On the Feasibility of Consistent Computations

Sven Laur[1] and Helger Lipmaa[2,3]

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

Abstract. In many practical settings, participants are willing to deviate from the protocol only if they remain undetected. Aumann and Lindell introduced a concept of covert adversaries to formalize this type of corruption. In the current paper, we refine their model to get stronger security guarantees. Namely, we show how to construct protocols, where malicious participants cannot learn anything beyond their intended outputs and honest participants can detect malicious behavior that alters their outputs. As this construction does not protect honest parties from selective protocol failures, a valid corruption complaint can leak a single bit of information about the inputs of honest parties. Importantly, it is often up to the honest party to decide whether to complain or not. This potential leakage is often compensated by gains in efficiency—many standard zero-knowledge proof steps can be omitted. As a concrete practical contribution, we show how to implement consistent versions of several important cryptographic protocols such as oblivious transfer, conditional disclosure of secrets and private inference control.
Keywords. Consistency, equivocal and extractable commitment, oblivious transfer, private inference control.

1 Introduction

Although classical results assure the existence of secure two- and multi-party protocols for any functionality in the presence of malicious adversaries, the computational overhead is often prohibitively large in practice. Hence, cryptographers have sought more restricted models for malicious behavior, which are still realistic but facilitate more efficient protocol construction. A model of covert adversaries [2] proposed by Aumann and Lindell considers a setting, where corrupted parties are unwilling to deviate from the protocol unless they remain uncaught. More precisely, they defined a hierarchy of security models, where malicious behavior that alters the outputs of honest parties is detectable with high probability. However, none of these models guarantee input-privacy because a malicious adversary might potentially issue a detectable attack that completely reveals inputs of all honest parties. We extend their hierarchy with a new security model (*consistent computing*), which guarantees that malicious participants cannot learn anything beyond their intended outputs and honest participants can detect malicious behavior that alters their outputs. As a result, a valid corruption complaint leaks only *a single bit* of information about the inputs of honest

Objective	Input-privacy	Output-privacy	Complaint handling	Detectability
Multi-party protocols				
Security	Yes	Yes	Secure	Optional
Consistency	Limited leaks	Limited leaks	Possible	Optional
K-leakage	Limited leaks	Limited leaks	Possible	No
Covert Model	No	No	Impossible	Partial
Privacy	No	No	Impossible	No
Two-party protocols				
Security	Yes	Yes	Secure	Yes
Consistency	Yes	Yes	Possible	Yes
K-leakage	Limited leaks	Limited leaks	Possible	No
Covert Model	No	No	Impossible	Yes
Privacy	No	No	Impossible	No

Table 1. Comparison of various security objectives in a malicious model

parties as opposed to *the complete disclosure*. Moreover, an honest participant can often decide whether to complain or not. If a complaint is not filed, then no information will be leaked at all unless the adversary learns it indirectly.

Our security model also guarantees that no participant can change their input during a multi-round protocol, which consists of many sub-protocols, i.e., there exists an input that is consistent with all outputs. Additionally, the client can prove cheating to third parties without active participation from the server, since the protocol failure together with a proof that shows correctness of client's actions is sufficient. Hence, our security model is sufficient for many client-server applications, where a server's long-term reputation is more valuable than information revealed by corruption complaints.

Finally, note that the ability to detect cheating from legitimate protocol failures can be important, as well. A good example is private inference control [31], where the client makes queries to the server's database. To protect the server's privacy, certain query patterns are known to be forbidden and should be rejected, though without the server necessarily getting to know which one of the "forbidden" query patterns was used. Hence, a client really needs to know whether the query failed due to insufficient privileges or the server just cheated.

Our contributions. Our main contribution is the new security model, which provides more strict security guarantees than the semihonest model, all flavors of covert models [2], and the k-leakage model [23] as depicted in Table 1.

We also present concrete, efficient protocols for consistent adaptive oblivious transfer and consistent conditional disclosure of secrets. Notably, all our constructions are much more efficient than their fully secure counterparts. For instance, the new consistent oblivious transfer protocol is secure against unbounded malicious clients, uses 2 messages per query, and has communication and computation comparable to that of the underlying private oblivious trans-

fer protocol. As a main technical tool, we use list commitment schemes, which allow to commit to a list of elements so that, given a short certificate, one can later verify the value of a single element of the committed list. Besides conventional hiding and binding properties, we need equivocality and extractability. See Sect. 3 for details and constructions.

Notation. Throughout this paper, k denotes the security parameter, $\{\mathcal{A}_k\}$ is a shorthand for a non-uniform adversary. The shorthand $t(k) \in \text{poly}(k)$ denotes that $t(k)$ can be bounded by a polynomial and $\varepsilon(k) \in \text{negl}(k)$ means that $\varepsilon(k)$ decreases asymptotically faster than any reciprocal of a polynomial.

Full Version and History. Full version of this paper can be found at [20]. The first version of this eprint from the March of 2006 already defines consistency (although under a different name). The 2-message argument system from [14] was influenced by the first version of the eprint.

2 Definition of Consistent Computations

Achieving security against malicious behavior usually involves a large computational overhead, since one must provide a universal fraud detection mechanism such that honest parties can detect a fraud even if it does not affect their concrete private outputs. As a possible trade-off between efficiency and security, we could protect honest parties only against such actions that alter their outputs. As a result, malicious adversaries might still cause selective protocol failures, where honest parties fail if their inputs are in a specific range. In the following, we use the standard ideal versus real world paradigm to formalize this concept of consistent computations for various protocols. Note that we use standard security definitions [8, 18] with modified ideal world implementations, which give additional power to the adversary, see Fig. 1 as an example.

For clarity and brevity, we present the definitions without delving into subtle technical issues. In particular, we have omitted all low-level details of the ideal and real world executions, as these are thoroughly discussed in common reference materials [8, 18]. Other more model specific details are separately discussed at the end of the section.

Idealized implementations. In an idealized two-party protocol corresponding to consistent computing, both parties send their inputs x_1, x_2 to the trusted third party TTP, which computes the corresponding outputs y_1, y_2. Next, a corrupted participant sends the description of a randomized halting predicate $\pi(\cdot)$ to TTP, who internally computes $\pi(x_1, x_2)$. If $\pi(x_1, x_2) = 1$, then TTP halts the computations and sends $\perp$ to the honest participant. If $\pi(x_1, x_2) = 0$, then TTP sends back the outputs y_i exactly the same way as in the standard ideal model. In particular, the corrupted party can still cause a premature abortion and thus still learn its output.

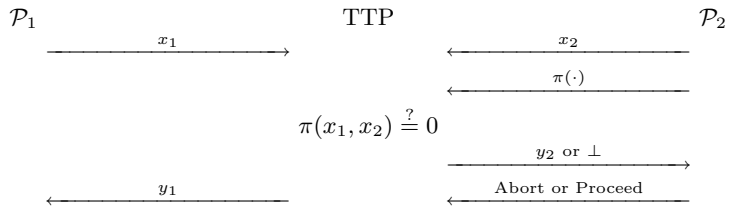

Fig. 1. Ideal world model for consistent two-party computations. A corrupted participant $\mathcal{P}_2$ can cause selective halting by specifying a predicate $\pi(\cdot)$. In the standard model, the dominant party $\mathcal{P}_2$ can cause only a premature abortion

Generalization to the multi-party setting is straightforward. However, there are two subtle issues connected with *fairness* and *detectability*. A protocol guarantees *fair selective abortion* if an adversary can specify only a single predicate $\pi(\cdot)$ such that TTP halts the computations and sends $\perp$ to all participants iff $\pi(x_1, \ldots, x_n) = 1$. Alternatively, corrupted participants can separately specify different halting predicates $\pi_i(\cdot)$ for each party $\mathcal{P}_i$ and thus some parties might get their outputs while others do not. Also, note that the identity of the malicious coalition might remain hidden for multi-party protocols, whereas this cannot happen in a two-party protocol. A consistent protocol provides *detectability* if TTP sends $\perp$ to $\mathcal{P}_i$ together with the identity of a corrupted participant who specified the halting predicate whenever $\pi_i(x_1, \ldots, x_n) = 1$.

Consistency can also be formalized for adaptive computations, where the outputs of each round can depend on the inputs submitted in previous rounds. For the sake of brevity, we define consistency only for client-server protocols, where the server initially commits to his or her input, and after that the client can issue various oblivious queries. This model covers many practical settings such as selling digital goods and private inference control [1, 31]. To start such a protocol, a server sends his or her input x to TTP. After that the client(s) can adaptively issue various queries q_i to TTP. When a query q_i arrives, TTP sends a notification message to the server who can then specify a description of a halting predicate $\pi_i(x_1, \ldots, x_i)$. Next, TTP evaluates the predicate and sends $f(q_i, x)$ back to the client if $\pi_i(q_1, \ldots, q_i) = 0$, otherwise the client receives $\perp$. As a small subtlety, note that the ability to issue halting predicates one-by-one is needed only in the adaptive corruption model, where there are many clients and the server might become corrupted in the middle of computations.

Formal security definition. As usual, we define security of a protocol by comparing its output distribution in the standalone setting with the corresponding output distribution in the ideal world. More formally, fix a security parameter k and let $\mathfrak{D}_k$ denote the input distribution of all parties including the adversary $\mathcal{A}_k$. W.l.o.g. we assume that each input is a pair (ϕ_i, x_i), where the auxiliary input ϕ_i models the internal state of the participant before the protocol and x_i is the actual protocol input. Now, if we fix the exact details how protocols are executed

and what the plausible attacks are, then a protocol instance Π_k and an adversary $\mathcal{A}_k$ together determine uniquely a joint output distribution $\textsc{Real}_{\mathfrak{D}_k}(\mathcal{A}_k, \Pi_k)$ of all parties including $\mathcal{A}_k$. Let $\textsc{Ideal}_{\mathfrak{D}_k}(\mathcal{A}_k^\circ, \Pi_k^\circ)$ denote the joint output distribution determined by the ideal world adversary and the corresponding ideal world implementation Π_k°. We say that the protocol family $\{\Pi_k\}$ *securely implements* $\{\Pi_k^\circ\}$ if for any non-uniform polynomial-time adversary $\{\mathcal{A}_k\}$ there exists a non-uniform polynomial-time adversary $\{\mathcal{A}_k^\circ\}$ such that for any input distribution family $\{\mathfrak{D}_k\}$, the output distributions $\textsc{Real}_{\mathfrak{D}_k}(\mathcal{A}_k, \Pi_k)$ and $\textsc{Ideal}_{\mathfrak{D}_k}(\mathcal{A}_k^\circ, \Pi_k^\circ)$ are computationally indistinguishable. If the output distributions are statistically indistinguishable or coincide, then we can talk about statistical and perfect security. Finally, a protocol family $\{\Pi_k\}$ *correctly implements* $\{\Pi_k^\circ\}$ if for any input distribution family $\{\mathfrak{D}_k\}$ the output distributions coincide provided that the adversaries remain inactive (corrupt nobody) in both worlds.

Basic properties. Note that the only difference between the formal definitions of consistency and security in the malicious model is in the description of the ideal world execution. Hence, we can treat a consistent protocol as a secure implementation of a modified functionality that allows explicit specification of halting predicates. As a result, standard composability results carry over and each consistent protocol in a sequential composition can be replaced with the ideal implementation. However, the resulting hybrid protocol does not necessarily correspond to a consistent ideal world execution. For instance, if we execute two client-server protocols in a row, then the server's input is not guaranteed to be the same for both ideal implementations. Also, a malicious server can specify two halting predicates instead of a single one.

The main advantage of consistent computations over other weakened security models is an explicit correctness guarantee. By the construction of the idealized model of consistent computations, an honest participant reaches the accepting state iff his or her output is consistent with the inputs submitted in the beginning of the protocol. Hence, a successful protocol run provides consistency guarantees in the real world, as well. Consequently, a non-accepting honest participant can prove without the help of other participants that a malicious attack was carried out. Moreover, any consistent protocol can be augmented with a complaint handling mechanism that reveals nothing beyond the validity of the complaint.

Theorem 1. *Let a protocol family $\{\Pi_k\}$ be a correct and consistent implementation of a functionality $\{\Pi_k^\circ\}$ such that all messages are signed by their creators. Then an honest participant can prove the existence of a malicious attack that alters his or her output without help from others provided that the signature scheme is secure. This proof can be converted to a zero-knowledge proof if the messages received by the honest participant reveal nothing about his or her input.*

Proof (Sketch). For the proof, note that by our security assumptions no participant can forge messages sent by others. Hence, if an honest party reveals his or her input and randomness together with all received messages, then anybody can verify correctness of her computations. Since the protocol implements correctly

$\{\Pi_k^\circ\}$, semi-honestly behaving participants cannot cause a non-accepting output. This proof can be converted to a zero-knowledge proof, since it is sufficient to present all received messages and then prove that there exists a valid input and randomness that leads to the non-accepting state. The corresponding statement belongs to an **NP**-language and thus has an efficient zero-knowledge proof. The claim follows as messages in the proof reveal nothing about his or her input. □

Note that the last assumption in Theorem 1 is not a real restriction and can be easily met by using a secure public key cryptosystem. Namely, if all messages are encrypted with public keys of corresponding recipients, then messages leak no information to outside observers but the protocol remains consistent.

However, differently from secure computations, a valid complaint reveals additional information, namely, the adversary learns that the corresponding halting predicate $\pi_i(x_1, \ldots, x_n)$ holds. On the other hand, a honest party does not have to issue a complaint and thus the adversary is *not guaranteed* to learn halting predicates—in some applications, the honest parties can untraceably recover from protocol failures. For all consistent and detectable protocols, such a complaint also reveals the identity of the maliciously behaving participant. Hence, there is a trade-off between the utility of a single bit $\pi_i(x_1, \ldots, x_n)$ and a long-term reputation of a participant. As a result, consistency of computations is an adequate protection mechanism for all settings, where participants are unwilling to cheat if they are caught with high probability or a single bit leakage is much smaller compared to the amount of information revealed by legitimate protocol outputs. For instance, the intended output of privacy-preserving data-aggregation is usually several kilobytes (if not megabytes) long, and therefore the effect of a single bit leakage is likely to be irrelevant.

The same argumentation holds also for consistent protocols without accountability. However, finding the identity of the culprit is difficult in such settings, because everybody must prove the correctness of their actions and the corresponding zero-knowledge proofs can be intractable in practice. Finally, note that the potential damage of a valid complaint depends on the set of possible halting predicates π_i. In Section 6, we study this question explicitly and show how to restrict the class of enforceable predicates.

Relation to other security definitions. The concept of covert corruption is rather old and has been discussed in many contexts. The earliest definitions were given for the multi-party setting [15, 9] and only recently modified to work in two-party settings by Aumann and Lindell [2]. In particular, note that the definition of *t-detectability* given in [15] and various definitions of *ε-detectability* given in [2] guarantee only that malicious behavior, which alters the outputs of honest parties, is detected with notable probability. However, none of the definitions limit the amount of information acquired during a successful fraud attempt. Thus, our definition of consistent computations is a natural *strengthening* of these definitions, which also guarantees the privacy of inputs. Another related security notion is the *k-leakage model* [23], where the adversary can learn up to k bits of auxiliary information about the inputs of honest parties. Similarly

to consistent computations, the adversary cannot alter outputs without being detected. However, differently from consistent computations, the information is *guaranteed* to reach adversary and such an attack is undetectable. Hence, the k-leakage model provides less strict security guarantees. See Table 1 for a comprehensive summary.

Subtle details. Note that halting predicates must be efficiently computable. Otherwise, participation in an idealized computation can provide significant gains to the adversary. Hence, we require that for any adversary $\{\mathcal{A}_k\}$, the time needed to evaluate halting predicates is polynomial in the running-time of $\{\mathcal{A}_k\}$.

Also, observe that many important cryptographic protocols are not secure in the strict sense. The problem starts with classical zero-knowledge proofs, for which, we know only how to construct simulators $\mathcal{A}^\circ$ that work in expected polynomial time. However, a model where ideal world adversaries have expected polynomial running time causes many technical and philosophical drawbacks [19]. For instance, we loose sequential composability guarantees. Hence, we use an alternative formalization. A protocol $\{\Pi_k\}$ is secure in a *weak polynomial security model*, if for any time bound $t(k) \in \mathrm{poly}(k)$, for any notable difference $\varepsilon(k) \in \Omega(k^{-c})$, and for any polynomial-time real world adversary $\{\mathcal{A}_k\}$, there exists an polynomial-time ideal world adversary $\{\mathcal{A}_k^\circ\}$ such that no non-uniform distinguisher $\{\mathcal{B}_k\}$ with running-time $t(k)$ that can distinguish $\textsc{Real}_{\mathfrak{D}_k}(\mathcal{A}_k, \Pi_k)$ and $\textsc{Ideal}_{\mathfrak{D}_k}(\mathcal{A}_k^\circ, \Pi_k^\circ)$ with advantage more than $\varepsilon(k)$. This definition has the virtue of being formalized with strict time bounds and thus free of technical issues. In particular, it is sequentially composable and formalizes our knowledge about the reductions as precisely as possible.

3 List Commitment Schemes

To achieve consistency, a corrupted participant must be unable to change his or her input during the protocol without being caught. The latter can be achieved by forcing participants to commit to their inputs. More precisely, we need commitment schemes for lists of elements, such that individual elements can later be decommitted by presenting short certificates. A *list commitment scheme* is a quadruple of probabilistic polynomial-time algorithms $(\mathsf{gen}, \mathsf{com}, \mathsf{cert}, \mathsf{open})$ with the following semantics. The key-generator algorithm $\mathsf{gen}(1^k)$ is used to generate public parameters ck that fix the message space $\mathcal{M}_k$ and the maximal number of list elements $N_k \in \mathrm{poly}(k)$. Given a list $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{M}^n$ with $n \le N_k$, the commitment algorithm $\mathsf{com}_{\mathsf{ck}}(\boldsymbol{x})$ outputs a pair (c, d) of commitment and decommitment values. The certificate generation algorithm $\mathsf{cert}_{\mathsf{ck}}(d, i)$ returns a partial decommitment value (*certificate*) s_i for the ith element. The verification algorithm $\mathsf{open}_{\mathsf{ck}}(c, s)$ returns either a pair (i, x_i) or $\perp$. It is required that $\mathsf{open}_{\mathsf{ck}}(c, \mathsf{cert}_{\mathsf{ck}}(d, i)) = (i, x_i)$ for all possible values of $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(\boldsymbol{x})$. We now define various (optional) security properties through games that are played between a challenger and a nonuniform adversary.

Binding and hiding. A list commitment scheme is *computationally binding* if every polynomial-time adversary $\{\mathcal{A}_k\}$ wins the following game with negligible probability:

1. Challenger generates $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and sends ck to $\mathcal{A}_k$.
2. $\mathcal{A}_k$ generates a commitment $\hat{c}$ and two certificates $\widehat{s}_0$ and $\widehat{s}_1$.
3. $\mathcal{A}_k$ wins if the commitment can be opened to different values of x_i. That is, locations coincide $i_0 = i_1$ but $\bot \neq x_0 \neq x_1 \neq \bot$ for the openings $(i_0, x_0) \leftarrow \mathsf{open}_{\mathsf{ck}}(\hat{c}, \widehat{s}_0)$ and $(i_1, x_1) \leftarrow \mathsf{open}_{\mathsf{ck}}(\hat{c}, \widehat{s}_1)$.

A list commitment scheme is *statistically hiding* if for any non-uniform adversary $\mathcal{A}$ the probability that $\mathcal{A}_k$ wins the following game is negligibly close to one half:

1. Challenger generates $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and sends ck to $\mathcal{A}_k$.
2. $\mathcal{A}_k$ sends two lists $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)} \in \mathcal{M}^n$ with $n \leq N_k$ to the challenger.
3. Challenger generates a random bit $b \leftarrow \{0,1\}$, computes $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(\boldsymbol{x}^{(b)})$ and sends the commitment value c to $\mathcal{A}_k$.
4. In the next phase, $\mathcal{A}_k$ can make a number of oracle queries to $\mathsf{cert}_{\mathsf{ck}}(d, \cdot)$ provided that $x_i^{(0)} = x_i^{(1)}$ for any queried index i.
5. $\mathcal{A}_k$ wins the game if he or she correctly guesses the bit b.

Equivocality. In several proofs, we use simulators that send a fake commitment value $\hat{c}$ to the adversary and then gradually open parts of it according to the instructions sent by TTP. To preserve the closeness of real and simulated executions in such a setting, the commitment scheme must be equivocal. A list commitment scheme lc is *perfectly equivocal* if there exist three additional algorithms gen°, com° and equiv, such that no unbounded adversary $\{\mathcal{A}_k\}$ can distinguish between the following two experiments:

Normal Execution:

1. Challenger generates $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$ and sends ck to $\mathcal{A}_k$.
2. $\mathcal{A}_k$ sends $\boldsymbol{x} = (x_1, \ldots, x_n)$ to the oracle $\mathcal{O}$ who computes $(c, d) \leftarrow \mathsf{com}_{\mathsf{ck}}(\boldsymbol{x})$, $s_i \leftarrow \mathsf{cert}_{\mathsf{ck}}(d, i)$ and replies with $(c, s_1, \ldots, s_n)$.

Simulated Execution:

1. Challenger generates $(\mathsf{ek}, \mathsf{ck}) \leftarrow \mathsf{gen}^\circ(1^k)$ and sends ck to $\mathcal{A}_k$.
2. The oracle $\hat{\mathcal{O}}$ computes $(\hat{c}, \eta) \leftarrow \mathsf{com}^\circ_{\mathsf{ek}}(n)$ and, given $\boldsymbol{x} = (x_1, \ldots, x_n)$ from $\mathcal{A}_k$, computes $\hat{s}_i \leftarrow \mathsf{equiv}_{\mathsf{ek}}(\hat{c}, \eta, i, x_i)$ and replies with $(\hat{c}, \widehat{s}_1, \ldots, \widehat{s}_n)$.

One can build non-interactive equivocal commitment schemes based on any one-way functions in the common reference string (CRS) model [12]. In the standard model, 3 messages are needed to implement an equivocal commitment scheme. Thus, all subsequent results that use equivocal commitment schemes require at least 3 messages. However, as the initialization phase can be shared between different runs, the round complexity is not a problem in practice.

Extractability. Many commitment schemes have an explicit extraction mechanism such that a person who possesses some extra information sk can open commitments without decommitment value, see for instance [29, 13]. These commitments are often used in simulator constructions, where one has to extract inputs for committed values. For obvious reasons, such trapdoors do not exist when the final commitment value is shorter than the length of a committed string.

Buldas and Laur showed that if the creator of a commitment gets no additional information besides the commitment parameters ck, then all committed elements are efficiently extractable given white-box access to the committing algorithm and to the randomness used by it. See the definition of *knowledge-binding* and corresponding proofs in [5]. However, in the context of two- and multi-party computations, an adversary always gets additional inputs and thus we must amend the definition. A list commitment scheme is *white-box extractable* if for any polynomial-time adversary $\{\mathcal{A}_k\}$ there exists a polynomial-time extractor machine $\{\mathcal{K}_{A_k}\}$ such that for any input distribution $\mathfrak{D}_k$ and for any family of advice strings $\{a_k\}$ the adversary $\mathcal{A}_k$ can win the following game with negligible probability. The family of advice strings $\{a_k\}$ in the game models unknown future events, which might help the adversary to open the commitments.

1. Challenger generates $\mathsf{ck} \leftarrow \mathsf{gen}(1^k)$, $\phi \leftarrow \mathfrak{D}_k$ and a new random tape ω.
2. $\mathcal{A}_k$ gets ϕ and ck as inputs and ω as the random tape and outputs a list commitment c together with size n, $(c, n) \leftarrow \mathcal{A}_k(\phi, \mathsf{ck}; \omega)$.
3. $\mathcal{K}_{\mathcal{A}_k}$ gets ϕ, ck and ω as inputs and outputs $(\hat{x}_1, \ldots, \hat{x}_n) \leftarrow \mathcal{K}_{\mathcal{A}_k}(\phi, \mathsf{ck}; \omega)$.
4. Given advice a_k, $\mathcal{A}_k$ outputs certificates $(s_1, \ldots, s_m) \leftarrow \mathcal{A}_k(a_k)$.
5. The adversary wins if $\mathcal{A}_k$ outputs 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 $\exists j : \bot \neq (i, x_*) = \mathsf{open}_{\mathsf{ck}}(c, s_j) \wedge x_* \neq \hat{x}_i$.

Currently it is not know how to construct a non-interactive compressing commitment scheme that is provably white-box extractable.[4] However, if we consider interactive commitment schemes, where a sender executes a zero-knowledge proof of knowledge that he or she knows how to open all elements under the list commitment, we can construct such a knowledge extractor by definition. By using suitable zero-knowledge techniques as detailed in [24], the total communication between the receiver and the sender can be made sublinear, although the computational overhead might be too large for practical applications.

As the security of proofs of knowledge is often defined in a weaker model [3], we also relax other definitions to be compatible. A list commitment scheme is *weakly white-box extractable* if for any polynomial-time adversary $\{\mathcal{A}_k\}$ and an error bound $\varepsilon(k)$, there exists a extractor machine $\{\mathcal{K}_{\mathcal{A}_k}\}$ such that, for any input distribution $\mathfrak{D}_k$ and for any family of advice strings $\{a_k\}$, the adversary $\mathcal{A}_k$ wins the extractability game with a probability at most $\varepsilon(k)$ and the running-time of $\mathcal{K}_{\mathcal{A}_k}$ is at most $\mathrm{O}(\mathrm{poly}(k)/\varepsilon(k))$ times slower than $\mathcal{A}_k$.

[4] The results of [5] assure existence of extractors $\mathcal{K}_{\mathcal{A}_K, \mathfrak{D}_k}$ that depend on $\mathfrak{D}_k$.

Double-layered commitments. There are two principally different ways how to construct a list commitment scheme with extractability and equivocality properties. First, one can commit elements individually using ordinary commitment scheme with these properties, such as [13]. As a result, we get strong extractability guarantees but cannot get beyond linear communication complexity. Alternatively, we can first build a double-layered equivocal commitment scheme, and then add extractability by an interactive proof of knowledge. A *double-layered commitment scheme* dlc is specified by a conventional commitment scheme cs and a list commitment scheme lc. The key-generator procedure $\mathsf{dlc.gen}$ runs both key generation procedures and outputs a pair of resulting public parameters $(\mathsf{ck}_1, \mathsf{ck}_2)$. To commit to $\boldsymbol{x} = (x_1, \ldots, x_n)$, one first computes conventional commitments $(c_i, d_i) \leftarrow \mathsf{cs.com}_{\mathsf{ck}_1}(x_i)$ for $i \in \{1, \ldots, n\}$ and then outputs $(c_*, d_*) \leftarrow \mathsf{lc.com}_{\mathsf{ck}_2}(c_1, \ldots, c_n)$. To decommit x_i one has to first decommit c_i by giving $\mathsf{lc.cert}_{\mathsf{ck}_2}(d_*, i)$ and then also reveal d_i so that the receiver could compute $\mathsf{cs.open}_{\mathsf{ck}_1}(c_i, d_i)$. Other operations are defined analogously.

Theorem 2. *Let* lc *be a binding commitment scheme and* cs *be a conventional commitment scheme. Then* dlc *inherits statistical hiding, perfect hiding; computational binding; statistical equivocality and perfect equivocality from* cs*.*

Proof. Hiding and binding are evident. For the equivocality, note that given the equivocation key ek for cs, it is possible to use $\mathsf{cs.com}^\circ$ 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 $\mathsf{cs.equiv}_{\mathsf{ek}}$ and the claim follows. □

The list commitment scheme does not have to be hiding. Hence, we can use hash trees to compress large lists into succinct digests. The corresponding construction is based on a collision-resistant hash function family $\{\mathcal{H}_k\}$ and the length of certificates is known to be of size $\mathrm{O}(k \log n)$. The Pedersen commitment scheme [27] is a good candidate of the conventional commitment scheme, as it is perfectly equivocal in the CRS model and can be easily set up in the standard model. More precisely, the public parameter is a uniformly chosen group element $y \in \langle g \rangle$ and the corresponding equivocality trapdoor is the discrete log of y. As the first option, parameters can be generated jointly by the sender and the 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 scheme is perfectly hiding. Then, we lose equivocality unless we are willing to find the discrete logarithm of y in exponential time.

4 Consistent Adaptive Oblivious Transfer

Oblivious transfer protocols are often used as building blocks for complex protocols. In an *adaptive oblivious transfer protocol*, a server has an input database $\boldsymbol{x} = (x_1, \ldots, x_n)$ of ℓ-bit strings and a client can adaptively query up to m elements from this database. The client should learn nothing beyond $x_{q_1}, \ldots, x_{q_m}$ and the the server should learn nothing. In particular, the client should learn

Client's inputs: adaptively chosen indexes $q_1, \ldots, q_m$.
Server's inputs: a database $\boldsymbol{x} = (x_1, \ldots, x_n)$.
Common inputs: a description of lc and ot.

TRUSTED SETUP

If needed, the trusted dealer executes the shared setup phase for ot.
The trusted dealer broadcasts public parameters $\mathsf{ck} \leftarrow \mathsf{lc.gen}(1^k)$ to everybody.

COMMITMENT PHASE

The server computes $(c, d) \leftarrow \mathsf{lc.com}_{\mathsf{ck}}(\boldsymbol{x})$ and sends the commitment (c, n) to the client. Then the server computes $s_i \leftarrow \mathsf{lc.cert}_{\mathsf{ck}}(d, i)$ for each $i \in \{1, \ldots, n\}$, and stores the database of partial decommitment values $\boldsymbol{s} \leftarrow (s_1, \ldots, s_n)$.

QUERY PHASE. To fetch the q_ith element form the database:

1. The client sends $Q_i \leftarrow \mathsf{ot.query}(q_i)$ to the server.
2. The server returns $R_i \leftarrow \mathsf{ot.reply}(\boldsymbol{s}, Q_i)$.
3. The client computes $A_i \leftarrow \mathsf{ot.decode}(q_i, R_i)$ and $(j, x_*) \leftarrow \mathsf{lc.open}_{\mathsf{ck}}(c, A_i)$. If $j = q_i$ then the client outputs x_*, otherwise the client outputs $\perp$.

Protocol 1: The new consistent adaptive oblivious transfer protocol

$\perp$ if its query is not in the range $\{1, \ldots, n\}$. In the asymptotic setting, all parameters m, n, ℓ must be polynomial in the security parameter k. Two standard security notions for the oblivious transfer protocol in the malicious model are security and privacy. In brief, private protocols guarantee only that a malicious client cannot learn anything beyond $x_{q_1}, \ldots, x_{q_m}$ but do not assure that an honest client indeed learns $x_{q_1}, \ldots, x_{q_m}$ if the server is malicious. As such they are inapplicable for many practical applications.

In most adaptive oblivious transfer protocols that are secure in the malicious model, the server first commits to his or her individual database elements, and then at every query helps the client to "decrypt" a single database element, see for example [7, 28]. A natural alternative is to use a sublinear-length commitment scheme together with suitable zero-knowledge techniques as detailed in [24]. However, the resulting low-communication protocol is only a theoretical solution with computational overhead that is too large for practical applications.

As a practical solution, we show how to convert any private oblivious transfer protocol into a consistent protocol with low computational and communicational overhead, see Prot. 1. By using protocols [17, 22] for oblivious transfer, we get an efficient protocol with almost optimal communication. For the sake of simplicity, we assume that the underlying private 1-out-of-n oblivious transfer protocol ot has 2 moves and is determined by a triple of algorithms $(\mathsf{query}, \mathsf{reply}, \mathsf{decode})$ such that for any $q_i \in \{1, \ldots, n\}$ and $\boldsymbol{x} \in \{0,1\}^{\ell \times n}$, we have $\mathsf{decode}(q_i, \mathsf{reply}(\boldsymbol{x}, \mathsf{query}(q_i))) = x_{q_i}$. This assumption is not a big restriction, since most practical oblivious transfer protocols are in this form, and generalization to multi-round protocols is obvious. As a second simplification, we use a trusted setup phase for generating the public parameters. One can eliminate the need for a trusted dealer by running a secure multiparty protocol, but the explicit use of the trusted setup makes security proofs more modular.

The underlying idea behind the protocol is rather simple. First, the server uses a list commitment scheme lc to commit all inputs $\boldsymbol{x}$. Then the server computes an intermediate database $\boldsymbol{s} = (s_1, \ldots, s_n)$ of certificates corresponding to every x_j. In an query phase, the client and the server execute the oblivious transfer protocol ot with the server's input $\boldsymbol{s}$ to fetch the q_jth certificate s_{q_j}. If this value opens a database element x_* that is consistent with the commitment of $\boldsymbol{x}$ and the query q_j, then we output x_*, otherwise we return $\perp$.

Theorem 3. *If the oblivious transfer protocol* ot *is computationally private in the shared setup model and the list commitment scheme* lc *is binding and equivocal, then Prot. 1 is a consistent adaptive* m*-out-of-*n *oblivious transfer protocol in the polynomial security model provided that* $n^m \in \mathrm{poly}(k)$.

Proof. For the proof, we fix a security parameter k, consider an adversary $\mathcal{A}_k$ and show how to convert it into an ideal world adversary $\mathcal{A}_k^\circ$ such that the output distributions are close enough for any input pair (ϕ_c, ϕ_s).

Security of honest client. Let $\mathcal{A}_k$ be a corrupted server and $\mathcal{C}_k$ an honest client. As the number of potential queries is polynomial, we can construct a black-box extractor $\mathcal{K}^{\mathcal{A}_k, \mathcal{C}}$ that fixes random coins of the client and the malicious server, and makes all n^m queries in order to recover all valid openings $(j, \hat{x}_j)$. As the slowdown is polynomial and the commitment scheme is binding, double openings $\hat{x}_j \neq \hat{x}'_j$ are revealed with negligible probability. Hence, we can use $\mathcal{K}^{\mathcal{A}_k, \mathcal{C}}$ in the construction of ideal world server. By the definition, the oblivious transfer protocol ot is private in the shared setup model if for any adaptively chosen inputs vectors $q = (q_1, \ldots, q_m)$ and $\overline{q} = (\overline{q}_1, \ldots, \overline{q}_m)$ the output distribution of $\mathcal{A}_k$ is computationally indistinguishable. Hence, we can replace the missing messages in the ideal world by simulating the honest receiver with input $\overline{q} = (1, \ldots, 1)$. We can combine these results and consider the following ideal world adversary $\mathcal{A}_k^\circ$:

1. Run the setup phase to obtain public parameters for lc and ot.
2. Choose randomness ω and store $(\hat{x}_1, \ldots, \hat{x}_n) \leftarrow \mathcal{K}^{\mathcal{A}_k, \mathcal{C}_k}(\phi_s, \mathsf{ck}; \omega)$.
3. Send $(\hat{x}_1, \ldots, \hat{x}_n)$ to TTP and specify halting predicates $\pi_1, \ldots, \pi_m$ through the interaction between the client $\mathcal{C}_k(q)$ and the adversary $\mathcal{A}_k(\phi_s, \mathsf{ck}; \omega)$, that is, $\pi_i(q_1, \ldots, q_i) = 1$ iff $\mathcal{C}_k$ with input $q_1, \ldots, q_i$ obtains $x_{q_i} \neq \perp$.
4. Output whatever $\mathcal{A}_k(\phi_s, \mathsf{ck}; \omega)$ outputs in interaction with $\mathcal{C}_k(\overline{q})$.

Let (ψ_c, ψ_s) denote the outputs of the real execution and $(\psi_c^\circ, \psi_s^\circ)$ the outputs of the ideal execution. W.l.o.g. we can assume that the output of $\mathcal{A}_k$ contains $\phi_s, \mathsf{ck}, \omega$ and thus given the advice ϕ_c we can efficiently compute ψ_c form ψ_s or ψ_s°. Hence, the distributions (ψ_c, ψ_s) and (ψ_c, ψ_s°) must be computationally indistinguishable, or otherwise we can distinguish ψ_s form ψ_s°, which violates the privacy of ot. Now, note that for fixed $(\phi_s, \mathsf{ck}, \omega)$ the corresponding outputs ψ_c and ψ_c° can differ only if the client recovers $x_{q_j} \neq \hat{x}_{q_j}$. As this can happen with negligible probability, the distributions (ψ_c, ψ_s°) and $(\psi_c^\circ, \psi_s^\circ)$ must be computationally indistinguishable and thus also (ψ_c, ψ_s) and $(\psi_c^\circ, \psi_s^\circ)$.

SECURITY OF HONEST SERVER. Since the output of the server in the ideal and real model is empty, only the output of a malicious client $\mathcal{A}_k$ must be analyzed. Consider a hybrid implementation of the protocol, where all instances of ot are replaced with ideal implementations of oblivious transfer protocol with the database $\boldsymbol{s}$. Then as the ot protocol is private in the shared setup model, there exists an adversary $\mathcal{A}_k^*$ such that the output distributions of $\mathcal{A}_k$ and $\mathcal{A}_k^*$ are computationally indistinguishable.

To complete the proof, we construct a true ideal world adversary $\mathcal{A}_k^\circ$ and show that the outputs of $\mathcal{A}_k^\circ$ and $\mathcal{A}_k^*$ are computationally indistinguishable. Indeed, let the ideal world adversary $\mathcal{A}_k^\circ$ proceed as follows:

1. Generate the equivocality key $(\mathsf{ek}, \mathsf{ck}) \leftarrow \mathsf{lc.gen}^\circ(1^k)$ and broadcast ck.
2. Compute $(\hat{c}, \eta) \leftarrow \mathsf{lc.com}^\circ_{\mathsf{ck},\mathsf{ek}}(n)$ and send $\hat{c}$ to the adversary $\mathcal{A}_k^*$.
3. If $\mathcal{A}_k^*$ queries q_j, obtain x_{q_j} from TTP and reply $\hat{s}_j \leftarrow \mathsf{lc.equiv}_{\mathsf{ek}}(\hat{c}, \eta, q_j, x_{q_j})$.
4. Return whatever the adversary $\mathcal{A}_k^*$ finally outputs.

Then it is easy to see that in the hybrid world $\mathcal{A}_k^*$ plays the first equivocality game with the honest server and in the ideal world $\mathcal{A}_k^*$ plays the second equivocality game with a challenger consisting of the simulator $\mathcal{A}_k^\circ$, TTP and the honest server. To nitpick, $\mathcal{A}_k^*$ does not query all faked decommitment values at once, but clearly we can write a wrapper that queries all decommitments and then gradually releases them to $\mathcal{A}_k^*$. Thus, the outputs of $\mathcal{A}_k^*$ and $\mathcal{A}_k^\circ$ must be computationally indistinguishable or otherwise $\mathcal{A}_k^*$ together with the distinguisher would break the equivocality property. □

Corollary 1. *If* ot *is (weakly) statistically server-private and* lc *is statistically equivocal, then Prot. 1 is (weakly) statistically server-private.*

Proof. If ot is statistically private, then for each $\mathcal{A}_k$ there exists $\mathrm{poly}(k)$ times slower $\mathcal{A}_k^*$ such that the output distributions are statistically close. Weak statistical privacy guarantees only the existence of $\mathcal{A}_k^*$ without bounds on the running time. Both claims follow as $\mathcal{A}_k^\circ$ is only $\mathrm{poly}(k)$ times slower than $\mathcal{A}_k^*$. □

The limitation that the number of potential queries must be polynomial in k seems to be essential for getting a low-communication solution with a small computational overhead. To bypass this restriction, we can either use list commitment schemes that are both extractable and equivocal.

Corollary 2. *If the oblivious transfer protocol* ot *is computationally private in the shared setup model and the list commitment scheme* lc *is (weakly) white-box extractable and equivocal, then Prot. 1 is a consistent adaptive m-out-of-n oblivious transfer protocol in the (weak) polynomial security model.*

Proof. Note that the algorithm pair $(\mathcal{A}_k, \mathcal{C}_k)$ can be treated as a compound adversary, which generates a list commitment (c, n) and then later opens m elements according to the advice $a = (q_1, \ldots, q_m)$. As the commitment scheme is white-box extractable, there exists an extractor machine $\mathcal{K}_{\mathcal{A}_k,\mathcal{C}_k}$ that, given the parameters ck, the server's input ϕ_s and the random tape ω, outputs a list

of candidate elements $\hat{\boldsymbol{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ such that at the end of the execution $\mathcal{C}_k$ accepts $x_{q_j} \neq \hat{x}_{q_j}$ with negligible probability. This extractor can be used in the simulator construction of Thm. 3 instead of $\mathcal{K}^{\mathcal{A}_k, \mathcal{C}_k}$.

Weak extractability. The same construction is valid for a weakly extractable commitment scheme. However, in this case for any notable error bound $\varepsilon(k)$, we can choose $\mathcal{K}_{\mathcal{A}_k, \mathcal{C}_k}$ such that (ψ_c, ψ_s°) and $(\psi_c^\circ, \psi_s^\circ)$ are $\varepsilon(k)$-close. As (ψ_c, ψ_s) and (ψ_c, ψ_s°) are computationally indistinguishable, we can guarantee that, for a large enough k, distributions (ψ_c, ψ_s) and $(\psi_c^\circ, \psi_s^\circ)$ are computationally $2\varepsilon(k)$-close. As the slowdown is $\mathrm{O}(\mathrm{poly}(k)/\varepsilon(k))$, we have established that for any notable error bound $\varepsilon(k)$, we can construct a polynomial-time ideal world adversary, i.e., the correspondence $\{\mathcal{A}_k\} \mapsto \{\mathcal{A}_k^\circ\}$ is valid in the weak polynomial model. □

Comparison with other protocols. If n^m is polynomial in k, then we can use very communication efficient list commitments that stretch the input $\mathrm{O}(k \log n)$. By combining it with the most efficient private oblivious transfer protocol [17] we get a protocol with a communication complexity $\mathrm{O}(k \cdot m \log^2 n)$. Moreover, if we neglect the setup, then for the amortized round complexity is two messages per query. The latter is significantly better than the communication complexity $\Omega(mn)$ of the secure adaptive oblivious transfer protocols [11, 10, 6, 25] relying on zero-knowledge proofs. With an explicit use of the PCP theorem one can achieve polylogarithmic communication [24] but this approach is only optimal in the asymptotic sense.

As for the computational complexity, note that additional computational overhead (compared to private protocols) comes from the commitment phase. For a hash tree based list commitment scheme, this computational overhead is $\mathrm{O}(n)$ hashing and commitment operations. If the number of queries is bounded or $n^m \in \mathrm{poly}(k)$, then there are no additional costs besides computing the commitment. If the server must handle an unbounded number of queries, the server has to prove that he or she knows how to open the commitment. In a communication inefficient version proof, the server sends all lower level commitment values $c_1, \ldots, c_n$ to the client and proves knowledge of each decommitment value. The client first checks that the root of the Merkle tree is correct and then verifies individual proofs. Such zero-knowledge proofs are particularly efficient for Petersen commitments. Again the overhead is $\mathrm{O}(n)$ operations. By using suitable conversion methods [24] we can achieve polylogarithmic communication by increasing the computational overhead by a polynomial factor. Although the construction still relies on the PCP theorem, the underlying proof is much simpler—the server does not have to prove correctness in the query phases.

Aumann and Lindell described a 1-out-of-2 oblivious transfer protocol [2], which is secure in the covert model. Although the resulting security guarantees are weaker than for the consistent protocol, see Table 1, their protocol still has 7 messages and a much higher communication complexity. To be fair, three of those messages are used to implement trusted setup for the private oblivious transfer but there are still 4 messages per query and a malicious sender can change its input during the protocol.

5 Consistent Conditional Disclosure of Secrets

Let $\boldsymbol{q} = (q_1, \ldots, q_n)$ denote the client's vector of inputs and let x be a secret possessed by the server. Then *conditional disclosure of secrets* (CDS) for a predicate ρ is a protocol, where the client should learn

$$\mathsf{cds}_\rho(\boldsymbol{q}, x) = \begin{cases} x, & \text{if } \rho(\boldsymbol{q}) = 1 \ , \\ \perp, & \text{otherwise} \ , \end{cases}$$

and the server should learn nothing. CDS protocols are often used to convert client-server protocols secure in a semihonest model to protocols that preserve the privacy of inputs in the malicious model, see [1, 21] for the details.

In the context of the current work, we are more interested in the direct application of CDS protocols. Namely, note that a CDS protocol provides a way to distribute a secret only if the client's input satisfies certain condition, i.e., the client has credentials to access the secret. As an example, consider a video on demand service, where a client should obtain a key to a video stream only if his or her balance is non-negative: $\mathit{credit} > 0$. However, the server should be unable to tell the client's exact balance. The CDS protocols described in [1, 21] consist of two moves and the client's *query* consists of ciphertexts of $q_1, \ldots, q_n$. As the CDS protocols of say [21] are based on an additively homomorphic cryptosystem, the server can do a limited amount of cryptocomputing to form ciphertexts that decrypt to the secret if the condition ρ is met. Thus, the client must often send some additional encryptions of auxiliary inputs $w_1, \ldots, w_n$ to help the server, i.e., $\boldsymbol{q} = (\mathit{credit}, w_1 \ldots, w_n)$ for our example. Since the solutions [1, 21] provide only privacy in the malicious model, it is difficult to prove that the server maliciously declines access and the server cannot easily refute false accusations.

Now consider an extended CDS protocol, where the server first publicly commits to x and the CDS protocol is executed to recover the corresponding decommitment value. As the proof of Thm. 3 and Cor. 2 directly generalizes, the resulting protocol is consistent under the same assumptions. If the set of plausible client inputs is exponential, the exhaustive knowledge extraction technique from Thm. 3 becomes infeasible and the construction, where the server proves that he or she knows how to open the commitment is the only option. The latter is not a big problem, as many conventional commitment schemes have efficient proofs of knowledge for this. For instance, the equivocal Pedersen commitment scheme has this property. Also, note that the server does not have to prove knowledge of the decommitment value to everybody. It is sufficient, if the server proves it to a respected peer during an initialization phase. If we can guarantee that such *auditors* are semihonest, then we can further optimize the proof.

Moreover, Thm. 1 assures that the client can prove that the server acts maliciously to third parties. As anybody can repeat the second phase of a CDS protocol enlisted in [1, 21] with a different secret $\overline{x}$, the corresponding honest-verifier zero-knowledge proof is very efficient. The complaining client has to reveal $\overline{x}$ to the prover and then additionally prove (in zero-knowledge if necessary) that the reply of the server is invalid.

The ability to complain makes CDS protocols very appealing in TV or military broadcasts with complex access policy, where credentials are granted by giving out random keys. This problem is commonly known as *private inference control* [31]. In this setting, a 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 is to release correct keys. 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. However, 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 make the service *accountable* against inside attacks the client should be able to prove to third parties that the denial is illegitimate.

We emphasize that the proofs of knowledge can be skipped if it is possible to force the server to construct commitments of keys semi-honestly during the initialization phase either by organizational means or by auditing. As efficient CDS protocols exists for all **NP/poly** predicates [21], we have established that accountable private inference control is possible. More importantly, the solution is really practical if a complaining client is willing to reveal his input.

6 Discussion And Open Problems

Both solutions for oblivious transfer and conditional disclosure of secrets are based on a simple principle: the server first creates a list of possible answers and commits to it. Since all answers are independent of each other and a client can verify that the answer is correct, the server has to prove only that he knows how to decommit and not that all answers are consistent with some server's input. As soon as the answers must satisfy a certain constraint or the client cannot check whether he or she obtained a decommitment value for a correct answer, the construction of a consistent protocol becomes much more complicated.

Nevertheless, any such protocol must give rise to a list commitment scheme. Indeed, we can view any client-server protocol for computing $f(q, x)$ as a compact commitment to a list with elements $x_q = f(q, x)$ where q takes all plausible values. For three-move protocols, the first message is the commitment and the second message together with the third corresponds to interactive opening procedure. The second and third message can be compacted into a single decommitment value provided that a colluding client and server cannot fool third parties who know the first message. As the query should not leak information about other entries, construction of such commitment schemes with implicit correctness guarantees seems a highly non-trivial task. Hence, the question whether one can construct three-move consistent protocols for other tasks is an interesting question, which might shed a light on what type of restrictions are implicitly enforceable by the design of a list commitment scheme.

Another open question is how much can be learned from the complaints and whether is it possible to limit this exposure. By the definition of consistency

the complaint leaks an output of a polynomial-time randomized predicate. In practice, we can further restrict the set of enforceable predicates π. For instance, one can force memoryless consistency in the oblivious transfer protocol. Namely, a client-server protocol is *memoryless-consistent* if the halting predicates $\pi_1, \ldots, \pi_m$ are independent from previous queries, i.e., $\pi_i(q_1, \ldots, q_i) = \pi_i(q_i)$ and the server cannot relate results of different queries.

Theorem 4. *Prot. 1 is memoryless consistent if no instantiations of* ot *protocols share random variables.*

Proof. Assume that an adversary A breaks the memoryless-consistent property of Prot. 1. That is, it can force the client to abort iff a predicate π_i holds on client's queries $(q_1, \ldots, q_i)$, where π_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 $\pi_i(q_a, q_b)$. □

Analogous results can be stated for protocols consisting of several CDS protocols. However, memoryless consistency has a certain cost. Many efficient protocols for oblivious transfer [30, 1, 22] and CDS [1, 21] are based on homomorphic encryption. In these protocols, the trusted setup phase assures that the client indeed knows the secret key. This setup phase is replaced with a corresponding proof of knowledge in practice. Now, if each sub-protocol has a different key pair, the preprocessing phase becomes rather complex. Hence, it is beneficial to share the key among many protocol instances, see [21] for further details.

By doing so we loose memoryless consistency and thus a natural question arises: can we still bound the set of enforceable halting predicates. As all of these protocols send the client input in an encrypted form to the server and the replies are also encryptions, it is easy to force affine predicates. Given a list of encryptions $\mathsf{Enc}(q_1), \ldots, \mathsf{Enc}(q_\ell)$, the server can multiply all replies with

$$\mathsf{Enc}((q_1\alpha_1 + \cdots + q_i\alpha_i - \beta)r) = (\mathsf{Enc}(q_1)^{\alpha_1} \cdots \mathsf{Enc}(q_i)^{\alpha_i}\mathsf{Enc}(-\beta))^r$$

for a random message space element r. As a result, the replies are unaltered when $q_1\alpha_1 + \cdots + q_i\alpha_i = \beta$ and uniformly distributed otherwise. Consequently, the server can easily force halting predicates corresponding to *affine combinations* of received ciphertexts $\pi_i(q_1, \ldots, q_i) = [q_1\alpha_1 + \cdots + q_i\alpha_i = \beta]$. By multiplying replies with several such ciphertexts, the server can also force conjunctions of such affine combinations.

Note that these attacks are applicable for any additively homomorphic encryption scheme. Hence, one can ask whether this is a *complete* description of halting predicates or not. Of course, this question makes sense only for deterministic predicates, as any client server interaction can be formalized as a randomized predicate. For all deterministic predicates, it is reasonable to compare the behavior of a concrete cryptosystem with its idealized counterpart that is implemented through encryption, decryption and ciphertext-addition oracles.

We say that a homomorphic cryptosystem has *special cryptocomputing properties* if the malicious server can force deterministic predicates that cannot be forced if the underlying cryptosystem is ideal. As there are cryptosystems that allow to cryptocompute quadratic polynomials [4] and even polynomials of any length [16], cryptosystems with special properties exist. However, in all of these cases these properties follow directly from the design of a cryptosystem. Thus, it is reasonable to assume that standard additively homomorphic cryptosystems, such as Paillier [26], are without special properties and the set of enforceable predicates is limited to affine tests and their conjunctions. Any provable rejection to this fact would be interesting by itself as it would advance the set of cryptocomputable predicates.

Acknowledgments. Both authors were supported by the European Regional Development Fund through the Estonian Centre of Excellence in Computer Science, EXCS. The second author was also supported by Estonian Science Foundation, grant #8058.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced Oblivious Transfer: How to Sell Digital Goods. In Proc. of EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135, Springer-Verlag, 2001.
2. Aumann, Y., Lindell, Y.: Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In Proc. of TCC 2007. LNCS, vol. 4392, pp. 137–156, Springer, 2007.
3. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In Proc. of CRYPTO 1992. LNCS, vol. 740, pp. 390–420, Springer-Verlag, 1993.
4. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In Proc. of TCC 2005. LNCS, vol. 3378, pp. 325–341, Springer-Verlag, 2005.
5. Buldas, A., Laur, S.: Knowledge-Binding Commitments with Applications in Time-Stamping. In Proc. of PKC 2007. LNCS, vol. 4450, pp. 150–165, Springer-Verlag, 2007.
6. Cachin, C., Camenisch, J.: Optimistic Fair Secure Computation. In Proc. of CRYPTO 2000. LNCS, vol. 1880, pp. 93–111, Springer-Verlag, 2000.
7. Camenisch, J., Neven, G., Shelat, A.: Simulatable Adaptive Oblivious Transfer. In Proc. of EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590, Springer-Verlag, 2007.
8. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology **13**(1), pp. 143–202, 2000.
9. Canetti, R., Ostrovsky, R.: Secure Computation with Honest-Looking Parties: What If Nobody Is Truly Honest? In Proc. of STOC 1999, pp. 255–264, ACM Press, 1999.
10. Cramer, R., Damgård, I.: Linear zero-knowledge – a note on efficient zero-knowledge proofs and arguments. In Proc. of STOC 1997, pp. 436–445, ACM Press, 1997.
11. Crépeau, C., van de Graaf, J., Tapp, A.: Committed Oblivious Transfer and Private Multi-Party Computation. In Proc. of CRYPTO 1995. LNCS, vol. 963, pp. 110–123, Springer-Verlag, 1995.

12. Crescenzo, G.D., Ishai, Y., Ostrovsky, R.: Non-Interactive and Non-Malleable Commitment. In Proc. of STOC 1998, pp. 141–150, ACM Press, 1998.
13. Di Crescenzo, G.: Equivocable And Extractable Commitment Schemes. In Proc. of SCN 2002. LNCS, vol. 2576, pp. 74–87, 2002.
14. Di Crescenzo, G., Lipmaa, H.: Succinct NP Proofs from An Extractability Assumption. In Proc. of CIE 2008. LNCS, vol. 5028, pp. 175–185, Springer-Verlag, 2008.
15. Franklin, M.K., Yung, M.: Communication complexity of secure computation. In Proc. of STOC 1992, pp. 699–710, ACM Press, 1992.
16. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In Proc. of STOC 2009, pp. 169–178, ACM Press, 2009.
17. Gentry, C., Ramzan, Z.: Single-Database Private Information Retrieval with Constant Communication Rate. In Proc. of ICALP 2005. LNCS, vol. 3580, pp. 803–815, Springer-Verlag, 2005.
18. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, 2001.
19. Goldreich, O.: On Expected Probabilistic Polynomial-Time Adversaries: A Suggestion for Restricted Definitions and Their Benefits. In Proc. of TCC 2007. LNCS, vol. 4392, pp. 174–193, Springer-Verlag, 2007.
20. Laur, S., Lipmaa, H.: On the Feasibility of Consistent Computations. Eprint 2006/088.
21. Laur, S., Lipmaa, H.: A New Protocol for Conditional Disclosure of Secrets And Its Applications. In Proc. of ACNS 2007. LNCS, vol. 4521, pp. 207–225, Springer-Verlag, 2007.
22. Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In Proc. of ISC 2005. LNCS, vol. 3650, pp. 314–328, Springer-Verlag, 2005.
23. Mohassel, P., Franklin, M.K.: Efficiency Tradeoffs for Malicious Two-Party Computation. In Proc. of PKC 2006. LNCS, vol. 3958, pp. 458–473, Springer-Verlag, 2006.
24. Naor, M., Nissim, K.: Communication Preserving Protocols for Secure Function Evaluation. In Proc. of STOC 2001, pp. 590–599, ACM Press, 2001.
25. Ogata, W., Kurosawa, K.: Oblivious Keyword Search. Journal of Complexity **20**(2–3), pp. 356–371, 2004.
26. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proc, of EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238, Springer-Verlag, 1999.
27. Pedersen, T.P.: Non-Interactive And Information-Theoretic Secure Verifiable Secret Sharing. In Proc. of CRYPTO 1991. LNCS, vol. 576, pp. 129–140, Springer-Verlag, 1991.
28. Peikert, C., Vaikuntanathan, V., Waters, B.: A Framework for Efficient And Composable Oblivious Transfer. In Proc. of CRYPTO 2008. LNCS, vol. 5157, pp. 554–571, Springer-Verlag, 2008.
29. Santis, A.D., Crescenzo, G.D., Persiano, G.: Necessary and Sufficient Assumptions for Non-iterative Zero-Knowledge Proofs of Knowledge for All NP Relations. In Proc. of ICALP 2000. LNCS, vol. 1853, pp. 451–462, Springer-Verlag, 2000.
30. Stern, J.P.: A New And Efficient All Or Nothing Disclosure of Secrets Protocol. In Proc. of ASIACRYPT '98. LNCS, vol. 1514, pp. 357–371, Springer-Verlag, 1998.
31. Woodruff, D.P., Staddon, J.: Private Inference Control. In Proc. of ACMCCS2004, pp. 188–197, ACM Press, 2004.