Somewhat Non-committing Encryption and Efficient Adaptively Secure Oblivious Transfer*

Juan A. Garay[1], Daniel Wichs[2], and Hong-Sheng Zhou[3,**]

[1] AT&T Labs – Research
garay@research.att.com
[2] New York University
wichs@cs.nyu.edu
[3] University of Connecticut
hszhou@cse.uconn.edu

Abstract. Designing efficient cryptographic protocols tolerating adaptive adversaries, who are able to corrupt parties on the fly as the computation proceeds, has been an elusive task. In this paper we make progress in this area. First, we introduce a new notion called *semi-adaptive* security which is slightly stronger than static security but *significantly weaker than fully adaptive security*. The main difference between adaptive and semi-adaptive security is that semi-adaptive security allows for the case where one party starts out corrupted and the other party becomes corrupted later on, but *not* the case where both parties start out honest and become corrupted later on. As such, semi-adaptive security is much easier to achieve than fully adaptive security. We then give a simple, generic protocol compiler which transforms any semi-adaptively secure protocol into a fully adaptively secure one. The compilation effectively decomposes the problem of adaptive security into two (simpler) problems which can be tackled separately: the problem of semi-adaptive security and the problem of realizing a weaker variant of secure channels.

We solve the latter problem by means of a new primitive that we call *somewhat non-committing encryption* resulting in significant efficiency improvements over the standard method for realizing secure channels using (fully) non-committing encryption. Somewhat non-committing encryption has two parameters: an equivocality parameter ℓ (measuring the number of ways that a ciphertext can be "opened") and the message sizes k. Our implementation is very efficient for small values ℓ, *even* when k is large. This translates into a very efficient compilation of semi-adaptively secure protocols for tasks with small input/output domains (such as bit-OT) into fully adaptively secure protocols.

Indeed, we showcase our methodology by applying it to the recent Oblivious Transfer protocol by Peikert *et al.* [Crypto 2008], which is only secure against static corruptions, to obtain the first efficient, adaptively secure and composable OT protocol. In particular, to transfer an n-bit message, we use a constant number of rounds and $O(n)$ public key operations.

* Partial work was carried out when the first author was at Bell Labs and the second and the third authors were visiting Bell Labs.

** Research supported by NSF grants 0447808 and 0831306.

S. Halevi (Ed.): CRYPTO 2009, LNCS 5677, pp. 505–523, 2009.

1 Introduction

When defining the security of cryptographic protocols, we generally strive to capture as wide a variety of adversarial attacks as possible. The most popular method of doing so is the *simulation paradigm* [17] where the security of a real-world protocol is compared to that of an ideal-world (perfectly secure) implementation of the same task. Within the simulation paradigm there are several flavors. Firstly, basic simulation only guarantees security for single copy of a protocol executing in isolation. The *Universal Composability* (UC) framework [3,4] extends the simulation paradigm and defines security for protocols executed in arbitrary environments, where executions may be concurrent and even maliciously interleaved. Secondly, we generally distinguish between *static* and *adaptive* security. Static security protects against an adversary who controls some fixed set of corrupted parties throughout the computation. Adaptive security, on the other hand, defends against an adversary who can corrupt parties adaptively at any point during the course of the protocol execution For adaptive security, we also make a distinction between the *erasure model*, where honest parties are trusted to securely erase data as mandated by the protocol, and the *non-erasure model*, where no such assumptions are made. Traditionally, the design of protocols in the non-erasure model is viewed as significantly more difficult. For example, and in contrast to the erasure model, we do not have general constant-round protocols for many tasks, and even many simple tasks (e.g., encryption) seem to be less efficient in rounds and computation. Nevertheless, although solutions in the erasure model may in some scenarios be acceptable, it is both of fundamental interest and practical value to achieve the stronger security notion whenever possible; this is the subject of this work.

The seminal result of [8] shows that it is theoretically possible to design an adaptively secure and universally composable protocol for a large class of natural tasks, assuming the presence of some trusted setup such as a randomly selected common reference string (CRS). Unfortunately, the final protocol of [8] should be viewed as a *purely theoretical* construction, ad its reliance on expensive Cook-Levin reductions precludes a practical implementation. Alternative efficient approaches to two-party and multi-party computation received a lot of attention in the recent works of [13,16,20,21,22,24,25]. However, all of these results sacrifice some aspect of security to get efficiency – e.g., they only provide (stand-alone or UC) static security, or UC adaptive security but only for honest majority, or UC adaptive security in the erasure model, etc. The recent work of [20] *can* provide UC adaptive security for all (well-formed) tasks in constant rounds assuming the adversary corrupts all but one of the participants, but only given an efficient adaptively secure Oblivious Transfer (OT) protocol. However, as we will discuss, no such protocols were known. Lastly, we mention the work of [5], which gives a generic compiler from static to adaptive security using secure channels. Unfortunately, this compiler does not provide full adaptive security (does not allow for post-execution corruptions) and, as was noted in [24], crucially relies on rewinding and hence cannot be used in the UC framework.

Indeed, thus far *no* efficient protocols for general multi-party computation, or even for many specific two-party function evaluation tasks, achieve adaptive security. This is not surprising given the difficulty of realizing adaptive security for even the most fundamental task in cryptography: *secure communication*. As was observed in [6], standard security notions for encryption do not suffice. Adaptively secure communication schemes, also called *non-committing encryption* schemes, were introduced and constructed in [6] and studied further in [1,11], but these protocols are fairly complicated and inefficient for large messages.

It turns out that many useful two-party tasks (e.g., Oblivious Transfer, OR, XOR, AND, Millionaires' problem, etc.) are *strictly harder* to achieve than secure communication, the reason being that these tasks allow two honest parties to communicate by using the corresponding ideal functionality. For example, using OT, an honest sender can transfer a message to a receiver by setting it as *both* of his input values. Therefore, an adaptively secure OT protocol for the transfer of k bit messages can be used as a non-committing encryption of a k bit message and so all of the difficulty and inefficiency of non-committing encryption must **also** appear in protocols for tasks such as OT. Further, unlike secure communication, many tasks also require security against the active and malicious behavior of the *participants*. This might lead us to believe that the two difficulties will be compounded making efficient adaptively secure implementations of such tasks infeasible or too complicated to contemplate.

Taking Oblivious Transfer as an example, this indeed seems to be the case. The recent work of [26], proves a (black-box) separation between enhanced trapdoor permutations (which allow for static OT) and adaptively secure OT, showing that the latter is indeed "more complex" in a theoretical sense. This complexity is reflected in practice as well. We are aware of only two examples (albeit inefficient) of adaptively secure OT protocols, from [2] and [8]. Both of these works first construct an OT protocol for the honest-but-curious setting and then compile it into a fully secure protocol using generic and inefficient zero knowledge proofs. In both constructions, the underlying honest-but-curious OT protocols rely on ideas from non-committing encryption[1] and hence inherit its complexity. Since the full constructions require us to run zero knowledge proofs *on top of* the complex underlying honest-but-curious protocol, there is little hope of making them efficient by only using proofs for simple relations. This is in contrast to static security (and adaptive security in the erasure model) for which we *have* recently seen efficient constructions of OT protocols. For example, [14,16,21] construct OT protocols by only using simple and efficient zero-knowledge proofs. Interestingly, Ishai *et al.* [19] give the first OT protocol constructions against malicious corruptions *without* using zero knowledge proofs; this result was later strengthened in [18]. Two very recent and efficient concrete protocols not using zero-knowledge proofs are given in [23,28]. The protocol of [28] is particularly exciting since it is a UC-secure protocol in the CRS model which runs in two rounds and uses a constant number of

[1] The protocol of [2] implicitly uses the plug-and-play approach from [1], while the protocol of [8] uses non-committing encryption in a generic way.

public key operations. Achieving adaptive security based on these protocols has, however, remained as an open problem.

Our contributions. In this work we construct the first efficient (constant round, $O(n)$ public-key operations for the transfer of an n-bit message) adaptively secure Oblivious Transfer protocol in the non-erasure model. Along the way we develop several techniques of independent interest which are applicable to adaptive security in general.

First, we introduce a new notion called *semi-adaptive* security which is slightly stronger than static security but significantly weaker than fully adaptive security. At a high level, semi-adaptive security allows for the case where one party starts out corrupted and the other party becomes corrupted later on, but *not* the case where both parties start out honest and become corrupted later on. In particular, a semi-adaptively secure protocol for a task like OT, *does not* yield a non-committing encryption scheme and hence does not (necessarily) inherit its difficulty. We then give a generic compiler which transforms any semi-adaptively secure protocol into a (fully) adaptively secure protocol. The compiler is fairly simple: we take the original protocol and execute it over a secure communication channel (i.e., all communication from one party to another is sent over a secure channel). The compilation effectively decomposes the problem of adaptive security into two (simpler) problems which can be tackled separately: the problem of semi-adaptive security and the problem of realizing secure channels.

Unfortunately, we saw that the construction of secure-channels is a difficult problem and existing solutions are not very efficient. Also, as we already mentioned, we cannot completely bypass this problem since adaptive security for many tasks *implies* secure channels. However, for the sake of efficiency, we would like to limit the use of secure channels (and hence the use of non-committing encryption) to a minimum. For example, we know that an OT protocol for one-bit messages implies a non-committing encryption of a one-bit message. However, to get adaptive security for a bit-OT protocol, our compiler, as described above, would use non-committing encryption to encrypt the *entire* protocol transcript, and hence much more than one bit!

We fix this discrepancy by introducing a new notion called *somewhat non-committing encryption*. Somewhat non-committing encryption has two parameters: the equivocality ℓ (measuring just how *non-committing* the scheme is) and the message size k. We first observe that somewhat non-committing encryption is efficient for small values of the equivocality parameter ℓ, *even* when k is large (i.e., when we encrypt long messages). Secondly, we observe that our compiler can use somewhat non-committing encryption where the equivocality ℓ is proportional to the size of the input and output domains of the functionality. As a result, we obtain a very efficient compiler transforming any semi-adaptively secure protocol for a task with small input/output domains (such as bit-OT) into a fully adaptively secure protocol. We also show that this methodology can, in special cases, be applied to tasks with larger domain sizes such as string-OT with long strings.

We apply our methodology to the OT protocol of Peikert *et al.* [28], resulting in the first efficient and adaptively secure OT protocols. Peikert *et al.* actually present a general framework for constructing static OT, and instantiate this framework using the Quadratic Residuocity (QR), Decisional Diffie-Hellman (DDH), and Lattice-based assumptions. In this work, we concentrate on the QR and DDH based schemes. We show that relatively small modifications suffice to make these schemes semi-adaptively secure. We then employ our compiler, using somewhat non-committing encryption, to convert them into (fully) adaptively UC-secure OT protocols. As we mentioned previously, the work of [20] shows how to efficiently realize all (well-formed) m-party functionalities with adaptive security assuming up to $m-1$ corruptions, given and adaptively secure OT protocol. Therefore, by plugging in our OT protocol construction, we get improved efficiency in generically compiled protocols for all tasks as above.

Concurrent and independent work. Following the line of work of [19,18], the recent result by Choi *et al.* [9] gives a generic black-box compiler from semi-honest adaptively secure OT to fully malicious adaptively secure OT, using cut-and-choose techniques. Although the end result of our work is the same (adaptively secure OT), the two works take very different approaches which complement each other well: the compiler of [9] transforms semi-honest + adaptive security into malicious + adaptive security in the special case of OT, while our compiler is a general transformation from malicious + semi-adaptive security to malicious + adaptive security. The two starting notions of security (semi-honest + adaptive vs. malicious + semi-adaptive) are incomparable and thus both compilers are useful in different scenarios. In particular, our compiler can be used in conjunction with the OT protocol of [28] and results in an extremely efficient adaptively secure OT protocol using a constant number of rounds and $O(n)$ public-key operations to transfer an n-bit string.[2] In contrast, the compiler of [9] shows how to base adaptively secure OT on a simulatable cryptosystem in a black-box way, but at the expense of running $\Omega(\lambda^2)$ copies of the underlying semi-honest OT protocol, where λ is the security parameter, and thus requiring $\Omega(\lambda^2 n)$ operations for n-bit OT. Therefore our OT protocol can be significantly more efficient.

Due to space limitations, proofs, together with background material, full description of our enhanced version of the QR dual-mode cryptosystem, efficiency considerations, and our DDH version of adaptively secure bit- and string-OT, can be found in the full version of the paper [15].

2 Somewhat Non-committing Encryption

2.1 Adaptive Security in Two-Party Protocols

What are some of the challenges in achieving adaptive security for a two-party protocol? Let's assume that a protocol π between two parties P_0, P_1 realizes a

[2] Technically, if one thinks of n as a function of λ, we require $O(\max(\lambda, n))$ operations.

task $\mathcal{F}$ with respect to static adversaries. That means that there is a static simulator which can simulate the three basic cases: both parties are honest throughout the protocol, exactly one party is corrupted throughout the protocol or both parties are corrupted throughout the protocol. To handle adaptive adversaries, we require two more capabilities from our simulator: the ability to simulate a *first corruption* (i.e., the case that both parties start out honest and then one of them becomes corrupted) and simulating the *second corruption* (i.e., one party is already corrupted and the other party becomes corrupted as well).

Simulating the first corruption is often the harder of the two cases. The simulator must produce the internal state for the corrupted party in a manner that is consistent with the protocol transcript so far and with the actual inputs of that party (of which the simulator had no prior knowledge). Moreover, the simulator needs to have all the necessary trapdoors to continue the simulation *while* only one party is corrupted. Achieving both of these requirements at once is highly non-trivial and this is one of the reasons why efficient protocols for adaptively secure two-party computation have remained elusive.

Interestingly, simulating the first corruption becomes much easier if the protocol π employs secure channels for all communication between parties. At a high level, the simulator does not have to do any work while both parties are honest, since the real-world adversary does not see any relevant information during this time! When the first party *becomes* corrupted, we can just run a static simulation for the scenario in which this party *was* corrupted from the beginning but acting honestly and using its input. Then, we can "lie" and pretend that this communication (generated *ex post facto*) actually *took place* over the secure channel when both parties were honest. The lying is performed by setting the internal state of the corrupted party accordingly. Since our lie corresponds to the simulation of a statically corrupted party (which happens to act honestly), all of the trapdoors are in place to handle future mischievous behavior by that (freshly corrupted) party. The only problem left is in handling the second corruption – but this is significantly easier! To formalize this, we will define a notion of *semi-adaptive security* where the simulator needs to be able to simulate static corruptions as well as the case where one party starts out corrupted and the other party becomes corrupted later on (but *not* the case where *both* parties start out honest and may become corrupted later). The formal notion (with some additional restrictions imposed on the simulator) appears in Section 2.4.

Informally, we have argued that if two-party protocol is semi-adaptively secure, then the protocol is also fully adaptively secure if all communication between the parties is sent over an idealized secure channel. Unfortunately, idealized secure channels are hard to achieve physically and implementing such channels cryptographically in the real world requires the inefficient use of *non-committing encryption* [6] to encrypt the entire protocol transcript. Luckily, it turns out that we often do not need to employ fully non-committing encryption to make the above transformation hold. Indeed, we define a weaker primitive called *somewhat non-committing encryption* and show that this primitive can be implemented with significantly greater efficiency than (fully) non-committing

encryption, and that it is often *good enough* to transform a semi-adaptively secure protocol into a fully adaptively secure protocol when the sizes of the input/output domains are small.

2.2 Defining *Somewhat* Non-committing Encryption

First recall the notion of non-committing encryption from [6], which is a protocol used to realize secure channels in the presence of an *adaptive* adversary. In particular, this means that a simulator can produce "fake" ciphertexts and later explain them as encryptions of *any possible* given message. Several non-committing encryption schemes have appeared in literature [6,1,11], but the main disadvantage of such schemes is the computational cost. All of the schemes are interactive (which was shown to be necessary in [27]) and the most efficient schemes require $\Omega(1)$ public-key operations *per bit* of plaintext.

We notice that it is often unnecessary to require that the simulator can explain a ciphertext as the encryption of *any* later-specified plaintext. Instead, we define a new primitive, which we call *somewhat non-committing* encryption, where the simulator is given a set of ℓ messages during the generation of the fake ciphertext and must later be able to plausibly explain the ciphertext as the encryption of *any one of those* ℓ *messages*. In a sense, we distinguish between two parameters: the plaintext size (in bits) k and the equivocality ℓ (the number of messages that the simulator can plausibly explain). For fully non-committing encryption, the equivocality and the message size are related by $\ell = 2^k$. Somewhat non-committing encryption, on the other hand, is useful in accommodating the case where the equivocality ℓ is very small, but the message size k is large.

Functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}}$

The ideal functionality $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}}$ interacts with an initiator I and a receiver R. It consists of a channel-setup phase, after which the two parties can send arbitrarily many messages from one to another. The functionality is parameterized by a non-information oracle $\mathcal{N}$.

Channel setup: Upon receiving $(\texttt{ChSetup}, sid, I)$ from party I, initialize the machine $\mathcal{N}$ and record the tuple $(sid, \mathcal{N})$. Pass the message $(\texttt{ChSetup}, I)$ to R. In addition, pass this message to $\mathcal{N}$ and forward its output to the adversary $\mathcal{S}$.

Message transfer: Upon receiving $(\texttt{Send}, sid, P, m)$ from party P where $P \in \{I, R\}$, find a tuple $(sid, \mathcal{N})$ and, if none exists, ignore the message. Otherwise, send the message $(\texttt{Send}, sid, P, m)$ to the *other* party $\overline{P} = \{I, R\} - \{P\}$. In addition, invoke $\mathcal{N}$ with $(\texttt{Send}, sid, P, m)$ and forward its output to $\mathcal{S}$.

Corruption: Upon receiving a message $(\texttt{Corrupt}, sid, P)$ from the adversary, send $(\texttt{Corrupt}, sid, P)$ to $\mathcal{N}$ and forward its output to $\mathcal{S}$. After the first corruption, stop the execution of $\mathcal{N}$ and give $\mathcal{S}$ complete control over the functionality (i.e., $\mathcal{S}$ learns all inputs and can specify any outputs).

Fig. 1. The parameterized secure-channel ideal functionality, $\mathcal{F}_{\mathrm{SC}}^{\mathcal{N}}$

It is challenging to define an ideal-functionality for *somewhat* non-committing encryption, since the ideal world captures a notion of security which is too strong. Here, we take the approach of [7] where ideal-world functionalities are weakened by the inclusion of a *non-information oracle* which is a PPT TM that captures the information leaked to the adversary in the ideal world. The ideal functionality for secure channels, given in Figure 1, is parameterized using a non-information oracle $\mathcal{N}$ which gets the values of the exchanged messages m and outputs some side information to the adversary $\mathcal{S}$. The security of the secure channel functionality $\mathcal{F}_{\text{SC}}^{\mathcal{N}}$ depends on the security properties required for the machine $\mathcal{N}$ and thus we can capture several meaningful notions. Let us first start with the most secure option which captures (fully) non-committing encryption.

Definition 1. *Let* $\mathcal{N}^{\text{full}}$ *be the oracle, which, on input* $(\texttt{Send}, sid, P, m)$*, produces the output* $(\texttt{Send}, sid, P, |m|)$ *and, on any inputs corresponding to the* `ChSetup`, `Corrupt` *commands, produces no output. We call the functionality* $\mathcal{F}_{\text{SC}}^{\mathcal{N}^{\text{full}}}$*, or just* $\mathcal{F}_{\text{SC}}$ *for brevity, a* (fully) non-committing secure channel*. A real-world protocol which realizes* $\mathcal{F}_{\text{SC}}$ *is called a* non-committing encryption scheme (NCE).

Above, the oracle $\mathcal{N}$ never reveals anything about messages m exchanged by two honest parties, even if (both of the) parties later get corrupted. Hence the functionality is fully *non-committing*. To define *somewhat* non-committing encryption we first give the following definitions of non-information oracles.

Definition 2. *A machine* $\mathcal{R}$ *is called a* message-ignoring *oracle if, on any input* $(\texttt{Send}, sid, P, m)$*, it ignores the value* m *and processes only the input* $(\texttt{Send}, sid, P, |m|)$*. A machine* $\mathcal{M}$ *called a* message-processing *oracle if it has no such restrictions. We call a pair of machines* $(\mathcal{M}, \mathcal{R})$ well-matched *if no PPT distinguisher* $\mathcal{D}$ *(with oracle access to either* $\mathcal{M}$ *or* $\mathcal{R}$*) can distinguish the message-processing oracle* $\mathcal{M}$ *from the message-ignoring oracle* $\mathcal{R}$*.*

We are now ready to define the non-information oracle used by a somewhat non-committing secure channel ideal functionality.

Definition 3. *Let* $(\mathcal{M}, \mathcal{R})$ *be a well-matched pair which consists of a message-processing and a message-ignoring oracle respectively. Let* $\mathcal{N}^{\ell}$ *be a (stateful) oracle with the following structure.*

- *Upon initialization,* $\mathcal{N}^{\ell}$ *chooses a uniformly random index* $i \xleftarrow{\$} \{1, \ldots, \ell\}$*. In addition it initializes a tuple of* ℓ independent *TMs:* $\langle \mathcal{N}_1, \ldots, \mathcal{N}_{\ell} \rangle$ *where* $\mathcal{N}_i = \mathcal{M}$ *and, for* $j \neq i$*, the machines* $\mathcal{N}_j$ *are independent copies of the message-ignoring oracle* $\mathcal{R}$*.*
- *Whenever* $\mathcal{N}^{\ell}$ *receives inputs of the form* $(\texttt{ChSetup}, sid, P)$ *or* $(\texttt{Send}, sid, P, m)$*, it passes the input to each machine* $\mathcal{N}_i$ *receiving an output* y_i*. It then outputs the vector* $(y_1, \ldots, y_{\ell})$*.*
- *Upon receiving an input* $(\texttt{Corrupt}, sid, P)$*, the oracle reveals the internal state of the message-processing oracle* $\mathcal{N}_i$ *only.*

For any such oracle $\mathcal{N}^\ell$, we call the functionality $\mathcal{F}_{\text{SC}}^{\mathcal{N}^\ell}$ an ℓ-equivocal non-committing secure channel. *For brevity, we will also use the notation $\mathcal{F}_{\text{SC}}^{\ell}$ to denote $\mathcal{F}_{\text{SC}}^{\mathcal{N}^\ell}$ for some such oracle $\mathcal{N}^\ell$. Lastly, a real world protocol which realizes $\mathcal{F}_{\text{SC}}^{\ell}$ is called an* ℓ-equivocal non-committing encryption scheme (ℓ-NCE).

As before, no information about messages m is revealed during the "send" stage. However, the internal state of the message-processing oracle $\mathcal{N}_i$, which is revealed upon corruption, might be "committing." Nevertheless, a simulator can simulate the communication between two honest parties over a secure channel, as modeled by $\mathcal{F}_{\text{SC}}^{\ell}$, in a way that allows him to later explain this communication as any one of ℓ possibilities. In particular, the simulator creates ℓ message-processing oracles and, for every `Send` command, the simulator chooses ℓ distinct messages $m_1, \ldots, m_\ell$ that he passes to the oracles $\mathcal{M}_1, \ldots, \mathcal{M}_\ell$ respectively. Since message-processing and message-ignoring oracles are indistinguishable, this looks indistinguishable from the side information produced by $\mathcal{F}_{\text{SC}}^{\ell}$. Later, when a corruption occurs, the simulator can convincingly explain the entire transcript of communication to any one of the ℓ possible options, by providing the internal state of the appropriate message-processing oracle $\mathcal{M}_i$.

2.3 The ℓ-NCE Scheme Construction

The construction of ℓ-NCE is based on a *simulatable public-key system* [11], wherein it is possible to generate public keys obliviously, without knowing the corresponding secret key, and to explain an honestly (non-obliviously) generated public key as one which was obliviously generated. In a similar way, there should be a method for obliviously generating ciphertexts (without knowing any plaintext) and to explain honestly generated (non-oblivious) ciphertexts as obliviously generated ones. Refer to the full version for review of the syntax and security properties of such a scheme. Our ℓ-NCE protocol construction, shown in Figure 2, uses a fully non-committing secure channel, but only to send a *very short* message during the setup phase. In addition, it uses a simulatable public-key system and a symmetric key encryption scheme where ciphertexts are indistinguishable from uniformly random values (the latter can be constructed from any one way function). For very long communications and small ℓ, our ℓ-NCE scheme is significantly more efficient than (full) NCE.

Theorem 1. *The protocol in Figure 2 is an ℓ-NCE scheme. Specifically, it UC-realizes functionality $\mathcal{F}_{\text{SC}}^{\ell}$ in the presence of an active and adaptive adversary.*

The main efficiency consideration is the use of fully non-committing encryption of the index i (which is small). We show in the full version that our scheme uses a total of expected $\mathcal{O}(\log \ell)$ public-key operations, expected $\mathcal{O}(\ell\lambda)$ communication and expected constant rounds of interaction for the channel setup phase, where λ is the security parameter. Alternatively, if one would like to set up $n = \Omega(\lambda)$ channels in parallel, this can be done in strict $\mathcal{O}(n \log \ell)$ public-key operations, strict $\mathcal{O}(n\ell\lambda)$ communication and strict constant number of rounds of interaction. After channel-setup, encryption is non-interactive and requires only

Let $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a *simulatable public-key system* and $\widetilde{\mathsf{KG}}$, $\widetilde{\mathsf{Enc}}$ be the corresponding *oblivious key generator* and *oblivious ciphertext generator* algorithms. Further, let $(\mathsf{KG}^{\mathsf{sym}}, \mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ be a symmetric-key encryption scheme in which ciphertexts are indistinguishable from uniformly random values of the same length.

Channel Setup. An initiator I sets up a channel with a receiver R as follows:

1. The initiator I sends a random index $i \in \{1, \ldots, \ell\}$ to R over a fully non-committing secure channel.
2. The initiator I generates ℓ public keys. For $j \in \{1, \ldots, \ell\} \setminus \{i\}$, the keys $pk_j \leftarrow \widetilde{\mathsf{KG}}()$ are sampled obliviously, while $(pk_i, sk_i) \leftarrow \mathsf{KG}()$ is sampled correctly. The keys $pk_1, \ldots, pk_\ell$ are sent to R while I stores sk_i.
3. The receiver R chooses a random key $K \leftarrow \mathsf{KG}^{\mathsf{sym}}$ and computes $C_i = \mathsf{Enc}_{pk_i}(K)$ correctly. In addition, R samples $C_j \leftarrow \widetilde{\mathsf{Enc}}_{pk_j}()$ obliviously for $j \in \{1, \ldots, \ell\} \setminus \{i\}$ and sends the ciphertexts $C_1, \ldots, C_\ell$ to I.
4. The initiator I decrypts the key $K \leftarrow \mathsf{Dec}_{sk_i}(C_i)$. Both parties store (K, i).

Encryption. An initiator I encrypts a message m to a receiver R as follows:

1. The initiator I computes $E_i \leftarrow \mathsf{Enc}^{\mathsf{sym}}_K(m)$ and chooses E_j for $j \in \{1, \ldots, \ell\} \setminus \{i\}$ as uniformly random and independent values of length $|E_i|$. The tuple $(E_1, \ldots, E_\ell)$ is sent to R.
2. The receiver R ignores all values other than E_i. It computes $m \leftarrow \mathsf{Dec}^{\mathsf{sym}}_K(E_i)$.

Fig. 2. The ℓ-NCE protocol

symmetric-key operations. However, the encryption of a k bit message requires $\mathcal{O}(\ell k)$ bits of communication.

2.4 The Adaptive Security Protocol Compiler for Two-Party SFE

As an application of ℓ-NCE, we now give a general theorem showing that a protocol with semi-adaptive security can be compiled into a protocol with (full) adaptive security when all of the communication is encrypted using ℓ-NCE for some appropriate ℓ. However, we must first give a formal definition of semi-adaptive security.

Definition 4. *An adversarial strategy is second-corruption adaptive if* either *at least one of the parties is corrupted prior to protocol execution* or *no party is ever corrupted. In the former case, the other party can be adaptively corrupted at any point during or after protocol execution.*

Intuitively, we would like to say that a protocol is semi-adaptively secure if it is secure with respect to second-corruption adaptive strategies. Unfortunately, there are two subtleties that we must consider. Firstly, we know that most tasks cannot be realized in the Universal Composability framework without the use of *trusted setup*. However, the use of trusted setup complicates our transformation. The point of using (somewhat) non-committing encryption is that the simulator

can lie about *anything that occurs while both parties are honest*. However, we often rely on trusted setup in which some information is given to the adversary even when both parties are honest. For example, the usual modeling of a common reference string specifies that this string is made public and given to the adversary even when *none* of the participants in the protocol are corrupted. In this case the simulator is committed to such setup even if the parties communicate over secure channels. Therefore we require that, when trusted setup is used, the semi-adaptive simulator simulates this setup independently of which party is corrupted. We call this property *setup-adaptive simulation*.

The second subtlety comes from the following type of problem. As we outlined in our informal discussion, we wish to run the semi-adaptive simulator once the first party gets corrupted and then "lie" that the simulated conversation took place over the secure channel. However, when the first party gets corrupted after the protocol execution, then the ideal functionality has already computed the outputs using the honest inputs and will therefore not accept anymore inputs from the semi-adaptive simulator. Recall that we run the semi-adaptive simulator with respect to an adversary $\mathcal{A}$ which follows the protocol execution using the corrupted party's honest input x. If the semi-adaptive simulator extracts the same input x as the one used by $\mathcal{A}$, then we also know the corresponding output and can give it to the semi-adaptive simulator on behalf of the ideal functionality. Therefore it is crucial that the semi-adaptive simulator can only submit the actual input x. We call this property *input-preserving*. Putting Definition 4 and the above notions together, we are finally ready to define semi-adaptive security.

Definition 5. *We say that a protocol π semi-adaptively realizes the ideal functionality $\mathcal{F}$ if there exists a setup-adaptive and input-preserving PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$ and environment $\mathcal{Z}$ which follow a second-corruption adaptive adversarial strategy, we have $REAL_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.*

Lastly, we define the notion of a *well-structured* protocol. Since even non-committing encryption *commits* the simulator to the lengths of the exchanged messages, the number of such messages, and the identities of the sender and receiver of each message, we require that this information is fixed and always the same any given execution of a protocol. Almost all known constructed protocols for cryptographic tasks are well-structured and any protocol can be easily converted into a well-structured protocol.

First we look at the simple compiler using idealized secure channels.

Theorem 2. *Let $\mathcal{F}_{\mathrm{SFE}}^{f}$ be the two-party ideal functionality which computes some function f as defined in Figure 3. Assume that a well-structured two-party protocol π for $\mathcal{F}_{\mathrm{SFE}}^{f}$ is semi-adaptively secure. Let π' be the protocol in which the parties run π but only communicate with each other using non-committing secure channels as modeled by $\mathcal{F}_{\mathrm{SC}}$. Then π' is (fully) adaptively secure.*

As we already mentioned, this compiler is usually not very efficient because of its excessive use of secure channels and hence NCE. Recall that secure channels are employed so that, when both parties are honest, the adversary does not see

Functionality $\mathcal{F}_{\text{SFE}}^{f}$

The functionality $\mathcal{F}_{\text{SFE}}^{f}$ interacts with an *initiator* I and a *responder* R.

Input: Upon receiving the input value $(\texttt{Input}_I, sid, x_I)$ from the initiator I, record the value $\langle I, x_I \rangle$ and send the message $(\texttt{Input}_I, sid)$ to the adversary $\mathcal{S}$. Ignore future $(\texttt{Input}_I, \ldots)$ inputs. Similarly, upon receiving the input value $(\texttt{Input}_R, sid, x_R)$ from the responser R, record the value $\langle R, x_R \rangle$ and send the message $(\texttt{Input}_R, sid)$ to the adversary $\mathcal{S}$. Ignore future $(\texttt{Input}_R, \ldots)$ inputs.

Output: Upon receiving the message $(\texttt{Output}_I, sid)$ from the adversary $\mathcal{S}$, if either $\langle I, x_I \rangle$ or $\langle R, x_R \rangle$ is not recorded, ignore the message. Else if $\langle y_I, y_R \rangle$ is not recorded, then compute $(y_I, y_R) \leftarrow f(x_I, x_R)$ and record $\langle y_I, y_R \rangle$; send the output value $(\texttt{Output}_I, sid, y_I)$ to I. Ignore future $(\texttt{Output}_I, \ldots)$ messages from the adversary. Similarly, upon receipt of $(\texttt{Output}_R, sid)$ from the adversary, send the output value $(\texttt{Output}_R, sid, y_R)$ to R. Ignore future $(\texttt{Output}_R, \ldots)$ messages from the adversary.

Fig. 3. Two-party secure evaluation functionality for $f : X_I \times X_R \to Y_I \times Y_R$

any useful information and so this case is easy to simulate. Then, when the first party gets corrupted, our simulator simply makes up the transcript of the communication that should have taken place *ex post facto*. This transcript is generated based on which party got corrupted, what its inputs were and what its outputs were. However, we notice that for many simple protocols there are not too many choices for this information. The simulator must simply be able to credibly lie that the communication which took place over the secure channel corresponds to any one of these possible choices. Using this intuition, we show that a more efficient compiler using ℓ-NCE (for some small ℓ) suffices.

Theorem 3. *Let $\mathcal{F}_{\text{SFE}}^{f}$ be the two-party ideal functionality computing some function $f : X_I \times X_R \to Y_I \times Y_R$, as defined in Figure 3. Assume that a well-structured two-party protocol π for $\mathcal{F}_{\text{SFE}}^{f}$ is semi-adaptively secure. Let π' be the protocol in which the parties run π but only communicate with each other using ℓ-equivocal secure channels as modeled by $\mathcal{F}_{\text{SC}}^{\ell}$ where $\ell = |X_I||Y_I| + |X_R||Y_R|$. Then π' is (fully) adaptively secure.*

3 Efficient and Adaptively Secure Oblivious Transfer

We now apply our compiler of Theorem 3 to the concrete problem of bit- and string-OT, resulting in the first efficient protocols for this task. Refer to [4,15] for the specification of an ideal functionality for OT.

3.1 The PVW Oblivious Transfer Protocol

In [28], Peikert *et al.* construct an efficient OT protocol in the CRS model with UC security against a malicious but static adversary. They do so by introducing a new primitive called a *dual-mode cryptosystem*, which almost immediately yields

an OT protocol in the CRS model, and give constructions of this primitive under the DDH, QR and lattice hardness assumptions. We first present a brief review of dual-mode encryption as in [28], and then will define a modified version of this primitive which will allow us to get adaptive security.

A dual-mode cryptosystem is initialized with *system parameters* which are generated by a trusted third party. For any choice of system parameters, the cryptosystem has two types of public/private key pairs: *left* key pairs and *right* key pairs. The key-generation algorithm can sample either type of key pair and the user specifies which type is desired. Similarly, the encryption algorithm can generate a *left* encryption or a *right* encryption of a message. When the key pair type matches the encryption type (i.e. a left encryption of a message under a left public key) then the decryption algorithm (which uses the matching secret key) correctly recovers the message.

As shown in [28], a dual-mode cryptosystem can be used to get an OT protocol, as follows. The receiver chooses to generate a left or right key depending on his input bit σ, and the sender uses left-encryption ($b = 0$) for the left message x_0 and right-encryption for the right message. The receiver then uses the secret key to correctly decrypt the chosen message.

Security against malicious (static) adversaries in the UC model relies on the two different modes for generating the system parameters: *messy* mode and *decryption* mode. In *messy mode*, the system parameters are generated together with a *messy trapdoor*. Using this trapdoor, any public key (even one which is maliciously generated) can be easily labeled a left key or a right key. Moreover, in messy mode, when the encryption type does not match the key type (e.g., a left encryption using a right public key) then the ciphertext is statistically independent of the message. Messy mode is useful to guarantee security against a *corrupt receiver:* the messy trapdoor makes it easy to extract the receiver bit and to create a fake ciphertext for the message which should not be transferred. On the other hand, in *decryption mode*, the system parameters are generated together with a *decryption trapdoor* which can be used to decrypt both left and right ciphertexts. Moreover, in decryption mode, left public keys are statistically indistinguishable from right public keys. Decryption mode is useful to guarantee security against a *corrupt sender:* the decryption trapdoor is used to create a public key which completely hides the receiver's selection bit, and to compute a decryption trapdoor and extracting both of the sender's messages. In each mode, the security of one party (i.e., the sender in messy mode, and the receiver in decryption mode) is guaranteed information theoretically. To achieve security for both parties simultaneously all that is needed is one simple computational requirement: the system parameters generated in messy mode need to be computationally indistinguishable from those generated in decryption mode.

3.2 Semi-adaptively Secure OT

In order to make the PVW OT protocol adaptively secure using our methodology, we need to make it semi-adaptively secure (Section 2.4). We do so by a series of simple transformations.

First, we observe that in the PVW protocol, the simulator must choose the CRS crs_{ot} based on which party is corrupt – i.e. the CRS should be in messy mode to handle a corrupt receiver or in decryption mode to handle a corrupt sender. This is a problem for us since the definition of semi-adaptive security requires that the simulator be setup-adaptive which means that it must simulate the CRS independently of any information on which parties are corrupted. We solve this issue by using a coin-tossing protocol to choose the CRS of the PVW OT protocol. Of course, coin-tossing requires the use of a UC secure commitment scheme which also needs its own CRS (crs_{com})! However, if we use an (efficient) adaptively secure commitment scheme (e.g., [12,10]) then the simulator's choice of crs_{com} can be independent of which party is corrupted. Unfortunately, this approach only works if the CRS for the OT protocol comes from a uniform distribution (over some group) and this also is not the case in all instantiations of the PVW protocol. However, we observe that the CRS of the OT protocol (crs_{ot}) can be divided into two parts $crs_{\text{ot}} = (crs_{\text{sys}}, crs_{\text{tmp}})$, where a *system CRS* crs_{sys} can be independent of which party is corrupted (i.e., can be the same for both messy and decryption modes) but may not be uniform, while crs_{tmp} determines the mode and thus needs to depend on which party is corrupted, but this part is required to be uniform. Therefore we can use an ideal CRS functionality to choose the setup for our protocol which consists of $(crs_{\text{com}}, crs_{\text{sys}})$ and then run a coin-flipping protocol to choose the uniform crs_{tmp}.

Secondly, we must now consider the cases where one party is corrupted from the beginning, but the second party becomes corrupted adaptively during the protocol execution. Let us first consider the case where the sender starts out corrupted. In this case, to handle the corrupt sender, the simulator needs to simulate the execution in decryption mode. Moreover, to extract the sender's value, the simulator uses the decryption trapdoor to create a *dual public key* (on behalf of the receiver) which comes with both a left and a right secret key. Later, if the receiver becomes corrupted, the simulator needs to explain the randomness used by the receiver during key generation to create such a public key. Luckily, current dual-mode schemes already make this possible and we just update the definition with a property called *encryption key duality* to capture this.

Now, consider the case where the receiver is corrupted at the beginning but the sender might *also* become corrupted later on. In this case the simulator simulates the execution in messy mode. In particular, the simulator uses the messy trapdoor to identify the receiver key type (right or left) and thus extracts the receiver bit. Then the simulator learns the appropriate sender message for that bit and (honestly) produces the ciphertext for that message. In addition, the simulator must produce a "fake" ciphertext for the other message. Since, in messy mode, this other ciphertext is statistically independent of the message, it is easy to do so. However, if the sender gets corrupted later, the simulator must *explain* the fake ciphertext as an encryption of some particular message. To capture this ability, we require the existence of *internal state reconstruction* algorithm which can explain the fake ciphertext as an encryption of any message.

Again, we notice that the QR instantiation of the PVW scheme already satisfies this new notion as well.

We now specify our enhanced version of dual-mode encryption in more detail. Here we just describe the added features with respect to [28]; refer to [15] for the full description.

Enhanced Dual-Mode Encryption. A dual-mode cryptosystem for message space $\{0,1\}^n$ is defined by the following polynomial-time algorithms:

- $(crs, \tau) \leftarrow \mathsf{PG}(1^\lambda, \mu)$. The parameter generation algorithm PG is a randomized algorithm which takes security parameter λ and mode $\mu \in \{\mathrm{mes}, \mathrm{dec}\}$ as input, and outputs (crs, τ), where crs is a common reference string and τ is the corresponding trapdoor information. Note that PG includes two stages, $\mathsf{PG}_{\mathrm{sys}}$ and $\mathsf{PG}_{\mathrm{tmp}}$, i.e., compute $(G, crs_{\mathrm{sys}}, \tau_{\mathrm{sys}}) \leftarrow \mathsf{PG}_{\mathrm{sys}}(1^\lambda)$ and $(crs_{\mathrm{tmp}}, \tau_{\mathrm{tmp}}) \leftarrow \mathsf{PG}_{\mathrm{tmp}}(\mu, G, crs_{\mathrm{sys}}, \tau_{\mathrm{sys}})$ where G is a group with operator "+", and set $crs \leftarrow (crs_{\mathrm{sys}}, crs_{\mathrm{tmp}})$ and $\tau \leftarrow (\tau_{\mathrm{sys}}, \tau_{\mathrm{tmp}})$. Also note that the system CRS is independent of mode μ.
- $(pk, sk) \leftarrow \mathsf{KG}(crs, \sigma)$; $(c, \zeta) \leftarrow \mathsf{Enc}(crs, pk, b, m)$; $m \leftarrow \mathsf{Dec}(crs, pk, sk, c)$; and $\rho \leftarrow \mathsf{MessyId}(crs, \tau, pk)$ as in [28].
- $(c, \omega) \leftarrow \mathsf{FakeEnc}(crs, \tau, pk, \rho)$. The fake encryption algorithm $\mathsf{FakeEnc}$ is a randomized algorithm. For the messy branch ρ, the ciphertext c is faked by using the trapdoor τ, and some internal information ω is saved for reconstructing the random coins used for encryption.
- $\zeta \leftarrow \mathsf{Recons}(crs, \tau, pk, \rho, c, \omega, m)$. The internal state reconstruction algorithm Recons is a deterministic algorithm. When the plaintext m is supplied for the faked ciphertext c in messy branch ρ, the algorithm recovers the used random coins ζ based on previously generated internal information ω.
- $(pk, sk_0, sk_1) \leftarrow \mathsf{DualKG}(crs, \tau)$. The dual key generation algorithm DualKG is a randomized algorithm, which based on the trapdoor τ, outputs an encryption key pk, and two decryption keys sk_0, sk_1 corresponding to key type 0 and 1, respectively.

Definition 6 (*Enhanced* Dual-Mode Encryption). *An* enhanced dual-mode cryptosystem *is a tuple of algorithms as described above satisfying the following properties:*

- Completeness *as in* [28].
- Enhanced mode indistinguishability: *The CRSes generated by* PG *in messy mode and in decryption mode are indistinguishable in the sense that (i) the both system CRSes are identically distributed, and (ii) the two temporal CRSes are computationally indistinguishable from random elements in group* G.
- Messy branch identification and ciphertext equivocation: *For every* $(crs, \tau) \leftarrow \mathsf{PG}(1^\lambda, \mathrm{mes})$ *and every* pk, $\mathsf{MessyId}(crs, \tau, pk)$ *outputs a branch value* ρ *such that for every* $m \in \{0,1\}^n$, $\mathsf{Enc}(crs, pk, \rho, \cdot)$ *is simulatable.*

– Encryption key duality: *For every* $(crs, \tau) \leftarrow \mathsf{PG}(1^\lambda, \mathrm{dec})$*, there exists* $(pk, sk_0, sk_1) \leftarrow \mathsf{DualKG}(crs, \tau)$ *such that for every* $\sigma \in \{0,1\}$*,* (pk, sk_σ) *is statistically indistinguishable from the honestly generated key pair.*

Construction. Based on the above transformations, a generic construction for a semi-adaptively secure OT protocol is given in Figure 4. It consists of two phases, the coin tossing phase and the transferring phase (which is separated by a dot line in the figure). The CRS consists of two pieces: the first piece is a system CRS denoted as crs_{sys}, while the second piece is for an adaptively secure UC commitment protocol which will be used for constructing a coin tossing protocol. The UC commitment includes two stages, the commit (to a randomly selected value r by the receiver) and the open stages, which could be interactive, and is used to compute a temporal CRS crs_{tmp}. crs_{tmp} together with the system CRS crs_{sys} are used as the CRS for the transferring phase and we denote it as crs_{ot}. With crs_{ot} in hand, we "plug in" the PVW protocol, but based on the enhanced dual-mode cryptosystem to achieve message transferring.

Theorem 4. *Given an adaptively UC-secure commitment scheme and an enhanced dual-mode cryptosystem as in Definition 6, the protocol in Figure 4 semi-adaptively realizes* $\mathcal{F}_{\mathrm{OT}}$ *in the* $\mathcal{F}_{\mathrm{CRS}}$*-hybrid model.*

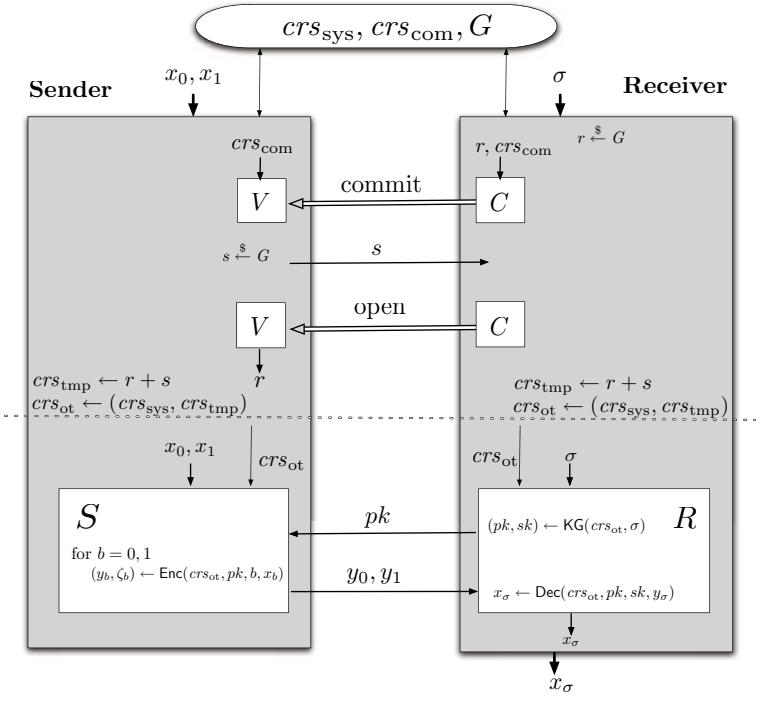

Fig. 4. Generic semi-adaptively secure OT protocol

By "plugging in" efficient instantiations of the two building blocks above, we obtain efficient concrete protocols for semi-adaptively secure OT. For example, good candidates for adaptively secure UC commitments can be found in [10,12], while a QR-based dual-mode encryption scheme is presented in [28]; in the full version we show that this scheme also satisfies Definition 6. As mentioned in Section 1, a semi-adaptively secure OT protocol can also be based on the DDH assumption. In this case, however, in order to make ciphertext equivocation possible, we also need an efficient Σ-protocol for the equality of discrete logs.

3.3 Efficient and Adaptively Secure OT

We now apply our compiler from Section 2.4 to the protocol in Figure 4, to immediately obtain efficient adaptively secure OT protocols in the UC framework.

Corollary 5. *Assume that the DDH, QR, and DCR assumptions hold. Then there exists an adaptively secure protocol that UC-realizes the* bit-*OT functionality $\mathcal{F}_{\text{OT}}$ in the $\mathcal{F}_{\text{CRS}}$-hybrid world, running in (expected) constant number of rounds and using (expected) constant number of public-key operations.*

Justification for the assumptions is as follows: efficient adaptive UC commitments can be realized in the CRS model under the DCR assumption [12], non-committing and somewhat non-committing encryption can be constructed under DDH ([11] and Section 2, respectively), while enhanced dual-model encryption exists under the QR assumption ([28] and Section 3.2).

In the full version we also show how to instantiate our framework using the DDH version of the PVW protocol, resulting in an efficient bit-OT protocol with similar parameters to the one above based on DCR. Further, we also show how to use the DDH version of PVW to also efficiently implement *string*-OT. This involves the semi-adaptively secure realization of an enhanced, *receiver-committed* version of bit-OT, where the receiver is also committed to his bit under an equivocal commitment scheme (e.g., a Pedersen commitment), as well as a generalization of our compiler; lastly, we use several copies of the enhanced bit-OT functionality to construct string-OT for long strings. (See [15] for details.) This strategy yields the following theorem.

Theorem 6. *Assume that the DDH and DCR assumptions hold. Then there exists an adaptively secure protocol that UC-realizes the* string-OT *functionality $\mathcal{F}_{\text{OT}}$ in the $\mathcal{F}_{\text{CRS}}$-hybrid world, and can transfer an n-bit string in (strict) constant number of rounds and using (strict) $O(n)$ public-key operations.*

Acknowledgements. We thank Ran Canetti, Yevgeniy Dodis, Yuval Ishai, Stas Jarecki, and Aggelos Kiayias for useful discussions. We also thank the anonymous referees for their constructive comments.

References

1. Beaver, D.: Plug and play encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997)
2. Beaver, D.: Adaptively secure oblivious transfer. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 300–314. Springer, Heidelberg (1998)
3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (December 2005)
5. Canetti, R., Damgård, I., Dziembowski, S., Ishai, Y., Malkin, T.: Adaptive versus non-adaptive security of multi-party protocols. J. Cryptology 17(3), 153–207 (2004)
6. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
7. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503. ACM, New York (2002)
9. Choi, S.G., Dachaman-Soled, D., Malkin, T., Wee, H.: Simple, black-box constructions of adaptively secure protocols. In: Reingold, O. (ed.) TCC. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
10. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: STOC, pp. 426–437 (2003)
11. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
12. Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
13. Damgård, I., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
14. Damgård, I., Nielsen, J.B., Orlandi, C.: Essentially optimal universally composable oblivious transfer. In: ICISC, pp. 318–335 (2008)
15. Garay, J., Wichs, D., Zhou, H.-S.: Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. Cryptology ePrint Archive: Report 2008/534 (2008)
16. Garay, J.A., MacKenzie, P., Yang, K.: Efficient and universally composable committed oblivious transfer and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 297–316. Springer, Heidelberg (2004)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
18. Haitner, I.: Semi-honest to malicious oblivious transfer – the black-box way. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008)
19. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: STOC, pp. 99–108. ACM, New York (2006)

20. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
21. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
22. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
23. Lindell, A.Y.: Efficient fully-simulatable oblivious transfer. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008)
24. Lindell, Y.: Adaptively secure two-party computation with erasures. In: Fischlin, M. (ed.) CT-RSA. LNCS, vol. 5473, pp. 117–132. Springer, Heidelberg (2009)
25. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
26. Lindell, Y., Zarosim, H.: Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. In: Reingold, O. (ed.) TCC. LNCS, vol. 5444, pp. 183–201. Springer, Heidelberg (2009)
27. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
28. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)