Consistent Adaptive Two-Party Computations

***** Draft, September 30, 2008 *****

Sven Laur[1] and Helger Lipmaa[2]

[1] University of Tartu, Estonia
[2] Cybernetica AS, Estonia

Abstract. We say that a (2-message) protocol is consistent if apart from the usual ideal-world attacks, a malicious server can also—instead of returning the output—force the honest client to halt based on the truth value of some predicate on the client inputs. This can happen for example when the server switches her inputs between answering to client's queries. Thus, the client can detect and then prove to third parties that server's answers to different queries are inconsistent; however, a justified public complaint violates his own privacy. In many applications, where the long-term reputation of a service provider is more important than the privacy of an individual query, consistent protocols can provide adequate protection with a minimal overhead compared to private protocols. Moreover, consistency is an interesting security notion by itself, having applications in say private inference control. We propose an efficient generic protocol for consistent cryptocomputing that combines a formal description of step-by-step computation (like branching programs) with an equivocable and extractable commitment scheme. The commitment scheme must have certificates that make it possible to verify that the output value of the computation is a result of a correct computation process. The new protocol requires two messages per query and uses a number of new (or very recent) cryptographic tools.

Keywords. Branching program, consistency, cryptocomputing, equivocable and extractable commitment.

1 Introduction

We consider two-party adaptive computations—like adaptive oblivious transfer—in the client-server model where the client issues queries to the server, while the server obtains no legitimate output. In such protocols, the server has an input x, and the client makes m adaptive queries q_j. The protocols consist of a one-time initialization phase (than can be costly) and then m query phases. We are interested in polylogarithmic-communication cryptocomputing protocols where during every query phase, the client sends some ciphertexts to the server, who after some computation returns a number of other ciphertexts.

Achieving full simulation-based security in this setting (and in the complexity-theoretic model) is often very costly: using zero-knowledge proofs results at least 3 messages per query, and the only known generic way to achieve sublinear communication [NN01] requires the costly computation of a PCP witness.[1] To reduce both communication and computation, it is customary [NP99a,AIR01,Lip05,IP07,LL07,Lip08] to design protocols that are private in the malicious model but do not aim to provide correctness. In such *semisimulatable* protocols, server's privacy is proved by using a simulation-based approach, but the privacy of the client is proved by using a simpler game-based approach. Constructing 2-message per query semisimulatable protocols is by now a standard practice.

However, privacy is not sufficient in many applications where one also needs some form of *consistency* of protocol outputs. As already shown by Naor and Pinkas [NP99b], just privacy permits selective-failure attacks, in which a malicious sender can induce transfer failures that are dependent on the message that the receiver requests. As an illustrative example, consider *private inference control* [WS04] in the next setting. The server holds a database of private keys that are used to encrypt various content, e.g., documents with different confidentiality levels. Clients have acquired different credentials and the server's task

[1] It is known how to construct 2-message per query simulatable protocols for say adaptive $(n, 1)$-oblivious transfer protocols but they have linear communication. See for example [PVW08].

is to release correct keys. Imagine that the corresponding document server is used in a large organization, such as the army headquarters, and for security reasons the server should not learn which documents are accessed by different clients. At the same time, the server should deny access for clients who do not have appropriate credentials. (Such a goal can be efficiently accomplished with *conditional disclosure of secrets* [GIKM00,AIR01,LL07].) In particular, in private inference control, the client should be able to distinguish between denial of service attacks, where the server acts maliciously, and legitimate denials, where the client has no right to obtain a corresponding key. Moreover, to protect the service against inside attacks the client should be able to prove to third parties that the denial is illegitimate.

OUR CONTRIBUTION. To improve on the efficiency of secure protocols and on the other hand to decrease the influence of selective-failure attacks in semisimulatable protocols, we advocate an intermediate approach. We allow the server's misbehavior to depend on the concrete inputs of the client. We require that an honest client must always be able to detect and prove malicious behavior. We also require the protocol to remain client-private unless the client issues a complaint. Moreover, a publicly issued complaint should reveal only a single bit of information ("client's inputs were such that client's output was incorrect") about client's inputs to the malicious server. Thus, the corresponding security notion, *consistency*, is strictly weaker than security in the malicious model. In other words, secure protocols prevent, consistent protocols detect, and semisimulatable (private) protocols permit selective failures without even detecting them.

We now give an informal description of the new consistent cryptocomputing (that is, computing on ciphertexts) protocol. The protocol can be based on any semisimulatable protocol where the computation of the final result is done by retrieving server's output f from some memory location after following some formally established computation process. In the consistent protocol, the client will follow *exactly* the same computation process but instead of f it receives it's "certificate". The value of f should be extractable from the certificate, and the certificate should prove that the server retrieved f by following correct computation and from a correct location. More precisely, there should exist a server's input that is consistent with the certificate. Thus our construction "marries" a formal computation process with an extractable (DAG-commitment) scheme. Despite of this simple description (and in fact, simple construction), the security definitions and the security proofs are not obvious at all. We will now proceed to explain technical intricacies.

The definition of consistency is our first contribution. Formally, a malicious adversary can enforce a complaint if client's inputs satisfy certain halting predicate π. We modify the ideal world by allowing a malicious server to submit to the trusted third party a description of an efficient halting machine Halt-Machine. At the jth query q_j, the trusted third party halts if $\mathsf{Halt\text{-}Machine}(q_1, \ldots, q_j) = \top$. Otherwise, he returns the correct function value. A protocol is *consistent* if for every adversary in the real world, there exists a simulator in the (modified) ideal world, such that the restricted joint output distributions in the real and ideal world are computationally indistinguishable; see Sect. 2. The machine Halt-Machine can be seen as implementing m halting predicates $\pi_1(q_1)$, $\pi_2(q_1, q_2)$, ..., $\pi_m(q_1, \ldots, q_m)$. If after round j, the client issues a valid complaint about server's incorrect behavior, the server will know that $\pi_j(q_1, \ldots, q_j)$ holds. Different protocols allow the server to enforce different classes of halting predicates. In particular, there is a trade-off between the protocol complexity and the restrictiveness of enforceable halting predicates. See Sect. 6.

We need so called DAG-commitment schemes that satisfy relatively strong security notions, equivocability and DAG-extractability. As our second contribution, in Sect. 3, we define DAG-commitment schemes and their security notions, and then propose a concrete efficient DAG-commitment scheme. A DAG-commitment scheme enables to commit to the set of sink labels of a DAG such that, given a short certificate, one can later verify the value of a single sink label of the committed DAG. The new construction combines an extension of hash trees with a suitable usual commitment scheme. We need the DAG-commitment scheme to be DAG-extractable in the sense of [BL07] (where this property was called knowledge-binding). That is, we need a DAG-commitment scheme where given a succinct commitment and access to the code and the random coins of the committer, one can construct a list of candidate committed elements, such that the probability that an adversary can later open the commitment to an element not in the list is negligible. We prove by using a white-box reduction that every binding DAG-commitment scheme is also DAG-extractable, and thus it suffices for the usual commitment scheme to be binding. We also prove that the new DAG-commitment scheme is equivocable when the usual commitment scheme is equivocable. Note that it is possible to achieve equivocability in the complex-theoretic model by requiring the participants to do additional work in the initialization phase; this does not increase the number of rounds per query.

Our third contribution is the consistent cryptocomputing protocol itself. It enables to cryptocompute any functionality that can be computed by a family of polynomial-size branching programs where the values of the sinks are independent from each other. (This is the case when uses a complete binary decision tree to cryptocompute oblivious transfer like in [KO97,Ste98,Lip05] or in the case of conditional disclose of secrets [AIR01,LL07].) It combines recent techniques by [IP07,Lip08] for semisimulatable cryptocomputing with an equivocable and DAG-extractable DAG-commitment scheme in a manner we described earlier: it follows the private branching program protocol of [IP07], but instead of just the original output to the protocol, it also retrieves a certificate which proves that this output was retrieved from a correct branch of the branching program. See Sect. 4 for more.

The new consistent cryptocomputing protocol is secure against unbounded malicious clients, uses 2 messages per single query, and has communication and computation comparable to that of the underlying semisimulatable protocol. We also present more efficient protocols for specific functionalities like adaptive m-out-of-n oblivious transfer and adaptive conditional disclosure of secrets (CDS), see Sect. 5. In particular, consistent adaptive CDS is directly applicable in private inference control where one is actually *not* interested in fully-simulatable security. This shows that consistent computing is not only a weaker security notion but it is interesting by itself. Finding other applications is an interesting open problem.
NOTATION. We assume that k is the security parameter, and that $A = \{A_k\}$ is a non-uniform adversary.

2 Consistent Computation: Security Definitions

PRELIMINARIES. Many privacy-preserving applications—for example oblivious transfer and CPIR—are asymmetric, so that only one participant (*client*) obtains output whereas the other (*server*) is just the service provider. Their inputs are often asymmetric, too, with usually client's input consisting of the query index but server's input consisting of the whole list. More precisely, the server holds an input x and the client makes m adaptive queries $q_1, \ldots, q_m$ to retrieve the outputs of some functions $f_1(q_1, x), \ldots, f_m(q_m, x)$.

We consider the security of a two-party protocol Π^f for adaptive computation of functionality f. In the ideal world, the participants forward their inputs to a trusted third party (TTP) that computes f and sends the corresponding outputs back to both parties. That is, the TTP receives x from the server, and for every client's input q_j he sends $\perp$ to the client, if the server issues an abort command, or $f_j(q_j, x)$, otherwise. Since both parties can halt in the real world, the TTP halts if one of the participants sends a halting instruction in the ideal world. Such formalism captures all client-server protocols where server's input is non-adaptive. A protocol Π^f is *correct*, if in the case of honest participants, the client obtains the desired outputs.

Consider an arbitrary non-uniform polynomial-time malicious server and an honest client in the real world. A protocol is *client-secure* if there exists a non-uniform polynomial-time simulator acting as the server in the ideal world such that the joint output distribution of the server and the client is computationally indistinguishable from the joint output in the real world. In the ideal world the honest client just submits his inputs and outputs the values received from TTP. *Server-security* is defined analogously; because server does not have an output, server-secrecy coincides with *server-privacy*. Thus, a protocol is *(simulatably) secure* if it is client-secure and server-private. A protocol is *statically private* if malicious participants and their

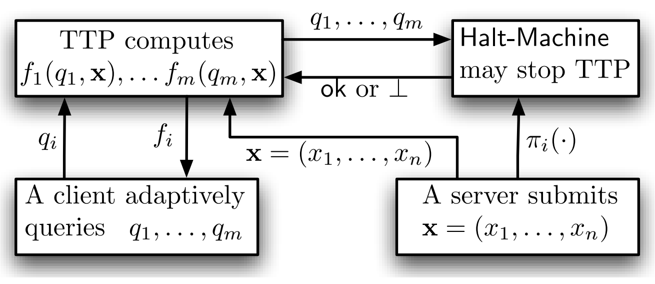

Fig. 1. Ideal world model for consistent computations. A malicious server can send a description of halting predicates $\pi_i(\cdot)$ to Halt-Machine but the communication is unidirectional.

simulators are unbounded, and the output distributions are statistically indistinguishable. (This uses a weak definition of statistical security that is common in the existing literature on two-message oblivious transfer protocols. According to a standard definition, it is required that the simulator works in time, polynomial in the working time of the malicious party.)

Security requirements in the client-server model are often asymmetric, as there are few servers and many clients. Servers are also more concerned about their reputation than clients. Thus, and to also increase efficiency, in the case of client-server protocols one often considers the notion of *semisimulatable privacy* [NP99a,AIR01,Lip05,IP07,LL07,Lip08], where only client-privacy is guaranteed. More precisely, client-privacy is only protected in the sense of *CPA-security*; that is, it is required that the server is not able to distinguish which of the two candidate inputs the client is using. Server-privacy is defined as before. It is possible to design efficient 2-message semisimulatable protocols for many applications [IP07,Lip08].

A semisimulatable protocol provides no guarantees about the correctness of client's output. On the other hand, a simulatable protocol guarantees that a client can detect server's malicious behavior. More precisely, an honest client detects all possible deviations from the protocol that *may* change output for *some* client's input. Thus, the client can freely complain if needed since the server knows herself when she acts maliciously and thus a complaint does not give away any information about client's inputs. The generic way to implement such a "universal fraud detection" is based on zero-knowledge proofs. Although zero-knowledge proofs can be compressed by employing the PCP theorem [NN01], the existing techniques for this are only asymptotically communication-efficient and in practice, the resulting protocols have a prohibitively large communication and computation overhead. Due to the nature of zero-knowledge proofs, secure protocols seem to require 4 messages per query in the complexity-theoretic model.

CONSISTENCY. We propose a security notion that allows more efficient implementations than simulatable protocols but is considerably stronger than semisimulatability. First we note that the universal fraud detection mechanism is not always necessary. For example, any semisimulatable 1-out-of-n oblivious transfer is formally also simulatably secure because there are no consistency restrictions to client's outputs; acting as an honest client it is possible to extract a server's database that perfectly models client's outputs in the real world. To tackle this problem, one often constructs *verifiable* protocols where the parties first commit their inputs and then verify the consistency between the committed inputs and the outputs [CC00].

Intuitively, a protocol is consistent if it includes an existential fraud detection mechanism so that a client detects malicious behavior if and only if it infects the output that corresponds to her actual inputs. Therefore, by issuing a complaint the client releases a single bit of information ("my inputs were such that the result returned by the server was incorrect"). For formal definition of consistency, we make explicit changes into the ideal world model, see Fig. 1. In the new model, a malicious server can additionally influence the outputs of the client, after server's inputs x are submitted to the TTP, as follows. The server sends to the TTP a description of stateful randomized polynomial-time predicates $\pi_1(q_1), \ldots, \pi_m(q_1, \ldots, q_m)$. (Polynomial time-bound means here that there exists a $c > 0$ such that the time needed to evaluate $\pi_i(\cdot)$ is at most k^c times larger than the running time of the malicious server, where k is the security parameter.) Altogether such predicates describe a halting machine $\mathsf{Halt\text{-}Machine}$. Computation in the ideal world proceeds as follows. Given the jth query q_j, TTP feeds q_j into $\mathsf{Halt\text{-}Machine}$ that evaluates $\pi_j(q_1, \ldots, q_j)$. The TTP sends to the client $f_j(q_j, x)$ if $\pi_j(q_1, \ldots, q_j)$ holds, and $\perp$ otherwise.

Definition 1 (Consistency). *A protocol Π^f is* consistent *if for any non-uniform polynomial-time malicious client or server, there exists a non-uniform polynomial-time adversary (*simulator*) in the modified ideal world such that the restricted joint output distribution of the real world and the modified ideal world are computationally indistinguishable for any set of inputs, and for any auxiliary string* hist *that is given as input both to the malicious party and the simulator.*

Here, the explicit use of hist assures sequential composability. Note that a consistent protocol is strictly more secure than the straightforward sequential composition of semisimulatable protocols, since in a consistent protocol it is impossible to influence client's output by changing server's inputs during the protocol.

Whenever a client issues a valid public complaint, some information is leaked. As the ideal and real world outputs are indistinguishable, it is sufficient to consider the effect of complaints in the ideal world. If the client notifies the server whether the jth computation failed or not, then the malicious server learns the value of $\pi_j(q_1, \ldots, q_j)$. Therefore, the decision to issue a complaint must depend on whether the client values more the utility of $f_j(q_j, x)$ or the privacy of $\pi_j(q_1, \ldots, q_j)$. Here, the precise predicate π_j is unknown to him. To limit privacy exposure, it is important to limit the possible class of enforceable halting predicates $\pi_i(\cdot)$. We consider this explicitly in Sect. 6, where we define various classes of enforceable predicates.

If a client reaches the accepting state, then his output is the same as it would be in the ideal world. Thus, a non-accepting client can prove to the court, without interacting with the server, that server's behavior was indeed malicious. Also, an honest server can easily disprove false accusations, since server's honest behavior leads always to acceptance. If a protocol is only semisimulatable, then to disprove a false accusation the honest server must repeat her computations and prove in zero-knowledge that her actions were correct. Even that might be insufficient, since sometimes it is possible to change protocol inputs to hide malicious behavior. Moreover, to prove such claims, the server has to store all computations which is clearly impractical if she serves many clients. Similar concerns have been addressed earlier in the literature, see [AL07] for discussion and further references. However, all models provided by Aumann and Lindell allow significantly larger exposure, since they do not use halting predicates and in the case of a successful cheating they give the inputs of honest parties directly to the adversary.

3 DAG-Commitment Schemes

Within this paper, a DAG is directed acyclic graph with a single source and n sinks. All protocols presented in this paper use DAG-commitment schemes, a generalization of commitment schemes that makes it possible to commit to the set of sinks of any DAG. In this section, we give a formal definition of the syntax and semantics of DAG-commitment schemes. The most novel property here is DAG-extractability in the sense of [BL07] (where this property was called knowledge-binding). This is followed by a new construction.

DEFINITIONS. Here, and in the next, $\mathcal{G}$ is a short encoding of a DAG, and $x = (x_1, \ldots, x_n) \in \mathcal{M}^n$ is a list of $\mathrm{poly}(k)$ elements mapped to n sinks of $\mathcal{G}$. We will assume that $\mathcal{G}$ is publicly known and fixed, and thus usually omit it as an argument to com, cert and open. A *DAG-commitment scheme* is a quadruple $\mathsf{gc} = (\mathsf{gen}, \mathsf{com}, \mathsf{cert}, \mathsf{open})$ of efficient algorithms:

- Key-generator $\mathsf{gen}(1^k)$ generates public parameters ck.
- Commitment algorithm outputs a pair $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(\mathcal{G}, x)$ of commitment and decommitment values.
- For $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(\mathcal{G}, x)$ and $i \in \{1, \ldots, n\}$, $\mathsf{cert}_{\mathsf{ck}}(\mathcal{G}, d, i)$ returns a certificate ct_i.
- If $\mathsf{ct}_i = \mathsf{cert}_{\mathsf{ck}}(\mathcal{G}, d, i)$ and $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(\mathcal{G}, x)$ then $\mathsf{open}_{\mathsf{ck}}(\mathcal{G}, c, \mathsf{ct}_i) := (i, x_i)$. Incorrect certificates lead to $\mathsf{open}_{\mathsf{ck}}(\mathcal{G}, c, \mathsf{ct}_i) = \perp$.

Every DAG-commitment scheme must be binding and hiding. For binding, consider the next game:

1. Challenger generates $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and sends ck to A_k.
2. A_k generates a commitment $\hat{c}$ and two certificates $\hat{\mathsf{ct}}_0, \hat{\mathsf{ct}}_1$.
3. A_k wins if the certificates allow to open $\hat{c}$ to different values of x_i. That is: Let $(i_b, x_b) \leftarrow \mathsf{open}_{\mathsf{ck}}(\hat{c}, \hat{\mathsf{ct}}_b)$ for $b \in \{0, 1\}$. A_k wins if and only if $i_0 = i_1 \neq \perp$ and $\perp \neq x_0 \neq x_1 \neq \perp$.

We say gc is (τ_k, ε_k)-binding if $\Pr[A_k \text{ wins}] \leq \varepsilon_k$ for any non-uniform adversary A_k that works in time τ_k.

For hiding (in the sense of CPA-security), consider the next game:

1. Challenger generates $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and sends ck to A_k.
2. A_k generates two lists x^0, x^1 of sink labels, and sends them to the challenger. It is required that the databases have elements only from $\mathcal{M}$, and that $|x^0| = |x^1| = n$, where $n = \mathrm{poly}(k)$ is the number of sinks of a fixed DAG $\mathcal{G}$.
3. Challenger generates a random bit $b \leftarrow \{0, 1\}$, and sets $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(x^b)$. He sends c to A_k.
4. Adversary makes a number of queries to $\mathsf{cert}_{\mathsf{ck}}(d, \cdot)$, where she is restricted to query $\mathsf{cert}_{\mathsf{ck}}(d, \cdot)$ only on indexes i with $x_i^0 = x_i^1$.
5. Adversary outputs a bit b', and wins if $b = b'$.

We say gc is (τ_k, ε_k)-hiding if $2 \cdot |\Pr[A_k \text{ wins in the hiding game}] - 1/2| \leq \varepsilon_k$ for any non-uniform adversary A_k that works in time τ_k.

A DAG-commitment scheme is *computationally hiding* if it is $(poly(k), k^{-\omega(1)})$-hiding, and *computationally binding* if it is $(poly(k), k^{-\omega(1)})$-binding. In the case of unbounded adversaries, we speak respectively about *statistical hiding* and *statistical binding*.

A DAG-commitment scheme gc is *perfectly equivocable* [CIO98] if there exist three additional algorithms $\widehat{\mathsf{gen}}$, $\widehat{\mathsf{com}}$ and equiv, such that no unbounded adversary A can distinguish between the following two experiments for any fixed DAG $\mathcal{G}$ with $n = \mathrm{poly}(k)$ sinks and even after submitting many vectors x:

- The first experiment models the normal operation of the DAG-commitment scheme. It sets $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$, and sends ck to A. Then, A gets an access to the oracle $\mathcal{O}$ that, given $x = (x_1, \ldots, x_n)$, computes $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(x)$, $\mathsf{ct}_i \leftarrow \mathsf{cert}_{\mathsf{ck}}(d, i)$ for $i \in \{1, \ldots, n\}$, and outputs $(c, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$.
- The second experiment models the faked commitments. First, it sets $(\mathsf{ek}, \mathsf{ck}) \leftarrow \widehat{\mathsf{gen}}(1^k)$, and sends ck to A. Then, A gets an access to the oracle $\hat{\mathcal{O}}$ that, given $x = (x_1, \ldots, x_n)$, computes $(\hat{c}, \eta) \leftarrow \widehat{\mathsf{com}}_{\mathsf{ck}}(\mathsf{ek}, n)$, $\hat{\mathsf{ct}}_i \leftarrow \mathsf{equiv}_{\mathsf{ek}}(\hat{c}, \eta, i, x_i)$, and outputs $(\hat{c}, \hat{\mathsf{ct}}_1, \ldots, \hat{\mathsf{ct}}_n)$.

Clearly, adversary's chances to distinguish the experiments do not increase if it queries the certificates adaptively depending on the received commitment and certificate values: formally, we can build a wrapper around the adversary that first queries the whole tuple and then gradually releases the requested elements. The definition of *statistical* and *computational equivocability* are analogous.

One can build non-interactive equivocable (non-DAG) commitment schemes based on any one-way functions in the common reference string (CRS) model [CIO98,Di 02,DG03]. In the standard model, 3 rounds are needed to implement an equivocable commitment scheme. Thus, all subsequent results that use equivocable commitment schemes require 3 messages. However, all but the last message can be transferred

during the initialization phase, so this is not a problem. The Pedersen commitment scheme [Ped91] is perfectly equivocable and the Fujisaki-Okamoto commitment scheme [FO97] is statistically equivocable in the CRS model. In non-interactive equivocable Pedersen commitment scheme, the CRS is $(\mathbb{G}, q, g, h)$ where $\mathbb{G}$ is a group of order q, and $g \neq h$ are its two random generators. The equivocation key is equal to $x = \log_g h$. In the interactive version of this scheme, the committer and the receiver jointly compute random g and h.
LIST-EXTRACTABILITY. We also need some sort of extractability. More precisely, given a certain trapdoor one should be able to reconstruct which elements were DAG-committed. However, because the commitments c are very short, they can be opened in many different yet legitimate ways. As a result, we use a somewhat weaker DAG-extractability property proposed by Buldas and Laur [BL07] (they called it knowledge-binding). Essentially, a DAG-commitment scheme is DAG-extractable if the DAG-committed elements are efficiently extractable given a white-box access to the committing algorithm and to the used randomness. However, the definition given in [BL07] has some minor technical shortcomings. In particular, the definition was given in a standalone model where the adversary did not have input. As a result their definition is not sequentially composable. We slightly extend the definition. Namely, for DAG-extractability, consider the next game, where $A = \{A_k\}$ is a non-uniform adversary, $\mathcal{K}_{A_k}$ is some fixed "extractor" machine, and history hist and advice advice are any fixed bitstrings:

1. Generate a new public key $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and a new random tape ω.
2. Adversary gets hist, ck as inputs and ω as her random tape. She outputs a DAG-commitment c to a "list" of size n, $(c, n) \leftarrow A_k(\mathsf{hist}, \mathsf{ck}; \omega)$.
3. Extractor gets hist, ck and ω as inputs. Extractor guesses some candidate list of sink labels, $(\hat{x}_1, \dots, \hat{x}_n) \leftarrow \mathcal{K}_{A_k}(\mathsf{hist}, \mathsf{ck}, \omega)$.
4. On input advice, the adversary outputs a set of certificates $(\mathsf{ct}_1, \dots, \mathsf{ct}_m) \leftarrow A_k(\mathsf{advice})$.
5. Adversary wins if she output at least one certificate that is consistent with the commitment and that corresponds to a list element, not correctly guessed by the extractor, i.e., if $\exists j : (\perp \neq (i, x) = \mathsf{open}_{\mathsf{ck}}(c, \mathsf{ct}_j) \wedge x \neq \hat{x}_i)$.

A DAG-commitment scheme gc is $(t_k, \tau^e_k, \varepsilon^e_k)$*-DAG-extractable* if for every t_k-time stateful randomized algorithm A_k there exists a dedicated τ^e_k-time extractor machine $\mathcal{K}_{A_k}$ such that $\max \Pr[A_k \text{ wins}] \leq \varepsilon^e_k$, where the maximum is taken over the choice of $\mathsf{hist}, \mathsf{advice} \in \{0,1\}^{t_k}$.

If a commitment scheme is not binding then it is trivially not DAG-extractable. More precisely, if gc is not $(\tau^b_k, \varepsilon^b_k)$-binding, then the probability ε^e_k to violate—in closely related time $\tau^e_k = (1 + o(1))\tau^b_k$—DAG-extractability is at least ε^b_k. Analogously to the case of proofs of knowledge [BG92], we define ε^b_k to be the knowledge error. A DAG-commitment scheme gc is *(computationally) DAG-extractable* if there exists a polynomial p such that $\tau^e_k = p(t_k)/(\varepsilon^e_k - \varepsilon^b_k)$ for any polynomial t_k.

Given this asymptotic definition of DAG-extractability, we obtain the next result. (A quite involved proof of this theorem is given in App. A.)

Theorem 1. *Any binding DAG-commitment scheme is also (white-box) DAG-extractable.*

Since in the next we use several (DAG-)commitment schemes in parallel, in such cases we denote the algorithms by prefixing the name of the concrete commitment scheme. For example, then we talk about algorithm $\mathsf{gc.open}$.
DOUBLE-LAYERED DAG-COMMITMENT PROTOCOL. Fix a DAG $\mathcal{G}$ with n sinks. Our construction of a DAG-commitment scheme has two layers. The lower layer is just an ordered list of individual commitments to n sink labels. As the second layer, we use a generalization of hash trees to DAG's to compress the list into a succinct root digest c_*. More formally, the *double-layered hash commitment* $\mathsf{dlh} = (\mathsf{dlh.gen}, \mathsf{dlh.com}, \mathsf{dlh.cert}, \mathsf{dlh.open})$ is specified by a conventional commitment scheme

$\mathsf{cs} = (\mathsf{cs.gen}, \mathsf{cs.com}, \mathsf{cs.open})$ and a collision-resistant hash function family $\{\mathcal{H}_k\}$. The setup procedure $\mathsf{dlh.gen}$ outputs (ck, h) for $\mathsf{ck} \leftarrow \mathsf{cs.gen}(1^k)$, $h \leftarrow \mathcal{H}_k$. To commit to $x = (x_1, \ldots, x_n)$, one first computes $(c_i, d_i) \leftarrow \mathsf{cs.com}_{\mathsf{ck}}(x_i)$ for $i \in \{1, \ldots, n\}$ and then computes a root hash c_* from $c_1, \ldots, c_n$ by using the standard recursive hash tree methodology as applied to DAG $\mathcal{G}$. One then outputs (c_*, n) as the commitment and $d_* = (c_1, \ldots, c_n, d_1, \ldots, d_n)$ as the decommitment. Let $\mathsf{t\text{-}cert}(c_i) = (c_i, \ldots)$ be the minimum amount of information needed to recompute the root hash c_* from the ith position in the corresponding DAG. A certificate for x_i is $(d_i, \mathsf{t\text{-}cert}(c_i)) \leftarrow \mathsf{dlh.cert}_{(\mathsf{ck},h)}(d_*, i)$. To open, one first validates $\mathsf{t\text{-}cert}(c_i)$ with respect to (c_*, n), and on success outputs $x_i \leftarrow \mathsf{cs.open}_{\mathsf{ck}}(c_i, d_i)$.

Intuitively, it is clear that dlh inherits the hiding and equivocability properties of the lower level commitment scheme, as the hashing level only compresses information. Similarly, it is binding if the lower level commitment is binding.

Theorem 2. *Assume that $\mathcal{H}$ is a collision-resistant hash function family and that* cs *is a (conventional) commitment scheme. Then* dlh *inherits the hiding, binding and equivocability from* cs*. Hiding and equivocability also preserve statistical and perfect security. The efficiency of the scheme is comparable to the hash tree.*

Proof. The claims about hiding and binding are evident. For the equivocability note that given the equivocation key ek for commitment scheme cs, it is possible to use $\mathsf{cs.}\widehat{\mathsf{com}}$ to generate a list $\hat{c}_1, \ldots, \hat{c}_n$ of fake commitments for lower level that can be later opened to any values using the function $\mathsf{cs.equiv}_{\mathsf{ek}}$. Thus, preservation of equivocability follows directly from the definition. $\square$

In practice, the Pedersen commitment scheme is a good choice for cs, since its public key is a group element $y \in \langle g \rangle$ such that computing the discrete log $x = \log_g y$ is intractable. Here, x is also the equivocability trapdoor. Hence, the key can be generated jointly by sender and receiver by using a secure three-message multiplication protocol to multiply two random group elements. Alternatively, the client may specify y since the Pedersen commitment is perfectly hiding for all y. Then, we lose equivocability unless we are willing to find the discrete logarithm of y in exponential time.

4 Generic Protocol for Consistent Cryptocomputing

Following [SYY99], by *cryptocomputing* we mean computing on encrypted data. In [IP07], Ishai and Paskin generalized the earlier computationally-private information retrieval protocols by [KO97,Ste98,Lip05] from complete ordered binary decision trees to arbitrary branching programs. This protocol was then refined in [Lip08]. The resulting semisimulatable protocol PrivateBP (*private branching programs*) can cryptocompute any functionality $f(q, x)$, where q is client's input and x is server's input, given that for any fixed x $f(\cdot, x)$ can be computed by a polynomial-size branching program, that is, belongs to the complexity class $\mathbf{L/poly}$ [Weg00]. For completeness's sake, we first describe a simple variant of PrivateBP; see [IP07,Lip08] for a precise description. After that, we show how to make this protocol consistent. We assume that the reader is familiar with the notions like branching program [Weg00] and (single-database) $(n, 1)$-CPIR [CGKS95,KO97].

PREVIOUS WORK: LIPMAA'S EFFICIENT $(2, 1)$-CPIR PROTOCOL. In [Lip05], Lipmaa proposed an efficient two-message client-private $(2, 1)$-CPIR protocol for ℓ-bit strings, see [Lip08] for a succinct description. This protocol is based on Damgård-Jurik cryptosystem [DJ01], and its security is based on the Decisional Composite Residuosity Assumption [Pai99]. Importantly, both messages of Lipmaa's protocol have length $\leq \ell + 2k$, and it can be applied for arbitrary values of ℓ. Denote the first message of Lipmaa's $(2, 1)$-CPIR protocol by $Q(\ell, q)$ and the second message by $R(\ell, x, Q)$, where $q \in \{0, 1\}$ is client's input and $x = (x_0, x_1)$ is server's input. Thus $|Q(\ell, q)|, |R(\ell, x, Q)| \leq \ell + 2k$. As shown in [Lip05], there exists a compress function C, such that $C(\ell, \ell', Q(\ell, q)) = Q(\ell', q)$ for any $\ell' < \ell$.

PREVIOUS WORK: THE PRIVATEBP PROTOCOL [IP07,LIP08]. Following [IP07,Lip08], one can cryptocompute an arbitrary class $\mathcal{F}$ of functions $f : \{0,1\}^n \rightarrow \{0,1\}^\ell$ as follows. (Our description and notation is based on [Lip08].) Assume that P_f is an efficient *layered* branching program [Weg00] that computes f. Fix server's input f and a branching program P_f that computes f. Let $\mathsf{len}(P_f)$ be its length and $\mathsf{size}(P_f)$ its size. The sinks of P_f are labeled by client's possible outputs. Let $\ell_{\max} := \ell + (\max_{g \in \mathcal{F}} \mathsf{len}(P_g) + 2)k$. The client submits her input bits q_i as $Q_i := Q(\ell_{\max}, q_i)$. At every node of the branching program that is labeled by a variable q_i, given input values b_0, b_1 to the node, and Q_i, the output value of the node will be $R(\ell^*, (b_0, b_1), C(\ell_{\max}, \ell^*, Q_i))$, where $\ell^* := |b_0| = |b_1|$. Thus the output of the branching program is equal to a $\max_g \mathcal{F}$-times application of $R(\cdot, \cdot, \cdot)$ to some sink label, selected by the encrypted branching variables. At the end of the cryptocomputing, the server returns the output value to the client who computes the sink label in question by recursively applying the local decoding process of the $(2,1)$-CPIR protocol $\max_{g \in \mathcal{F}} \mathsf{len}(P_g)$ times.

Thus, client-communication is upper-bounded by $n \cdot \ell_{\max}$. Server-communication consists of one message with length $\ell_{\max}$. Server's computation is linear in $\mathsf{size}(P_f)$. (For simplicity's sake, here and in the following we assume that the server does a unit amount of work at every node.) Therefore, every set $\mathcal{F}$ pf functions has a client-private cryptocomputing protocol with total communication $\leq (n+1)\ell_{\max}$. See [IP07,Lip08] for a precise description and generalizations. In particular, in [Lip08] it was shown that a variation of the PrivateBP has communication $(1 + o(1)\ell + \Theta(n \cdot \max_{g \in \mathcal{F}} \mathsf{len}(P_g))k$.

The described protocol by itself does not guarantee server-privacy. One can use the conditional disclosure of secrets protocol from [LL07] to achieve server-privacy, without losing too much in efficiency. In a nutshell, one randomizes the sink labels of the branching program so that in the first level of branching-program computation, the resulting ciphertext encrypts a value, statistically close to random, if client's encrypted input used for branching is invalid, that is, non-binary. See [Kal05,IP07] for alternative techniques and [Lip08] for more discussion. We assume that the PrivateBP protocol guarantees server-privacy.

OUR CONTRIBUTION: CONSISTENT CRYPTOCOMPUTING. We are interested in consistent cryptocomputing for adaptive protocols. In an *adaptive m-out-of-n f-protocol*, the server has an input x. Client's input is $q = (q_1, \dots, q_m)$. In each adaptive phase the client learns $f(q_j, x)$ if q_j is a valid input, and $\perp$ otherwise. Differently from the previous subsection, we assume that f is a publicly known function and that the sink values of the branching program are independent of each other. This is a serious limitation compared to the server-private protocols of [IP07,Lip08] where it was required that for every possible fixed server's input there exists a polynomial-size branching program and there was no limitation on the sink values. Still, one can construct efficient consistent protocols for several interesting functionalities.

We first give an informal description of the protocol. The protocol uses PrivateBP as the basis, but it can also be based on any other semisimulatable protocol where the computation of the final result is done by retrieving server's output f from some memory location after following some formally established computation process. In the consistent, protocol the client will follow *exactly* the same computation process but retrieve a certificate of f. The value of f should be extractable from the certificate. Moreover, the certificate should prove that the server retrieved f by following correct computation and from a correct location. More precisely, there should exist a server's input that is consistent with the certificate. Thus our construction "marries" a formal computation process (in this case, PrivateBP—but in the next section we will talk about other possibilities) with an extractable (DAG-commitment) scheme. The next formal description explicitly assumes the use of PrivateBP.

Let P be a branching program for f, on a DAG $\mathcal{G}$, such that the sink labels correspond to different server inputs x_j. Next, we construct an adaptive m-out-of-n f-protocol $\mathsf{Ada} = \mathsf{Ada}[\mathsf{gc}]$ from Lipmaa's $(2,1)$-CPIR protocol Γ and an equivocable and DAG-extractable DAG-commitment scheme gc that handles ℓ-bit elements and has $\leq \ell_{\mathrm{s}}(\mathcal{G})$-bit certificates:

- In the setup phase of Ada, the trusted dealer runs $\mathsf{ck} \leftarrow \mathsf{gc.gen}(1^k)$ to generate public parameters of the DAG-commitment scheme.
- In the commitment phase of Ada, the server first computes $(c, d) \leftarrow \mathsf{gc.com}_{\mathsf{ck}}(x)$ and sends the commitment c to the client. Then he computes $\mathsf{ct}_i \leftarrow \mathsf{gc.cert}_{\mathsf{ck}}(d, i)$ for every sink $i \in \{0, \ldots, n-1\}$.
- In the ith query phase of Ada with client's query q_i:
  1. The client sends $c_i \leftarrow Q(\ell_{\mathsf{s}}(\mathcal{G}) + (\mathsf{len}(P)+2)k, q_i)$ to the server.
  2. The server cryptocomputes Ada on client's input c_i, by using the semisimulatable PrivateBP protocol, and returns the result r back to the client.
  3. Let r' be the result of $\mathsf{len}(P)$-times application of Γ's decoding procedure to r. The client outputs x_{q_i} if $\mathsf{gc.open}_{\mathsf{ck}}(P, c, r') = (q_i, x_{q_i}) \neq \perp$, and $\perp$, otherwise.

For clarity's sake, we have described Ada using a trusted setup phase; we will later show how to eliminate the need for trusted dealer by running a certain multiparty protocol. Using trusted setup phase makes the security proofs more modular. Also, many semisimulatable oblivious transfer protocols [Lip05,AIR01,LL07] are given in the trusted setup model.

Theorem 3. *If* gc *is equivocable and binding and* Γ *is computationally client-private, then* Ada *is consistent.*

(Proof is given in App. B.)

Corollary 1. *If* gc *is statistically equivocable, then* Ada *is statistically server-private.* □

The construction of the knowledge-extractor was essentially white-box. We can however show the following.

Theorem 4 (Black-box reduction). *If the server's input space S is polynomial in the security parameter and* gc *is binding, then the knowledge-extractor $\mathcal{K}$ can be constructed by using black-box methods.*

Proof. Note that the advice of A^{ideal} is from the set S. Hence, one can evaluate $A^{ideal}(\mathsf{ck}, \mathsf{hist}, \mathsf{advice}; \omega)$ for all $|S|$ advice strings and reconstruct all valid certificates and corresponding openings in polynomial time. Since gc is binding, then the probability of a double opening is negligible, and it is possible to a uniquely reconstruct $\hat{x}$. □

For example, in the case of m-out-of-n adaptive oblivious transfer, where $|S| = n^m$, one can construct a polynomial-time black-box extractor $\mathcal{K}$ if m is a constant, or a subexponential-time black-box extractor $\mathcal{K}$ if $m = o(n/\log n)$.

5 Efficient Consistent Protocols For Specific Problems

Recall that Ada was based on an underlying semisimulatable protocol, where the client's outputs corresponded to sinks in a certain DAG. To make the protocol consistent, we let the client to obtain short certificates that the sink labels agree with the previous commitment. Next, we give a few examples.
ADAPTIVE OBLIVIOUS TRANSFER. In the special case of adaptive m-out-of-n oblivious transfer protocols, the consistent protocol of Sect. 4 is just a semisimulatable oblivious transfer protocol, based on Lipmaa's $(n, 1)$-CPIR protocol [Lip05], applied to the database of certificates. One can instead apply an arbitrary semisimulatable $(n, 1)$-CPIR protocol to the database of certificates to achieve the same result. In particular, one can use the Gentry-Ramzan $(n, 1)$-CPIR protocol [GR05] to minimize communication compared to the general solution of Sect. 4.

SECRET SHARING. The Benaloh-Leichter secret sharing scheme [BL88] for arbitrary adversary structures use also a tree structure, and propagate the secret up on the tree according to some fixed rules. We can make their protocol consistent by instead propagating the corresponding certificate. Details are omitted.
CONSISTENT ADAPTIVE CONDITIONAL DISCLOSURE OF SECRETS. In a conditional disclosure of secrets (CDS) protocol for some set S, the client obtains server's secret if and only if his own input to the protocol was "valid", i.e., belonged to S. In particular, consistent CDS provides a solution to the private inference control problem [WS04]; there, consistency is precisely the correct security notion and one is not interested in simulatability. In [AIR01,LL07], the authors proposed CDS protocols based on homomorphic cryptography. In their protocols, the server cryptocomputes some value based on fixed, problem-dependent, tree. As in the general case, one can transform their protocol to a consistent adaptive one by letting the server, instead of the original secret, to cryptocompute its certificate. This direct solution is more efficient than applying the general solution of Sect. 4.

Since the server can restore encrypted inputs of clients, consistent CDS protocols can be used in pay-per-view systems: one client's input corresponds to an encryption of $credit$ stored an updated by the server. It is straightforward to test that $credit > 0$ by using consistent CDS based on [LL07]. Another application is restricted access to private data, e.g., TV or military broadcasts with complex access policy, based on credentials that are represented as random keys.

6 Different Subclasses of Consistent Protocols

In a consistent protocol, an honest client can detect and prove that the server's outputs are not consistent. On the other hand, a malicious server may introduce selective protocol failures so that issuing a complaint leaks information about client's inputs. The definition of consistency limits the corresponding information gain to polynomial-time randomized predicates, i.e., the server gets to know if $\pi(q_1, \ldots, q_j) = 1$ for some polynomial-time randomized predicate π. In practice one might want to further restrict the set of enforceable predicates π. Next we will study this issue a bit more closely.
MEMORYLESS CONSISTENCY. An adaptive protocol is *memoryless-consistent* if the halting predicates $\pi_1, \ldots, \pi_m$ are independent from previous queries, i.e., $\pi_i(q_1, \ldots, q_i) = \pi_i(q_i)$. As a result, the server cannot relate results of different queries. As we show next, Ada is memoryless-consistent when the subsequent executions of the underlying protocol do not share random variables. (We only state the results for white-box reductions from now on.)

Theorem 5 (Memoryless consistency). *A semisimulatable protocol* Ada *is computationally memoryless-consistent, if* gc *is computationally binding, and the* m *instantiations of the underlying protocol do not share random variables.*

Proof. Assume that an adversary A breaks the memoryless-consistent property of Ada. That is, it can force the client to abort if and only if a predicate p_i holds on client's queries $(q_1, \ldots, q_i)$ thus far, where p_i is a non-trivial function of at least two different values q_a and q_b for $a < b \leq i$. Since the protocol is stateless then the adversary can play the role of the client in round $b > a$, to breach the privacy of the client in round a: given its knowledge of whether the client aborted in round b, it will have some advantage in guessing q_a, given the value $p_i(q_a, q_b)$. □

Note that there is no black-box construction from an arbitrary stateful semisimulatable 1-out-of-n f-protocol Π to a memoryless-consistent m-out-of-n protocol for f. A formal separation is following. Let Π be a CPA-secure public-key cryptosystem and assume that the client and the server use the same public key pk in all m rounds. Let E be the corresponding encryption algorithm. Now, if client augments all his queries with $E(q_j)$

and accepts the reply only if the server adds an encryption of q_j to her reply, then the corresponding protocol is certainly semisimulatable. However, if a malicious server replaces $E(q_j)$ with the encryption $E(q_{j-1})$ from the previous round then she can force an halting predicate $[q_{j-1} = q_j]$. More natural examples can be based on the semisimulatable homomorphic oblivious transfer protocol from [AIR01], where the client's message is just $E(q_j)$. If we construct the adaptive protocol so that all invocations of Π share the same pk then the server can force many affine halting predicates.

What is the practical relevance of this? In the theory of cryptography, protocols are required to be stateless. However, in practice one could want to use stateful protocols. For example, in public-key cryptography based protocols like the oblivious transfer protocols of [AIR01,Lip05], the parties may want to use the same client's public key in several instances — in fact, this will result in a private m-out-of-n adaptive protocol. Thus, combining such a protocol with a DAG-commitment scheme does *not* result in a memoryless-consistent adaptive protocol. On the other hand, if one uses a different public key in every round then the resulting adaptive protocol is also memoryless-consistent.

HOMOMORPHIC CONSISTENCY. The PrivateBP protocol [IP07,Lip08] but also many efficient oblivious transfer protocols [Ste98,AIR01,Lip05,LL07] are based on homomorphic cryptography. In such protocols, client encrypts his inputs by using a homomorphic cryptosystem, and thus the sender can cryptocompute (i.e., compute on ciphertexts) arbitrary affine combinations of inputs. More formally, let $\mathcal{P}_{\text{aff}}(q)$ consist of all base predicates of type $[\alpha q = \beta]$ and their conjunctions; let $\mathcal{P}_{\text{b-aff}}(q)$ consist of all base predicates of type $\alpha_1 q[1] + \cdots \alpha_k q[k] = \beta$ where $q[k] \ldots q[1]$ is the bit-representation of q, and their conjunctions. A protocol is *homomorphically consistent* if all enforceable halting predicates $\pi_1(q_1), \ldots, \pi_m(q_1, \ldots, q_m)$ belong either to the class $\mathcal{P}_{\text{aff}}(q_1, \ldots, q_j)$ or to the class $\mathcal{P}_{\text{b-aff}}(q_1, \ldots, q_j)$. Such a set of predicates is rather restricted, and also minimal if we use a homomorphic encryption. One can straightforwardly prove that if one uses an "ideal" homomorphic cryptosystem, that does not allow to cryptocompute any other operation on ciphertexts, together with the PrivateBP protocol, then the resulting adaptive protocol is homomorphically consistent.

Extensive work has been done on cryptographic protocols that use the properties of existing homomorphic cryptosystems. There is no evidence that any non-affine predicate can be cryptocomputed in the case of any of well-known homomorphic cryptosystems. Thus, it is reasonable to make the next seemingly novel gap assumption that only affine predicates can be computed-on-ciphertexts.

Assumption 1 (Gap homomorphic assumption for Π.) *Let Π be an additively homomorphic cryptosystem such as the Paillier [Pai99]. For any non-affine efficiently verifiable t-ary predicate π, $t \geq 1$, the probability that some polynomial-size probabilistic circuit A, given access to a randomly generated public key pk and to random encryptions of any t elements q_i, can output an encryption of $\pi(q_1, \ldots, q_t)$, is negligibly different to the probability that A can guess the value of $\pi(q_1, \ldots, q_t)$, given its knowledge of the distribution from which the plaintexts q_i are drawn.*

Alternatively, if Assumption 1 cannot be proved for some concrete cryptosystem, one should still describe the class of computable predicates, as it provides a subset of predicates that can be efficiently cryptocomputed using additively homomorphic encryption. The set of cryptocomputable predicates that can be computed with constant communication is currently unknown and is one of the most interesting problems in the design of efficient cryptographic protocols. For example, one can extend this discussion to any of the protocols that are based on cryptosystems that allow to cryptocompute quadratic polynomials [BGN05] in which case we can define a similar gap assumption.

References

AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag. 1, 2, 4, 5, 6

AL07. Yonatan Aumann and Yehuda Lindell. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In Vadhan [Vad07], pages 137–156. 2

BG92. Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, Santa Barbara, California, USA, August 16–20, 1992. Springer-Verlag, 1993. 3

BGN05. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In Joe Kilian, editor, *The Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer Verlag. 6

BL88. Josh Benaloh and Jerry Leichter. Generalized Secret Sharing and Monotone Functions. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35, Santa Barbara, California, USA, 21–25 August 1988. Springer-Verlag, 1990. 5

BL07. Ahto Buldas and Sven Laur. Knowledge-Binding Commitments with Applications in Time-Stamping. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Theory and Practice of Public-Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 150–165, Beijing, China, April 16–20, 2007. Springer-Verlag. 1, 3, 3, A, A

CC00. Christian Cachin and Jan Camenisch. Optimistic Fair Secure Computation. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 93–111, Santa Barbara, USA, August 20–24 2000. Springer-Verlag. 2

CGKS95. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25 1995. IEEE. 4

CIO98. Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 141–150, New York, May 23–26, 1998. 3

DG03. Ivan Damgård and Jens Groth. Non-Interactive And Reusable Non-Malleable Commitment Schemes. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 426–437, San Diego, CA, USA, June 9–11 2003. ACM Press. 3

Di 02. Giovanni Di Crescenzo. Equivocable And Extractable Commitment Schemes. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, 3rd International Conference, SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 74–87, Amalfi, Italy, September 11–13, 2002. Springer Verlag. 3

DJ01. Ivan Damgård and Mads Jurik. A Generalisation, A Simplification And Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, February 13–15, 2001. Springer-Verlag. 4

FO97. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In Burton S. Kaliski, Jr., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Santa Barbara, USA, 17–21 August 1997. Springer-Verlag. 3

GIKM00. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. *Journal of Computer and System Sciences*, 60(3):592–629, June 2000. 1

GR05. Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In Luis Caires, Guiseppe F. Italiano, Luis Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815, Lisboa, Portugal, 2005. Springer-Verlag. 5

IP07. Yuval Ishai and Anat Paskin. Evaluating Branching Programs on Encrypted Data. In Vadhan [Vad07], pages 575–594. 1, 2, 4, 6

Kal05. Yael Tauman Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In Ronald Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag. 4

KO97. Eyal Kushilevitz and Rafail Ostrovsky. Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 20–22, 1997. IEEE Computer Society. 1, 4

Lip05. Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In Jianying Zhou and Javier Lopez, editors, *The 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328, Singapore, September 20–23, 2005. Springer-Verlag. 1, 2, 4, 4, 5, 6

Lip08. Helger Lipmaa. Private Branching Programs: On Communication-Efficient Cryptocomputing. Technical Report 2008/107, International Association for Cryptologic Research, 2008. Available at http://eprint.iacr.org/2008/107. 1, 2, 4, 6

LL07. Sven Laur and Helger Lipmaa. A New Protocol for Conditional Disclosure of Secrets And Its Applications. In Jonathan Katz and Moti Yung, editors, *5th International Conference on Applied Cryptography and Network Security – ACNS'07*, volume 4521 of *Lecture Notes in Computer Science*, pages 207–225, Zhuhai, China, June 5–8, 2007. Springer-Verlag. 1, 2, 4, 4, 5, 6

NN01. Moni Naor and Kobbi Nissim. Communication Preserving Protocols for Secure Function Evaluation. In *Proceedings of the Thirty-Third Annual ACM Symposium on the Theory of Computing*, pages 590–599, Heraklion, Crete, Greece, July 6–8 2001. ACM Press. 1, 2

NP99a. Moni Naor and Benny Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 245–254, Atlanta, Georgia, USA, May 1–4, 1999. ACM Press. 1, 2

NP99b. Moni Naor and Benny Pinkas. Oblivious Transfer with Adaptive Queries. In Donald Beaver, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 573–590, Santa Barbara, USA, 15–19 August 1999. Springer-Verlag. 1

Pai99. Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer-Verlag. 4, 1

Ped91. Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, California, USA, August 11–15, 1991. Springer-Verlag, 1992. 3

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A Framework for Efficient and Composable Oblivious Transfer. In David Wagner, editor, *Advances in Cryptology — CRYPTO 2008, 28th Annual International Cryptology Conference*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, USA, August 17–21, 2008. Springer-Verlag. 1

Ste98. Julien P. Stern. A New And Efficient All Or Nothing Disclosure of Secrets Protocol. In Kazuo Ohta and Dingyi Pei, editors, *Advances on Cryptology — ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371, Beijing, China, October 18–22, 1998. Springer-Verlag. 1, 4, 6

SYY99. Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC^1. In *40th Annual Symposium on Foundations of Computer Science*, pages 554–567, New York, NY, USA, 17–18 October 1999. IEEE Computer Society. 4

Vad07. Salil Vadhan, editor. *The Fourth Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, Amsterdam, The Netherlands, February 21–24, 2007. Springer Verlag. 6

Weg00. Ingo Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. Monographs on Discrete Mathematics and Applications. Society for Industrial Mathematics, 2000. 4

WS04. David P. Woodruff and Jessica Staddon. Private Inference Control. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 188–197, Washington, DC, USA, October 25–29, 2004. 1, 5

A Proof of Thm. 1

Proof. This proof is similar to the recent proof of Buldas and Laur [BL07] that every binding DAG-commitment scheme is also DAG-extractable. We add an explicit treatment of history and allow the adversary output more than one certificate.

We give the proof in the exact setting. Fix k and any $t_k > 0$, $\varepsilon'_k > 0$. Assume that the list size is upper-bounded by some $N = \mathrm{poly}(k)$. Assume that gc is a $(\tau^b_k, \varepsilon^b_k)$-binding DAG-commitment scheme for $\tau^b_k = (1+o(1))(N/\varepsilon'_k+1)t_k$ and for some ε^b_k. Assume that $A = (A_1, A_2)$ is a t_k-time adversary, and denote by ω_i the random tape of A_i. Here $A_1(\mathsf{hist}, \mathsf{ck}; \omega_1)$ outputs a commitment c, n and a state η, and $A_2(\eta, \mathsf{advice}; \omega_2)$ outputs a set of valid certificates $\mathsf{ct} = \{\mathsf{ct}_1, \dots, \mathsf{ct}_m\}$ such that, for any i, $\mathsf{gc.open}_{\mathsf{ck}}(c, \mathsf{ct}_i) \neq \perp$.

Let $\tau^e_k = (1+o(1))(N/\varepsilon'_k+1)t_k = (1+o(1))\tau^b_k$. To simplify notation, let us write $(i, \hat{x}) \in x$ if and only if $\hat{x} = x_i$. We show that there exists a τ^e_k-time knowledge extractor $\mathcal{K}_{A_k}$ so that for any advice: if $\mathsf{ct} \neq \emptyset$ then $\mathcal{K}_{A_k}(\mathsf{hist}, \mathsf{ck}, \omega_1)$ produces a database $\hat{x} = (\hat{x}_1, \dots, \hat{x}_n)$ of candidate sink labels such that for any i, $\mathsf{gc.open}_{\mathsf{ck}}(c, \mathsf{ct}_i) \notin \hat{x}$ with a negligible probability.

For a fixed A, let $W = (w_{ij})$ be the matrix defined as follows. Its rows are indexed by $i = (\mathsf{ck}, \omega_1)$, and its columns are indexed by $j = (\mathsf{hist}, \mathsf{advice}, \omega_2)$. For such i, j, we set $w_{ij} = q$ where q is the list of locations of the revealed elements $\mathsf{ct} \leftarrow A_2(\eta, \mathsf{advice}; \omega_2)$. If $\mathsf{ct} = \emptyset$ then $q = \emptyset$. By Lem. 1, there exists a columnset $\mathcal{I}$ with $|\mathcal{I}| \le N/\varepsilon'_k$, such that for any j, $\Pr_i[\emptyset \ne w_{ij} \not\subseteq \mathcal{L}_i \wedge |\mathcal{L}_i| < N] \le \varepsilon'_k$, where $\mathcal{L}_i := \bigcup_{j\in\mathcal{I}} w_{ij}$ is the set of locations "revealed" by $\mathcal{I}$. Given such an $\mathcal{I}$, consider the next knowledge extractor $\mathcal{K}_{A_k} = \mathcal{K}_{A_k,\mathcal{I}}$:

1. Given $(\mathsf{hist}, \mathsf{ck}, \omega_1)$, store $(c, n, \eta) \leftarrow A_1(\mathsf{hist}, \mathsf{ck}; \omega_1)$ and set $\hat{x}_i = \perp$ for all $i \in \{1, \ldots, n\}$.
2. For each $(q, \omega_2) \in \mathcal{I}$ do:
   - Compute $\mathsf{ct} \leftarrow A_2(\eta, \mathsf{advice}; \omega_2)$.
   - For all $j \in \{1, \ldots, m\}$, compute $(q_j, x_j) \leftarrow \mathsf{gc.open}_{\mathsf{ck}}(c, \mathsf{ct}_j)$ and set $\hat{x}_{q_j} \leftarrow x_j$.
3. Return the last snapshot of $\hat{x}$.

Fix $(\mathsf{hist}, \mathsf{advice}, \omega_2)$. There are now two possibilities for $\mathcal{K}_{A_k}$ to fail. Let $(c, n, \eta) \leftarrow A_1(\mathsf{hist}, \mathsf{ck}; \omega_1)$ and $\mathsf{ct} \leftarrow A_2(\eta, \mathsf{advice}; \omega_2)$ as before. For a revealed element $(q_j, x_j) \leftarrow \mathsf{gc.open}_{\mathsf{ck}}(c, \mathsf{ct}_j)$ we can have $\hat{x}_j = \perp$ only if $|\mathcal{L}_{(\mathsf{ck},\omega_1)}| < N$. But by the construction of $\mathcal{I}$, the probability of the latter is not larger than ε'_k. The second possibility is that $x_j \notin \hat{x}$, i.e., $x_j \ne \hat{x}_{q_j} \ne \perp$. In this case, we can construct break the binding property of the DAG-commitment scheme. Clearly, $\mathcal{K}_{A_k}$ works in time τ^e_k and extracts x with probability $1 - (\varepsilon^b_k + \varepsilon'_k)$. Therefore, gc is $(t_k, \tau^e_k, \varepsilon^b_k + \varepsilon'_k)$-DAG-extractable and thus also DAG-extractable because $\tau^e_k = (1 + o(1))(N/\varepsilon'_k + 1)t_k = \mathrm{poly}(k)t_k/((\varepsilon^b_k + \varepsilon'_k) - \varepsilon^b_k)$. □

To prove Thm. 1, we are now left to prove the next technical result. It generalizes Lemma 1 from [BL07] to the case $m > 1$.

Lemma 1. *Consider a finite matrix $W = w_{ij}$, the rows of which are indexed by $i \in \mathcal{R}$, columns are indexed by $j \in \mathcal{C}$, and its entries are sets of integers. Assume that a certain probability measure is defined over the row indexes $\mathcal{R}$. For any $\delta > 0$ and $N \in \mathbb{N}$, there exists a set of column indexes $\emptyset \subseteq \mathcal{I} \subseteq \mathcal{C}$ such that $0 \le |\mathcal{I}| \le N/\delta$ and for every column $j \in \mathcal{C}$,*

$$\Pr_i[\emptyset \ne w_{ij} \not\subseteq \mathcal{L}_i \wedge |\mathcal{L}_i| < N] \le \delta \ ,$$

where $\mathcal{L}_i = \bigcup_{j\in\mathcal{I}} w_{ij} \setminus \{\emptyset\}$ is the set of nonzero elements "revealed" by $\mathcal{I}$.

Proof. Consider the next iterative procedure:

```
1. Set $\mathcal{I} \leftarrow \emptyset$ and initialize $\mathsf{ctr}_i \leftarrow N$ for $i \in \mathcal{R}$.
2. While exists $j \in \mathcal{C}$ such that $\Pr[i : w_{ij} \ne \emptyset] \ge \delta$ do
   (a) Choose a $j$ such that $\Pr[i : w_{ij} \ne \emptyset] \ge \delta$; insert $j$ into $\mathcal{I}$
   (b) For each row $i \in \mathcal{R}$ such that $w_{ij} \ne \emptyset$ do
       i. Set $w = \{w_1, \ldots, w_m\} \leftarrow w_{ij}$
       ii. For each $j' \in \mathcal{C}$, $t \in [m]$: If $w_t \in w_{ij'}$ then $w_{ij'} \leftarrow w_{ij'} \setminus \{w_t\}$
       iii. Set $\mathsf{ctr}_i \leftarrow \mathsf{ctr}_i - 1$
   (c) For all $j' \in \mathcal{C}$: if $\mathsf{ctr}_i = 0$ then $w_{ic'} \leftarrow \emptyset$
```

Let $\mathcal{N} = \{i : \exists w_{ij} \ne \emptyset\}$ denote the set of nonzero rows, and let $\mathcal{N}_{old}$, $\mathcal{N}_{new}$ denote the value of $\mathcal{N}$ before and after some update at Step 2. Let $\mathsf{mean}[\mathcal{N}] = \sum_{i\in\mathcal{N}} \mathsf{ctr}_i \cdot \Pr[i]$ be the average counter value. Then by the construction $\mathsf{mean}[\mathcal{N}_{new}] \le \mathsf{mean}[\mathcal{N}_{old}] - \delta$ after a single iteration of Step 2. As initially $\mathsf{mean}[\mathcal{N}] \le \mathcal{N}$, then after $\lfloor N/\delta \rfloor$ iterations $\Pr[\mathcal{N}] \le \mathsf{mean}[\mathcal{N}] < \delta$. Note that the algorithm nullifies the elements $w_{ij'}$ only if they already belong to $\mathcal{L}_r$ or $|\mathcal{L}_r| \ge N$. In the end, each column j contains at most a δ-fraction of elements that satisfy the predicate $w_{ij} \ne \emptyset \wedge w_{ij} \notin \mathcal{L}_r \wedge |\mathcal{L}_r| < N$ and the claim follows. (Note that $\mathcal{I}$ can be empty.) □

See Tbl. 1 for an example.

Table 1. An example, illustrating Lem. 1, with $\delta = 0.25$ and $N = 2$

$\mathcal{I} = \emptyset$					$\mathcal{L}$	ctr
$\{1,2\}$	$\{2,3\}$	$\emptyset$	$\{1,4\}$	$\{1,2\}$	$\emptyset$	2
$\emptyset$	$\emptyset$	$\emptyset$	$\{2,3\}$	$\emptyset$	$\emptyset$	2
$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\emptyset$	$\emptyset$	$\emptyset$	2
$\emptyset$	$\{2,3\}$	$\emptyset$	$\{1,3\}$	$\{1,4\}$	$\emptyset$	2
$\{1,2\}$	$\{3,4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2

$\mathcal{I} = \{1\}$					$\mathcal{L}$	ctr
$\emptyset$	$\{\mathbf{3}\}$	$\emptyset$	$\{4\}$	$\emptyset$	$\{1,2\}$	1
$\emptyset$	$\emptyset$	$\emptyset$	$\{2,3\}$	$\emptyset$	$\emptyset$	2
$\emptyset$	$\{\mathbf{3}\}$	$\{1,3\}$	$\emptyset$	$\emptyset$	$\{2,4\}$	1
$\emptyset$	$\{2,3\}$	$\emptyset$	$\{1,3\}$	$\{1,4\}$	$\emptyset$	2
$\emptyset$	$\{3,4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{1,2\}$	1

$\mathcal{I} = \{1,2\}$					$\mathcal{L}$	ctr
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{1,2,3\}$	0
$\emptyset$	$\emptyset$	$\emptyset$	$\{2,3\}$	$\emptyset$	$\emptyset$	2
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{2,3,4\}$	0
$\emptyset$	$\emptyset$	$\emptyset$	$\{\mathbf{1}\}$	$\{1,4\}$	$\{2,3\}$	2
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{1,2,3,4\}$	0

$\mathcal{I} = \{1,2,4\}$					$\mathcal{L}$	ctr
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{1,2,3\}$	0
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{2,3\}$	1
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{2,3,4\}$	0
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{4\}$	$\{1,2,3\}$	1
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{1,2,3,4\}$	0

B Proof of Theorem 3

Proof. Follows from Lemmas 2 and 3. □

Lemma 2 (Honest Client). *If* gc *is binding and* $\mathit{\Gamma}$ *is computationally client-private, then* Ada *is consistent.*

Proof. If the client is honest, we must consider the joint output distribution in the ideal and real world. Let A^{real} be the real-world adversary, and let hist be any history. Construct the next ideal-world adversary $A^{ideal}(\mathsf{hist}, \mathsf{ck}, q_1, \ldots, q_m)$: (1) Run the setup phase to obtain public parameters for gc and $\mathit{\Gamma}$. (2) Run the commitment phase with $A^{real}(\mathsf{hist})$ and output a commitment c. (3) Cryptocompute the branching program between $A^{real}(\mathsf{hist})$ and the honest client C with inputs $q_1, \ldots, q_m$. (4) Output the corresponding certificates $\mathsf{ct}_{q_1}, \ldots, \mathsf{ct}_{q_m}$ and the final state of A^{real}.

Define $\mathsf{advice} := (q_1, \ldots, q_m)$. Because gc is binding then it is also DAG-extractable. More precisely, assume that gc is $(t_k, \tau_k^b, \varepsilon_k^b)$-DAG-extractable, where t_k is the running time of $A^{ideal}(\mathsf{hist}, \mathsf{ck}, \mathsf{advice})$. Thus, there exists a τ_k^e-time knowledge extractor $\mathcal{K}_{A^{ideal}}$, that given $(\mathsf{ck}, \mathsf{hist})$, outputs a prediction $\hat{x}$ such that the probability ε_k^e that $A^{ideal}(\mathsf{ck}, \mathsf{hist}, \mathsf{advice})$ can produce a valid certificate, that leads to an element not in $\hat{x}$, is negligible. Here, by definition, $\tau_k^e = p(t_k)/(\varepsilon_k^e - \varepsilon_k^b)$ for some non-negative polynomial p. Thus, we can construct the next ideal-world adversary $\widehat{\mathsf{Sim}}$:

1. She runs the setup phase to obtain the public key ck and the randomness ω for A^{ideal}.
2. She computes $\hat{x} \leftarrow \mathcal{K}_{A^{ideal}}(\mathsf{ck}, \mathsf{hist}; \omega)$ and sends $\hat{x}$ to TTP.
3. She sends the description of $A^{ideal}(\mathsf{ck}, \mathsf{hist}, \mathsf{advice}; \omega)$ to Halt-Machine that evaluates $A^{ideal}(\mathsf{ck}, \mathsf{hist}, \mathsf{advice}; \omega)$ and stops TTP on the jth round if ct_{q_j} is not valid. (She can find out $(q_1, \ldots, q_m)$ by rewinding the client m times, and every time observing what is the client's $(j+1)$th query when Halt-Machine $= A^{ideal}(\mathsf{ck}, \mathsf{hist}, q_1, \ldots, q_j, 0, \ldots, 0; \omega)$.)
4. She runs $A^{ideal}(\mathsf{ck}, \mathsf{hist}, 1, \ldots, 1; \omega)$ and outputs the final state of A^{real}.

Clearly, also $\widehat{\mathsf{Sim}}$ works in time $\mathrm{poly}(t_k)/(\varepsilon_k^e - \varepsilon_k^b)$. By the construction the honest client receives $\perp$ in the ideal world whenever the client receives $\perp$ also in the real world w.r.t. the randomness ω. Otherwise, the client's outputs are different only if the prediction $\hat{x}_{q_j} \neq x_{q_j}$ and such event is negligible by the construction. Since $\mathit{\Gamma}$ is computationally client-private then the outputs of $\widehat{\mathsf{Sim}}$ and A^{real} are indistinguishable. The claim follows. □

Lemma 3 (Honest Server). *If* gc *is equivocable, then* Ada *is server-private.*

Proof. Assume that the server is honest. Then the output of the server in the ideal and real model is $\perp$ and we can consider only the output of a malicious client. Let World_0 be the ideal world, let World_1 be the real world.

To demonstrate the closeness of World_1 and World_0, let us convert a malicious client $\hat{C}_1$ from World_1 to a malicious client $\hat{C}_0$ for World_0 as follows:

1. Generate the equivocability key $(\mathsf{ek}, \mathsf{ck}) \leftarrow \mathsf{gc}.\widehat{\mathsf{gen}}(1^k)$ for the DAG-commitment scheme.
2. Compute $(\hat{c}, \eta) \leftarrow \mathsf{gc}.\widehat{\mathsf{com}}_{\mathsf{ck}}(\mathsf{ek}, n)$ and send $\hat{c}$ to the adversary $\hat{C}_1$.
3. Given $\hat{C}_1$'s query q_j, obtain x_{q_j} from TTP and compute $\hat{\mathsf{ct}}_j \leftarrow \mathsf{gc}.\mathsf{equiv}_{\mathsf{ek}}(\hat{c}, \eta, q_j, x_{q_j})$.
4. Return the output of the adversary $\hat{C}_1$.

Note that in both worlds $\hat{C}_1$ plays the equivocability game even if $\hat{C}_1$ uses history in the attack. In world World_1, the adversary $\hat{C}_1$ interacts with the oracle $\mathcal{O}$ and in World_0 it interacts with oracle $\hat{\mathcal{O}}$. Since the commitment scheme is equivocable, the outputs of $\hat{C}_1$ and $\hat{C}_0$ must be indistinguishable. Also, the computational overhead of $\hat{C}_0$ is only polynomial and thus the transformation is plausible. $\square$