsMessy($t, \text{pk}$) answers whether $\text{pk}$ is messy or not, when it is invoked with trapdoor $t$ on public key $\text{pk}$. Now we are ready to instantiate the algorithms for the DME scheme based on the LWE assumption. The description of the algorithms has been borrowed from the paper of [PVW08].
– (crs, t) ← SetupMessy(1^n) : In the messy mode the crs is generated as follows: Sample a matrix \( A \leftarrow \mathbb{Z}_q^{\kappa \times m} \) uniformly at random, along with a trapdoor \( t = (S, A) \) (as described in Section 5.3.2 of [GPV08]). Sample an independent row vector \( v_b \leftarrow \mathbb{Z}_q^{l \times m} \) uniformly at random for every \( b \in \{0, 1\} \). Set 

\[
\text{crs} = (A, v_0, v_1) \quad \text{and} \quad t = (S, A).
\]

– (crs, t) ← SetupDec(1^n) : In the decryption mode the crs is generated as follows: Sample a matrix \( A \leftarrow \mathbb{Z}_q^{\kappa \times m} \) uniformly at random. Choose a row vector \( w \leftarrow \mathbb{Z}_q^{1 \times m} \) uniformly at random. For every \( b \in \{0, 1\} \), sample a secret \( s_b \leftarrow \mathbb{Z}_q^n \) uniformly at random and an error row vector \( x_b \leftarrow \mathbb{X}_q^{1 \times m} \) (i.e., the \( m \) entries are chosen independently from error distribution \( \chi \)). Let 

\[
v_b = s_b^T A + x_b w.
\]

Set 

\[
\text{crs} = (A, v_0, v_1) \quad \text{and} \quad t = (w, s_0, s_1).
\]

– (pk, sk) ← KeyGen(crs, \( \sigma \)) : Given \( \text{crs} \) and \( \sigma \), sample a secret \( r \leftarrow \mathbb{Z}_q^n \) and a row vector \( x \leftarrow \mathbb{X}_q^{l \times m} \). Set 

\[
\text{pk} = r^T A + xv \quad \text{and} \quad \text{sk} = r.
\]

– y ← Enc(pk, b, m) : Given \( \text{pk} = A \), \( m \) and \( b \in \{0, 1\} \), \( m \) is encrypted as 

\[
y = \text{LWEEnc}((A, \text{pk} + v_b), m).
\]

– m ← Dec(sk, y) : Given \( \text{sk} = r \) and ciphertext \( v \), the underlying plaintext message is decrypted as 

\[
m = \text{LWEDec}(\text{sk}, y).
\]

– b ← FindMessy(pk, t) : Given \( \text{pk} \) and \( t = (S, A) \) in messy mode, invoking \( \text{IsMessy}((S, A), \text{pk} + v_b) \) for each \( b \in \{0, 1\} \), outputs the messy branch value for \( b \) and it is correct with overwhelming probability.

– (pk, sk_0, sk_1) ← DecKeyGen(t) : Given the \( \text{crs} = (A, v_0, v_1) \) in decryption mode and the trapdoor \( t = (w, s_0, s_1) \), the public and secret keys pair is formed as follows: 

\[
\text{pk} = (w, s_0, s_1).
\]

The paper of [PVW08] has proven that the above instantiation satisfies correctness and Properties 1-4 of DME. It has also shown in Lemma 7.4 that most of the keys are messy, proving Property 5. In particular, if the \( \text{crs} \) in the messy mode is generated as 

\[
\text{crs} = (A, v_0, v_1) \leftarrow \mathcal{F}_\text{RO}(\text{sid} | | c),
\]

then \( \mathcal{F}_\text{RO}(\text{sid} | | c) \) returns a messy key except with negligible probability for a random value \( c \), where 

\[
\mathcal{F}_\text{RO} : \{0, 1\}^{2\kappa} \xrightarrow{} \mathbb{Z}_q^{\kappa \times m} \times \mathbb{Z}_q^{2l \times m}.
\]