Improved Non-Committing Encryption with Applications to Adaptively Secure Protocols

Seung Geol Choi[1⋆], Dana Dachman-Soled[1⋆], Tal Malkin[1⋆], and Hoeteck Wee[2⋆⋆]

[1] Columbia University. {sgchoi,dglasner,tal}@cs.columbia.edu
[2] Queens College, CUNY. hoeteck@cs.qc.cuny.edu

Abstract. We present a new construction of non-committing encryption schemes. Unlike the previous constructions of Canetti et al. (STOC '96) and of Damgård and Nielsen (Crypto '00), our construction achieves all of the following properties:

- **Optimal round complexity.** Our encryption scheme is a 2-round protocol, matching the round complexity of Canetti et al. and improving upon that in Damgård and Nielsen.
- **Weaker assumptions.** Our construction is based on *trapdoor simulatable cryptosystems*, a new primitive that we introduce as a relaxation of those used in previous works. We also show how to realize this primitive based on hardness of factoring.
- **Improved efficiency.** The amortized complexity of encrypting a single bit is $O(1)$ public key operations on a constant-sized plaintext in the underlying cryptosystem.

As a result, we obtain the first non-committing public-key encryption schemes under hardness of factoring and worst-case lattice assumptions; previously, such schemes were only known under the CDH and RSA assumptions. Combined with existing work on secure multi-party computation, we obtain protocols for multi-party computation secure against a malicious adversary that may adaptively corrupt an arbitrary number of parties under weaker assumptions than were previously known. Specifically, we obtain the first adaptively secure multi-party protocols based on hardness of factoring in both the stand-alone setting and the UC setting with a common reference string.

Key words: public-key encryption, adaptive corruption, non-committing encryption, secure multi-party computation.

1 Introduction

Secure multi-party computation (MPC) allows several mutually distrustful parties to perform a joint computation without compromising, to the greatest

⋆ supported in part by NSF Grants CCF-0347839, CNS-0716245, CNS-0831094 and SBE-0245014.

⋆⋆ partially supported by a PSC-CUNY Award, and part of this work was done while a post-doc at Columbia University.

extent possible, the privacy of their inputs or the correctness of the outputs. An important criterion in evaluating the security guarantee is *how many* parties an adversary is allowed to corrupt and *when* the adversary determines which parties to corrupt. Ideally, we want to achieve the strongest notion of security, namely, against an adversary that corrupts an arbitrary number of parties, and *adaptively* determines who and when to corrupt during the course of the computation (and without assuming erasures[3]). Even though the latter is a very natural and realistic assumption about the adversary, most of the MPC literature only addresses security against a static adversary, namely one that chooses (and fixes) which parties to corrupt before the protocol starts executing. And if indeed such protocols do exist, it is important to answer the following question:

> What are the cryptographic assumptions under which we can realize MPC protocols secure against a malicious, adaptive adversary that may corrupt a majority of the parties?

Towards answering this question, we revisit the problem of constructing *non-committing encryption schemes*, a cryptographic primitive first introduced by Canetti et al. [CFGN96] as a tool for building adaptively secure MPC protocols in the presence of an honest majority. Informally, non-committing encryption schemes are semantically secure, possibly interactive encryption schemes, with the additional property that a simulator can generate special ciphertexts that can be opened to both a 0 and a 1. In a more recent work, Canetti et al. [CLOS02] (extending [B98]) showed how to construct adaptively secure oblivious transfer protocols starting from non-committing public-key encryption schemes (i.e. the key generation algorithm must be non-interactive), which may in turn be used to construct MPC protocols secure against a malicious, adaptive adversary that may corrupt an arbitrary number of parties.

Unfortunately, the only known constructions of non-committing public-key encryption schemes (PKEs) are based on the CDH and RSA assumptions [CFGN96] and the construction exploits in a very essential way that these assumptions give rise to families of trapdoor permutations with a common domain. If we allow for an interactive key generation phase, Damgård and Nielsen [DN00], building on [B97,CFGN96], constructed 3-round non-committing encryption schemes based on a more general assumption, that of *simulatable PKEs*, which may in turn be realized from DDH, CDH, RSA and more recently, worst-case lattice assumptions [GPV08] (see figure 1).

[3] Refer to [C00, Section 5.2] for a discussion on how trusted erasures may be a problematic assumption.

1.1 Our results

First, we present a new construction of non-committing encryption schemes, which simultaneously improves upon all of the previous constructions in [CFGN96,DN00]:

Optimal Round Complexity. We provide a construction of non-committing PKEs from simulatable cryptosystems. Our construction is surprisingly simple - a twist to the standard cut-and-choose techniques used in [DN00,KO04] - and also admits a fairly straight-forward simulation and analysis. In particular, our construction and the analysis are conceptually and technically simpler than those in [CFGN96,DN00]; we avoid having to analyze the number of one's in certain Binomial distributions as in [CFGN96] and to consider a subtle failure mode as in [DN00].

Reducing the assumptions. Informally, a simulatable PKE is an encryption scheme with special algorithms for obliviously sampling public keys and random ciphertexts without learning the corresponding secret keys and plaintexts; in addition, both of these oblivious sampling algorithms should be efficiently invertible.

We define a weaker assumption, which we refer to as trapdoor simulatable cryptosystems, and prove that it is sufficient for our construction and analysis to go through. Roughly speaking, we provide the inverting algorithms in a simulatable cryptosystem with additional trapdoor information (hence the modifier "trapdoor"), which makes it *easier* to design a simulatable cryptosystem.

Improved efficiency. While the main focus of this work is feasibility results (notably, reducing the computational assumptions for both non-committing encryption schemes and adaptively secure MPC), we show how to combine a variant of our basic construction with the use of error-correcting codes to achieve better efficiency. That is, the amortized complexity of encrypting a single bit is $O(1)$ public-key operations on a constant-sized plaintext in the underlying cryptosystem.

Thus, we obtain the following.

Theorem 1 (informal). *There exists a black-box construction of a non-committing public-key encryption scheme, starting from any trapdoor simulatable cryptosystem.*

Factoring-based constructions. Next, we derive trapdoor simulatable cryptosystems from a variant of Rabin's trapdoor permutations (c.f. [H99,S96,FF02]) based on the hardness of factoring Blum integers.

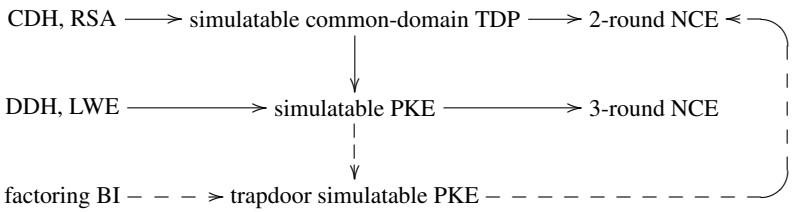

Fig. 1. Summary of previous results (solid lines) along with our contributions (dashed lines).

Theorem 2 (informal). *Suppose factoring Blum integers is hard on average. Then, there exists a trapdoor simulatable cryptosystem.*

We stress that we do not know how to construct a simulatable cryptosystem under the same assumptions; specifically, inverting the sampling algorithm for ciphertexts in our construction without the trapdoor (the factorization of the Blum integer modulus) appears to be as hard as factoring Blum integers. This shows that trapdoor simulatable cryptosystems is indeed a meaningful and useful relaxation. In the process, we also obtain the first factoring-based dense cryptosystems.[4] When combined with enhanced trapdoor permutations, this yields the first factoring-based non-interactive proofs of knowledge [DP92].

Oblivious transfer and MPC. We consider the applications of our main result to the constructions of adaptively secure oblivious transfer and general MPC protocols in both the stand-alone setting and the UC setting (c.f. [CLOS02,IPS08,CDSMW09]).

Theorem 3 (informal). *There exists a black-box construction of a* 6*-round* 1*-out-of-*ℓ *oblivious transfer protocol for strings in the* $\mathcal{F}_{\text{COM}}$*-hybrid model*[5] *in the UC setting that is secure against a malicious, adaptive adversary, starting from any trapdoor simulatable cryptosystem.*

We add that if the oblivious key generation algorithm in the trapdoor simulatable cryptosystem achieves statistical indistinguishability (which is the case for all of the afore-mentioned constructions), then we obtain an OT protocol that is secure against a computationally unbounded malicious sender. While our OT protocol is not as efficient as that in the recent work of Garay, Wichs and

[4] These are PKE schemes where a random string has a inverse polynomial probability of being a valid public key.

[5] $\mathcal{F}_{\text{COM}}$ is an ideal functionality for commitment.

Zhou [GWZ09] (we incur an additional multiplicative overhead that is linear in the security parameter), our protocol along with our general framework offers several advantages:

- In addition to relying on the $\mathcal{F}_{\text{COM}}$ functionality and a simulatable PKE (to implement non-committing encryption) as in our work, the [GWZ09] framework requires a so-called enhanced dual-mode cryptosystem. This is a relatively high-level CRS-based primitive from [PVW08] augmented with two main additional properties: the first has a flavor of oblivious sampling; the second requires that the underlying CRS be a common *random* string (modulo some system parameters) and not just a common reference string. This requirement is inherent to their framework, since this CRS is generated using a coin-tossing protocol. This latter requirement is very restrictive, and the only known construction of an enhanced dual-mode cryptosystem is based on the quadratic residuocity assumption.
- Our protocol immediately handles 1-out-of-ℓ OT, whereas [GWZ09] only addresses 1-out-of-2 OT, a limitation inherited from [PVW08].

Combined with [CLOS02,IPS08,CDSMW09], we obtain the following corollaries:

Corollary 1 (informal). *Assuming the existence of trapdoor simulatable cryptosystems, there exists adaptively secure multi-party protocols in the stand-alone setting and in the $\mathcal{F}_{\text{COM}}$-hybrid model in the UC setting against a malicious adversary that may adaptively corrupt any number of parties.*

Specifically, we obtain the first adaptively secure multi-party protocols based on hardness of factoring in both the stand-alone setting and the UC setting with a common reference string.

1.2 Additional related work

The problem of constructing encryption schemes that are secure against adaptive corruptions was first addressed in the work of Beaver and Haber [BH92]. They considered a simpler scenario where the honest parties have the ability to securely and completely erase previous states. For instance, an honest sender could erase the randomness used for encryption after sending the ciphertext, so that upon being corrupted, the adversary only gets to see the corresponding plaintext. An intermediate model, wherein we assume secure erasures for either the sender or receiver but not both (or, by limiting the adversary to corrupting at most one of the two parties), has been considered in several other works [JL00,CHK05,KO04].

Organization. We present an overview of our constructions in Section 2, preliminaries in Section 3, the formulation of a trapdoor simulatable PKE in Section 4, our factoring-based trapdoor simulatable PKE in Section 6, and our non-committing encryption scheme in Section 5. In Section 7, we show the construction of a 6-round oblivious transfer protocol.

2 Overview of our constructions

At a high level, our non-committing PKE is similar to that from previous works [CFGN96,DN00,KO04]. The receiver generates a collection of public keys in such a way that it only knows an α fraction of the corresponding secret keys; this can be achieved by generating an α fraction of the public keys using the key generation algorithm and the remaining $1 - \alpha$ fraction obliviously. Similarly, the sender generates a collection of ciphertexts in such a way that it only knows an α fraction of the corresponding plaintexts. Previous constructions all work with the natural choice of $\alpha = 1/2$ so that the simulator generates a collection of ciphertexts half of which are encryptions of 0 and the other half are encryptions of 1. As noted in [KO04], this is sufficient for obtaining non-committing PKEs wherein at most one party is corrupted. Roughly speaking, the difficulty in handling simultaneous corruptions of both the sender and the receiver with $\alpha = 1/2$ is that in the simulation, the sender's choice of the α fraction of keys completely determine the receiver's choice of the α fraction of ciphertexts whereas in an actual honest encryption, these choices are completely independent (we elaborate on this later in this section). The key insight in our construction is to work with a smaller value of α (turns out $1/4$ is good enough).

A toy construction. Consider the following encryption scheme, which is a simplification of that in [KO04,DN00]. The receiver generates a pair of public keys $(\mathrm{PK}_0, \mathrm{PK}_1)$ by generating one key (selected at random) using the key-generation algorithm, and the other using the oblivious sampling algorithm. To encrypt a bit b, the sender generates a pair of ciphertexts (C_0, C_1) as follows: pick a random bit r, set C_r to be $\mathsf{Enc}_{\mathrm{PK}_r}(b)$ and choose C_{1-r} using the oblivious sampling algorithm. To decrypt, the receiver decrypts exactly one of C_0, C_1 using the secret key that it knows. This construction corresponds to $\alpha = 1/2$ where α is the fraction of public keys for which the receiver knows the secret key, and also the fraction of ciphertexts for which the sender knows the plaintext. Observe that this encryption scheme has the following properties:

– It has a constant decryption error of $1/4$ if an obliviously sampled ciphertext is equally likely to decrypt to 0 or 1. As shown in [KO04], this error can be reduced by standard repetition techniques.

– It tolerates corruption of either the sender or the receiver, but not both. Consider a simulator that generates both of $(\mathrm{PK}_0, \mathrm{PK}_1)$ (along with $\mathrm{SK}_0, \mathrm{SK}_1$) using the key-generation algorithm, and a ciphertext (C_0, C_1) as follows: pick a random bit β, and set C_0 to be $\mathsf{Enc}_{\mathrm{PK}_0}(\beta)$ and C_1 to be $\mathsf{Enc}_{\mathrm{PK}_1}(1-\beta)$. Suppose the simulator later learns that this is an encryption of 0. If only the sender is corrupted, the simulator claims $r = \beta$ and that $C_{1-\beta}$ is obliviously sampled. If only the receiver is corrupted, it claims that it knows SK_β and that $\mathrm{PK}_{1-\beta}$ is oblivious sampled.

We highlight two subtleties in the above simulation strategy. First, it achieves 0 decryption error (as opposed to $1/4$ in an honest encryption); this can be fixed with a somewhat more involved simulation strategy. This in turn becomes pretty complicated once we use standard repetition techniques to reduce the decryption error. Next, it is always the case in the simulation that either both PK_0 and C_0 are obliviously sampled, or both PK_1 and C_1 are obliviously sampled. As such, this simulation strategy fails if both the sender and the receiver are corrupted, because in an actual encryption, which of $\mathrm{PK}_0, \mathrm{PK}_1$ and which of C_0, C_1 are obliviously sampled are determined independently.

Our encryption scheme. As noted in the introduction, the key insight in our construction is to work with a small value of α. In addition, following [DN00], we use a random k-bit encoding of 0 and 1, where k is the security parameter:

– The receiver generates $4k$ public keys $\mathrm{PK}_1, \ldots, \mathrm{PK}_{4k}$: k of them are generated using the key-generation algorithm, and the remaining $3k$ are generated using the oblivious sampling algorithm. The receiver then sends $\mathrm{PK}_1, \ldots, \mathrm{PK}_{4k}$ along with two random k-bit messages M_0, M_1.
– To encrypt a bit b, the sender sends $4k$ ciphertexts (one for each of $\mathrm{PK}_1, \ldots, \mathrm{PK}_{4k}$), of which k are encryptions of M_b, and the remaining ones are obliviously sampled.
– To decrypt, the receiver decrypts the k ciphertexts for which it knows the corresponding secret key. If any of the k plaintexts matches M_0, it outputs 0 and otherwise, it outputs 1.

Encoding 0 and 1 randomly as M_0 and M_1 is useful for two reasons:

– That an obliviously sampled ciphertext is equally likely to decrypt to 0 or 1 is no longer needed to guarantee correctness (c.f. [DN00]). Indeed, reasoning about decryptions of obliviously sampled ciphertext is non-trivial for the lattice-based simulatable PKEs in [GPV08].

– Constructing a simulator becomes much easier as we avoid having to generate distributions over k independent biased bits conditioned on the majority of the bits being 0, say. Generating such distributions arises for instance in [CFGN96] and is related to the first subtlety associated with the naive simulation strategy. In our construction, the simulated ciphertext comprises k encryptions of M_0, k encryptions of M_1, and $2k$ obliviously generated ciphertexts. Having these extra $2k$ obliviously generated ciphertexts (which is possible because $\alpha < 1/2$) is crucial for handling simultaneous corruptions of the sender and the receiver.

Trapdoor Simulatable PKEs from factoring. Our factoring-based trapdoor simulatable PKE construction consists of two main steps. First, we modify the Rabin trapdoor permutations based on squaring modulo Blum integer so that it remains a permutation over any arbitrary integer modulus. This relies on the following number-theoretical structural lemma implicit in [H99,S96,FF02][6]:

> Let N be an arbitrary odd k-bit integer, and let $Q_N = \{a^{2^k} \pmod N \mid a \in Z_N^*\}$. Then, the map $\psi : x \mapsto x^2$ defines a permutation over Q_N.

We also provide an efficient algorithm for inverting ψ given the factorization of N. Note that the standard algorithm for computing square roots does not guarantee that the output lies in Q_N. Moreover, the probability that a random square root lies in Q_N may be exponential small so we cannot repeatedly computing random square roots until we find one in Q_N; it's also not clear a-priori how to test membership in Q_N even given the factorization of N.

The next step transforms the family of trapdoor permutations ψ acting on the domain Q_N into a family of "enhanced" trapdoor permutations with the same domain Q_N, using an idea from [G04, Section C.1]. The latter has the property that we can obliviously sample a random element y in Q_N so that given y along with the coin tosses used to sample y, it is infeasible to compute the preimage of y under the permutation (note that the naive algorithm for sampling a random element of Q_N gives away its preimage under ψ). We will need the oblivious sampling algorithm for a random element in Q_N in our oblivious sampling algorithm for random ciphertexts. We will also need to realize trapdoor invertibility for the latter, which requires an efficient algorithm that given the factorization of N and an element y in Q_N, outputs a random 2^k'th root of y.[7]

[6] It was shown in [H99] that ψ defines a permutation over the subgroup O_N of Z_N^* of odd order, and that O_N contains Q_N; turns out $O_N = Q_N$. While Q_N is trivially sampleable, it is not clear a-priori how to sample from O_N.

[7] If we are given just N and not its factorization, this problem is at least as hard as factoring random Blum integers. This is in essence why we only obtain a factoring-based trapdoor simulatable PKE and not a simulatable PKE.

Note that iteratively computing random square roots k times does not work: after computing the first square root, we may not end up with a 2^{k-1}'th power.

3 Preliminaries

If A is a probabilistic polynomial time (hereafter, ppt) algorithm that runs on input x, $A(x)$ denotes the random variable according to the distribution of the output of A on input x. We denote by $A(x;r)$ the output of A on input x and random coins r. To simplify the notation, we will often omit quantifying over the distribution for r; it will usually be clear from the context when r is not fixed, that it is drawn from the uniform distribution over strings of the appropriate length.

We assume that the reader is familiar with the standard definitions of public-key encryption schemes and semantic security (c.f. [GM84,G04]). We stress that we allow decryption errors that are exponentially small in k:

Definition 1 (encryption scheme). *A triple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an encryption scheme, if* Gen *and* Enc *are ppt algorithms and* Dec *is a deterministic polynomial-time algorithm such that for every message* $m \in \{0,1\}^*$ *of polynomial length,* $\Pr[\mathsf{Gen}(1^k) \to (\mathrm{PK}, \mathrm{SK}), \mathsf{Enc}_{\mathrm{PK}}(m) \to c; \mathsf{Dec}_{\mathrm{SK}}(c) \neq m] < 2^{-\Omega(k)}$.

Non-committing encryption. For simplicity, we present the definition of a non-committing public-key encryption scheme for single-bit messages:

Definition 2 (non-committing encryption [CFGN96]). *A* non-committing (bit) encryption scheme *consists of a tuple* $(\mathsf{NCGen}, \mathsf{NCEnc}, \mathsf{NCDec}, \mathsf{NCSim})$ *where* $(\mathsf{NCGen}, \mathsf{NCEnc}, \mathsf{NCDec})$ *is an encryption scheme and* NCSim *is the simulation algorithm that on input* 1^k*, outputs* $(e, c, \sigma_{\mathrm{G}}^0, \sigma_{\mathrm{E}}^0, \sigma_{\mathrm{G}}^1, \sigma_{\mathrm{E}}^1)$ *with the following property: for* $b = 0, 1$ *the following distributions are computationally indistinguishable:*

- *the joint view of an honest sender and an honest receiver in a normal encryption of* b*:*
$$\{(e, c, \sigma_{\mathrm{G}}, \sigma_{\mathrm{E}}) \mid (e, d) = \mathsf{NCGen}(1^k; \sigma_{\mathrm{G}}), c = \mathsf{NCEnc}_e(b; \sigma_{\mathrm{E}})\}$$
- *simulated view of an encryption of* b*:*
$$\{(e, c, \sigma_{\mathrm{G}}^b, \sigma_{\mathrm{E}}^b) \mid \mathsf{NCSim}(1^k) \to (e, c, \sigma_{\mathrm{G}}^0, \sigma_{\mathrm{E}}^0, \sigma_{\mathrm{G}}^1, \sigma_{\mathrm{E}}^1)\}$$

It follows from the definition that a non-committing encryption scheme is also semantically secure.

Encrypting longer messages. Starting with a non-committing bit encryption scheme $(\mathsf{NCGen}, \mathsf{NCEnc}, \mathsf{NCDec}, \mathsf{NCSim})$, we may encrypt a longer message of length n by generating n independent public keys using NCGen, encrypting each bit of the message using a different public key and then concatenating the n ciphertexts. Note that this is different from the case of semantically secure encryption, where we may encrypt each bit using the same public key.

4 Trapdoor Simulatable Public Key Encryption

A *ℓ-bit trapdoor simulatable encryption scheme* consists of an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ augmented with $(\mathsf{oGen}, \mathsf{oRndEnc}, \mathsf{rGen}, \mathsf{rRndEnc})$. Here, oGen and $\mathsf{oRndEnc}$ are the oblivious sampling algorithms for public keys and ciphertexts, and rGen and $\mathsf{rRndEnc}$ are the respective inverting algorithms[8]. We require that, for all messages $m \in \{0,1\}^\ell$, the following distributions are computationally indistinguishable:

$$\{\mathsf{rGen}(r_{\mathrm{G}}), \mathsf{rRndEnc}(r_{\mathrm{G}}, r_{\mathrm{E}}, m), \mathrm{PK}, c \mid (\mathrm{PK}, \mathrm{SK}) = \mathsf{Gen}(1^k; r_{\mathrm{G}}), c = \mathsf{Enc}_{\mathrm{PK}}(m; r_{\mathrm{E}})\}$$
$$\text{and } \{\hat{r}_{\mathrm{G}}, \hat{r}_{\mathrm{E}}, \hat{\mathrm{PK}}, \hat{c} \mid (\hat{\mathrm{PK}}, \perp) = \mathsf{oGen}(1^k; \hat{r}_{\mathrm{G}}), \hat{c} = \mathsf{oRndEnc}_{\hat{\mathrm{PK}}}(1^k; \hat{r}_{\mathrm{E}})\}$$

It follows from the definition that a trapdoor simulatable encryption scheme is also semantically secure.

Encrypting longer messages. We note that if we started only with a trapdoor simulatable PKE for single bits, we may encrypt a longer message of length n by generating a single public key PK using Gen, and concatenating each of the message encrypted under PK.

5 Non-Committing Encryption from Weaker Assumptions

Theorem 4. *Suppose there exists a trapdoor simulatable encryption scheme. Then, there exists a non-committing encryption scheme as well as a universally composable oblivious transfer protocol secure against semi-honest, adaptive adversaries.*

We show how to construct a non-committing bit encryption scheme (NCGen, NCEnc, NCDec, NCSim) from a k-bit trapdoor simulatable PKE $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (augmented with $(\mathsf{oGen}, \mathsf{oRndEnc}, \mathsf{rGen}, \mathsf{rRndEnc})$). This is sufficient to establish the theorem by the connection between encrypting single bits and multiple bits as discussed in Sections 3 and 4. Our construction is presented in Figures 2 and 3.

[8] Existence of such inverting algorithms is called *trapdoor invertibility*. Compared to the simulatable cryptosystem (without trapdoor) defined in [DN00], rGen (resp. $\mathsf{rRndEnc}$) takes r_{G} (resp. $(r_{\mathrm{G}}, r_{\mathrm{E}}, m)$) as the additional trapdoor information.

Key Generation $\mathsf{NCGen}(1^k)$:

1. Pick M_0, M_1 at random from $\{0,1\}^k$.
2. Choose a random subset $T \subseteq [4k]$ of size k.
3. For $i = 1, 2, \ldots, 4k$, generate a pair $(\mathrm{PK}_i, \mathrm{SK}_i)$ as follows:

$$(\mathrm{PK}_i, \mathrm{SK}_i) = \begin{cases} \mathsf{Gen}(1^k) & \text{if } i \in T \\ \mathsf{oGen}(1^k) & \text{otherwise} \end{cases}$$

Set $e = [M_0, M_1, \mathrm{PK}_1, \ldots, \mathrm{PK}_{4k}]$ and $d = [T, \mathrm{SK}_1, \ldots, \mathrm{SK}_{4k}]$.

Encryption $\mathsf{NCEnc}_{\mathrm{PK}}(b)$:

1. Choose a random subset $S \subseteq [4k]$ of size k.
2. For $i = 1, 2, \ldots, 4k$, generate a ciphertext c_i as follows:

$$c_i = \begin{cases} \mathsf{Enc}_{\mathrm{PK}_i}(M_b) & \text{if } i \in S \\ \mathsf{oRndEnc}_{\mathrm{PK}_i}(1^k) & \text{otherwise} \end{cases}$$

Set $c = [c_1, \ldots, c_{4k}]$.

Decryption $\mathsf{NCDec}_{\mathrm{PK}}(c)$:

1. Compute $J = \{\mathsf{Dec}_{\mathrm{PK}_i}(c_i) | i \in T\}$.
2. If $M_0 \in J$, output 0; else, output 1.

Fig. 2. Non-Committing Encryption Scheme $(\mathsf{NCGen}, \mathsf{NCEnc}, \mathsf{NCDec})$

Correctness. We begin by establishing correctness.

- Assume that the input $[c_1, \ldots, c_{4k}]$ to the decryption algorithm is a random encryption of 0. Recall that $J = \{\mathsf{Dec}_{\mathrm{SK}_i}(c_i) \mid i \in T\}$ and we will output 0 unless $M_0 \notin J$. It is easy to see that $\Pr[M_0 \notin J] \leq \binom{3k}{k} / \binom{4k}{k} + 2^{-\Omega(k)}$ where the first summand comes from the probability that $S \cap T = \emptyset$ and the second bounds the probability of a decryption error in the underlying encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
- Assume that the input $[c_1, \ldots, c_{4k}]$ to the decryption algorithm is a random encryption of 1. Recall that $J = \{\mathsf{Dec}_{\mathrm{SK}_i}(c_i) \mid i \in T\}$ and we will output 1 unless $M_0 \in J$. To bound $\Pr[M_0 \in J]$, observe that the distribution of J depends only on $M_1, \mathrm{PK}_1, \ldots, \mathrm{PK}_{4k}$, $T, \mathrm{SK}_1, \ldots, \mathrm{SK}_{4k}$ and the coin tosses used to generate $c_1, \ldots, c_{4k}$, and is therefore independent of the choice of a random M_0. This means that for each $i \in T$, the probability that $\mathsf{Dec}_{\mathrm{SK}_i}(c_i)$ equals M_0 is 2^{-k}. Taking a union bound, we obtain $\Pr[M_0 \in J] \leq k \cdot 2^{-k}$.

Security. We need to show that for each $b = 0, 1$, a normal encryption of b and a simulated encryption of b are computationally indistinguishable. Note that the view in a normal encryption of b contains two sets T, S which we will label as

Simulation NCSim:

1. Pick M_0, M_1 at random from $\{0,1\}^k$.
2. Picking the sets S_0, S_1, T_0, T_1:
   - Pick two random subsets S_0, T_0 of $[4k]$ each of size k.
   - Pick two random subsets S_1, T_1 of $[4k] \backslash (S_0 \cup T_0)$ such that $|S_1 \cap T_1| = |S_0 \cap T_0|$.
3. Generating the keys: for $i = 1, 2, \ldots, 4k$, set

$$(\text{PK}_i, \text{SK}_i) = \begin{cases} \mathsf{Gen}(1^k; r_{\text{G}}^i) & \text{if } i \in T_0 \cup S_0 \cup T_1 \cup S_1 \\ \mathsf{oGen}(1^k; \hat{r}_{\text{G}}^i) & \text{otherwise} \end{cases}$$

4. Generating the ciphertext: for $i = 1, 2, \ldots, 4k$, set

$$c_i = \begin{cases} \mathsf{Enc}_{\text{PK}_i}(M_0; r_{\text{E}}^i) & \text{if } i \in S_0 \\ \mathsf{Enc}_{\text{PK}_i}(M_1; r_{\text{E}}^i) & \text{if } i \in S_1 \\ \mathsf{oRndEnc}_{\text{PK}_i}(\hat{r}_{\text{E}}^i) & \text{otherwise} \end{cases}$$

5. Simulating an opening to b: set $\sigma_{\text{G}}^b = \{T_b, u_{\text{G}}^{b,1}, \ldots, u_{\text{G}}^{b,4k}\}$ and $\sigma_{\text{E}}^b = \{S_b, u_{\text{E}}^{b,1}, \ldots, u_{\text{E}}^{b,4k}\}$, where

$$u_{\text{G}}^{b,i} = \begin{cases} r_{\text{G}}^i & \text{if } i \in T_b \\ \mathsf{rGen}(r_{\text{G}}^i) & \text{if } i \in T_0 \cup T_1 \cup S_0 \cup S_1 \setminus T_b \\ \hat{r}_{\text{G}}^i & \text{otherwise} \end{cases}$$

$$u_{\text{E}}^{b,i} = \begin{cases} r_{\text{E}}^i & \text{if } i \in S_b \\ \mathsf{rRndEnc}(r_{\text{G}}^i, r_{\text{E}}^i, M_{1-b}) & \text{if } i \in S_{1-b} \\ \hat{r}_{\text{E}}^i & \text{otherwise} \end{cases}$$

Set $e = [M_0, M_1, \text{PK}_1, \ldots, \text{PK}_{4k}]$, $c = [c_1, \ldots, c_{4k}]$. Additionally output $\sigma_{\text{G}}^0, \sigma_{\text{E}}^0, \sigma_{\text{G}}^1, \sigma_{\text{E}}^1$.

Fig. 3. Non-Committing Encryption Scheme NCSim

T_b, S_b and we will append to the view two sets T_{1-b}, S_{1-b} that are determined as follows: pick two random subsets S_{1-b}, T_{1-b} of $[4k] \setminus (S_b \cup T_b)$ such that $|S_1 \cap T_1| = |S_0 \cap T_0|$; call this distribution H_0. We will also append to the view in a simulated encryption of b the sets T_{1-b}, S_{1-b} as determined by the experiment NCSim; call this distribution H_{4k}. We will show that the augmented distributions H_0 and H_{4k} are computationally indistinguishable in two steps:

Reasoning about the sets. First, we claim that the 4-tuple (S_0, T_0, S_1, T_1) in the augmented distribution H_0 and in H_{4k} are identically distributed. If $b = 0$, this is obvious since the distributions are defined in exactly the same way. The case for $b = 1$ follows from a symmetry argument, namely that if we switch (S_0, T_0) with (S_1, T_1) in the experiment NCSim, we get exactly the same distribution. Henceforth, it suffices to argue that H_0 and H_{4k} are computationally indistinguishable, conditioned on some fixed (S_0, T_0, S_1, T_1)

in both H_0 and H_{4k}. We may now WLOG focus on the case $b = 0$. In fact, we may as well also fix M_0, M_1 in both H_0 and H_{4k}. In addition to $S_0, T_0, S_1, T_1, M_0, M_1$, the distributions H_0, H_{4k} comprise:

- $4k$ public keys $\mathrm{PK}_1, \ldots, \mathrm{PK}_{4k}$ (generated using either Gen or oGen);
- $4k$ ciphertexts $c_1, \ldots, c_{4k}$ (generated using either Enc or $\mathsf{oRndEnc}$);
- $4k$ sets of coin tosses $u_{\mathrm{G}}^1, \ldots, u_{\mathrm{G}}^{4k}$ for generating the public/secret keys; and
- $4k$ sets of coin tosses $u_{\mathrm{E}}^1, \ldots, u_{\mathrm{E}}^{4k}$ for generating the ciphertexts.

That is, we have $4k$ tuples of the form $(\mathrm{PK}_i, c_i, u_{\mathrm{G}}^i, u_{\mathrm{E}}^i), i = 1, \ldots, 4k$ in each view. Since S_0, T_0, S_1, T_1 are fixed, each of these $4k$ tuples are independently sampled from some distribution that only depends on the index i. Denote by $X_1, \ldots, X_{4k}$ the random variables for the $4k$ tuples in H_0, and $Y_1, \ldots, Y_{4k}$ the random variables for the $4k$ tuples in H_{4k}.

The hybrid argument. Next, we argue that X_i and Y_i are computationally indistinguishable for $i = 1, \ldots, 4k$, from which the indistinguishability of H_0 and H_{4k} follows via a hybrid argument. There are several cases we need to consider:

- $i \in T_0$ or $i \in [4k] \setminus (T_0 \cup S_0 \cup T_1 \cup S_1)$. It is easy to verify that in either of these cases, X_i and Y_i are identically distributed.
- $i \in S_1$ ("$\mathsf{oGen}, \mathsf{oRndEnc} \cong \mathsf{Gen}, \mathsf{Enc}$"). Here, X_i is the distribution
$$\{\hat{\mathrm{PK}}, \hat{c}, \hat{r}_{\mathrm{G}}, \hat{r}_{\mathrm{E}} \mid (\hat{\mathrm{PK}}, \perp) = \mathsf{oGen}(\hat{r}_{\mathrm{G}}), \hat{c} = \mathsf{oRndEnc}_{\hat{\mathrm{PK}}}(\hat{r}_{\mathrm{E}})\}$$
and Y_i is the distribution
$\{\mathrm{PK}, c, \mathsf{rGen}(r_{\mathrm{G}}), \mathsf{rRndEnc}(r_{\mathrm{G}}, r_{\mathrm{E}}, M_1) \mid (\mathrm{PK}, \mathrm{SK}) = \mathsf{Gen}(r_{\mathrm{G}}), c = \mathsf{Enc}_{\mathrm{PK}}(M_1; r_{\mathrm{E}})\}$.
Indistinguishability follows immediately from the security of the trapdoor simulatable PKE.
- $i \in S_0 \setminus T_0$ ("$\mathsf{oGen}, \mathsf{Enc} \cong \mathsf{Gen}, \mathsf{Enc}$"). Here, X_i is the distribution
$$\{\hat{\mathrm{PK}}, c, \hat{r}_{\mathrm{G}}, r_{\mathrm{E}} \mid (\hat{\mathrm{PK}}, \perp) = \mathsf{oGen}(\hat{r}_{\mathrm{G}}), c = \mathsf{Enc}_{\hat{\mathrm{PK}}}(M_0; r_{\mathrm{E}})\}$$
and Y_i is the distribution
$$\{\mathrm{PK}, c, \mathsf{rGen}(r_{\mathrm{G}}), r_{\mathrm{E}} \mid (\mathrm{PK}, \mathrm{SK}) = \mathsf{Gen}(r_{\mathrm{G}}), c = \mathsf{Enc}_{\mathrm{PK}}(M_0; r_{\mathrm{E}})\}.$$
Indistinguishability follows again from the security of the trapdoor simulatable PKE.
- $i \in T_1 \setminus S_1$ ("$\mathsf{oGen}, \mathsf{oRndEnc} \cong \mathsf{Gen}, \mathsf{oRndEnc}$"). Here, X_i is the distribution
$$\{\hat{\mathrm{PK}}, \hat{c}, \hat{r}_{\mathrm{G}}, \hat{r}_{\mathrm{E}} \mid (\hat{\mathrm{PK}}, \perp) = \mathsf{oGen}(\hat{r}_{\mathrm{G}}), \hat{c} = \mathsf{oRndEnc}_{\hat{\mathrm{PK}}}(\hat{r}_{\mathrm{E}})\}$$
and Y_i is the distribution
$$\{\mathrm{PK}, \hat{c}, \mathsf{rGen}(r_{\mathrm{G}}), \hat{r}_{\mathrm{E}} \mid (\mathrm{PK}, \mathrm{SK}) = \mathsf{Gen}(r_{\mathrm{G}}), \hat{c} = \mathsf{oRndEnc}_{\mathrm{PK}}(\hat{r}_{\mathrm{E}})\}.$$
Indistinguishability follows again from the security of the trapdoor simulatable PKE.

Improving the efficiency. Instead of using sets $S, T \subset [4k]$ of size k, we choose $S, T \subset [40]$ of size 10. The previous analysis still goes through, except we now have a constant decryption error. To address this problem, we first encode the message[9] with a linear-rate error-correcting code that corrects a constant fraction of errors, and then encrypt the codeword with the encryption scheme with constant error.

6 Trapdoor Simulatable PKE from Hardness of Factoring

Theorem 5. *Suppose factoring Blum integers is hard on average, and that Blum integers are dense, then there exists a trapdoor simulatable PKE.*

For simplicity, we only present a 1-bit trapdoor simulatable encryption scheme; we may encrypt longer messages by encrypting bit by bit.

A number-theoretic lemma. Fix any k-bit integer modulus N and we will work with the group Z_N^*. We will use $\mathsf{factor}(N)$ to denote the factorization of N, and we define $Q_N = \{a^{2^k} \mid a \in Z_N^*\}$. Now, consider the map $\psi_N : Q_N \to Q_N$ given by $\psi_N(x) = x^2 \pmod N$. As shown in [H99, Facts 3.5-3.7], ψ_N defines a permutation on Q_N. We provide a more direct proof which also yields an efficient algorithm to invert ψ_N given $\mathsf{factor}(N)$.

Claim. The map ψ_N defines a permutation on Q_N.

Proof. Let q denote the largest odd divisor of $\phi(N)$, where $\phi(\cdot)$ is the Euler's totient function. It is easy to see that $\phi(N)$ divides $2^k q$, since $N < 2^k$. Take any $y \in Q_N$, where $y = a^{2^k}$. Then by Euler's theorem, $y^q = 1 \pmod N$ and thus $\psi_N(y^{(q+1)/2}) = y \pmod N$. Clearly, $y^{(q+1)/2} \in Q_N$, so the map ψ_N is surjective. Moreover, the range and domain of ψ_N have equal sizes, so ψ_N must define a bijection. □

The construction. We sketch the construction here; the formal construction is shown in Figure 4.

STEP 1: First, we construct a family of "weakly one-way" enhanced trapdoor permutations. We start by modifying ψ_N to obtain a new family of permutations π_N; the modification is analogous to that in [G04, Section C.1] to obtain enhanced trapdoor permutations from Rabin's trapdoor permutations.

[9] The codeword length (or, equivalently the message length) should be $\Omega(k)$. Then, by Chernoff bound, the number of decryption errors remains a constant fraction of the codeword length with overwhelming probability.

Key generation $\mathsf{Gen}(1^k)$:

1. Run Bach's algorithm using the randomness r_{G} to sample random $N_1, \ldots, N_{k^3} \in \{0,1\}^k$ along with their factorization $\mathsf{factor}(N_1), \ldots, \mathsf{factor}(N_{k^3})$.
2. Set $\mathrm{PK} = [N_1, \ldots, N_{k^3}]$ and $\mathrm{SK} = [\mathsf{factor}(N_1), \ldots, \mathsf{factor}(N_{k^3})]$.

Encryption $\mathsf{Enc}(b)$:

1. Parse the randomness r_{E} as $(a_1, \ldots, a_{k^3}) \in Z^*_{N_1} \times \cdots \times Z^*_{N_{k^3}}, r_1, \ldots, r_{k^3} \in \{0,1\}^k$ and $b_1, \ldots, b_{k^3-1} \in \{0,1\}$.
2. Compute $b_{k^3} = b \oplus b_1 \oplus \cdots \oplus b_{k^3-1}$.
3. Compute $x_i = a_i^{2^k} \in Q_{N_i}, i = 1, \ldots, k^3$.
4. Output $[\pi_{N_i}(x_i), r_i, (x_i \cdot r_i) \oplus b_i, i = 1, \ldots, k^3]$.

Decryption $\mathsf{Dec}(c)$:

1. Parse c as $[y_i, r_i, \beta_i, i = 1, \ldots, k^3]$.
2. Compute $b_i = (\pi^{-1}_{N_i}(y_i) \cdot r_i) \oplus \beta_i, i = 1, \ldots, k^3$.
3. Output $b_1 \oplus \cdots \oplus b_{k^3}$.

Oblivious key generation $\mathsf{oGen}(1^k)$:

1. Parse the randomness $\hat{r}_{\mathrm{G}} \in \{0,1\}^{k^4}$ as $N_1, \ldots, N_{k^3} \in \{0,1\}^k$.
2. Output $(N_1, \ldots, N_{k^3})$.

Trapdoor invertibility key generation $\mathsf{rGen}(r_{\mathrm{G}})$:

1. Run $\mathsf{Gen}(r_{\mathrm{G}})$ to obtain $\hat{r}_{\mathrm{G}} = (N_1, \ldots, N_{k^3})$.
2. Output $\hat{r}_{\mathrm{G}}$.

Oblivious sampling of ciphertexts $\mathsf{oRndEnc}(1^k)$:

1. Parse the randomness $\hat{r}_{\mathrm{E}}$ as $(\gamma_1, \ldots, \gamma_{k^3}) \in Z^*_{N_1} \times \cdots \times Z^*_{N_{k^3}}, s_1, \ldots, s_{k^3} \in \{0,1\}^k$ and $\beta_1, \ldots, \beta_{k^3} \in \{0,1\}$.
2. Compute $y_i = \gamma_i^{2^k} \in Q_{N_i}, i = 1, \ldots, k^3$.
3. Output $[y_i, s_i, \beta_i, i = 1, \ldots, k^3]$.

Trapdoor invertibility for ciphertexts $\mathsf{rRndEnc}(r_{\mathrm{G}}, r_{\mathrm{E}}, b)$:

1. Use r_{G} to compute $\mathsf{factor}(N_1), \ldots, \mathsf{factor}(N_k)$. and parse r_{E} as in Enc.
2. Set $s_i = r_i$ and $\beta_i = (x_i \cdot r_i) \oplus b_i, i = 1, \ldots, k^3$.
3. Pick a random γ_i uniformly from the set $\{\gamma_i \in Z^*_{N_i} \mid \gamma_i^{2^k} = \pi_{N_i}(x_i)\}$.
4. Output $\hat{r}_{\mathrm{E}} = (\gamma_1, \ldots, \gamma_{k^3}, s_1, \ldots, s_{k^3}, \beta_1, \ldots, \beta_{k^3})$.

Fig. 4. Trapdoor Simulatable PKE from hardness of factoring Blum integers

The permutations $\pi_N : Q_N \to Q_N$ are indexed by a k-bit integer N and is given by:

$$\pi_N(x) \stackrel{\text{def}}{=} \psi_N^{k+1}(x) = x^{2^{k+1}} \pmod{N}$$

and the trapdoor is $\mathsf{factor}(N)$. We may sample from this family by running Bach's algorithm [B88,K02] to pick a random k-bit integer along with its factorization.

It is easy to verify π_N is a family of trapdoor permutations. Clearly, π_N is a permutation because it is the $(k+1)$-fold iterate of a permutation ψ_N. Given the index N, π_N is efficiently computable by repeated squaring. Given the trapdoor $\mathsf{factor}(N)$, π_N^{-1} is efficiently computable given $\mathsf{factor}(N)$, by simply mapping y to $y^{((q+1)/2)^{k+1}}$, i.e., raising y to the $(q+1)/2$'th power

$k+1$ times. Here, q denotes the largest odd divisor of $\phi(N)$, which is easy to compute with the trapdoor. Moreover, we can show that if N is a Blum integer (which occurs with probability $\Omega(1/k^2)$ [GM04,RS94]), then inverting π_N given N is at least as hard as factoring N. This implies that π_N is one-way with probability $\Omega(1/k^2)$ over the choice of N.

STEP 2: Construct a "weak" encryption scheme using the standard construction of PKE from trapdoor permutations via the Goldreich-Levin hard-core predicate. The public key is N, the secret key is $\mathsf{factor}(N)$, and to encrypt a bit b, we pick a random $x \in Q_N, r \in \{0,1\}^k$ and output $(\pi_N(x), r, (x \cdot r) \oplus b)$, where $x \cdot r$ is the standard dot-product of k-bit strings. Again, this scheme will be semantically secure with probability $\Omega(1/k^2)$ over the choice of N.

STEP 3: To boost the security of the "weak" encryption scheme, we define a new scheme where the public key is k^3 random k-bit strings $N_1, \ldots, N_{k^3}$ (with overwhelming probability, one of these is a Blum integer), and to encrypt a bit b, we pick random $b_1, \ldots, b_{k^3}$ such that $b = b_1 \oplus \cdots \cdot b_{k^3}$ and concatenate the encryptions of $b_1, \ldots, b_{k^3}$ under the respective public keys $N_1, \ldots, N_{k^3}$. By a standard argument (c.f. [Y82,DP92]), this encryption scheme is semantically secure in the standard sense.

Analysis. Indeed, we claim something stronger – that the encryption scheme derived in Step 3 is a trapdoor simulatable PKE.

- (Oblivious sampling & trapdoor invertibility for key generation) This is trivial, since a random public key corresponds to a string in $\{0,1\}^{4k}$. We can clearly sample such a public key without learning the secret key.
- (Oblivious sampling & trapdoor invertibility for random ciphertext) For simplicity, we present the algorithms for sampling random ciphertext for the scheme obtained in Step 2. Here, sampling is easy: on input the public key N, pick $\gamma \in Z_N^*, s \in \{0,1\}^k, \beta \in \{0,1\}$) and output (γ^{2^k}, s, β). To implement reverse sampling, we need an efficient algorithm that given $\mathsf{factor}(N)$ and $x \in Q_N$, output a random element of the set $\{\gamma \in Z_N^* \mid \gamma^{2^k} = \pi_N(x) = x^{2^{k+1}}\}$. This can be accomplished as follows: pick a random $\eta \in Z_N^*$ and output $x^2 \cdot \eta/(\eta^{2^k})^{((q+1)/2)^k}$, where q is as before the largest odd divisor of $\phi(N)$. This works because $\eta/(\eta^{2^k})^{((q+1)/2)^k}$ will be a random 2^k'th root of $1 \pmod N$.

For the actual proof of security, we will need to show that if N is a random Blum integer, then the following distributions are computationally indistinguishable for every b:

$$\{(N, \gamma, \pi_N(x), r, (x \cdot r) \oplus b)\} \text{ and } \{(N, \gamma, \gamma^{2^k}, r, \beta)\}$$

The first distribution corresponds to an encryption of b using modulus N and randomness (x, r) along with γ the output of rRndEnc (a random solution to the equation $\gamma^{2^k} = \pi_N(x)$). The second corresponds to an obliviously generated ciphertext along with the randomness. If there exists an efficient distinguisher, then there exists an efficient procedure A that on input N, γ, outputs $\pi_N^{-1}(\gamma^{2^k})$ with noticeable probability. Since squaring is a bijection on quadratic residues modulo Blum integers, the output of A is also the 4th root of γ^2. We may then use a reduction in [G04, Section C.1] to derive from A an algorithm for factoring N with noticeable probability.

7 Oblivious Transfer and MPC

We describe the construction underlying Theorem 3, which proceeds in two steps:

STEP 1: We begin with the [CLOS02] construction of a semi-honest OT protocol as applied to our non-committing encryption scheme, and observe that the protocol is secure against malicious senders. For that, we just need to show how to extract the sender's input when the receiver is honest. In this case, the simulator will generate the public keys sent by the receiver in the first message along with the secret keys, so that it can then extract the malicious sender's input by decrypting.

STEP 2: Next, we apply the compiler in [CDSMW09] to "boost" the security guarantee from tolerating semi-honest receivers to tolerating malicious receivers. (Note that we will not need to apply OT reversal as in [CDSMW09].)

Acknowledgements. We thank Ran Canetti, Yuval Ishai, Jonathan Katz, and Chris Peikert for helpful discussions and clarifications.

References

[B88] E. Bach. How to generate factored random numbers. *SIAM J. Comput.*, 17(2):179–193, 1988.

[B97] D. Beaver. Plug and play encryption. In *CRYPTO*, pages 75–89, 1997.

[B98] D. Beaver. Adaptively secure oblivious transfer. In *ASIACRYPT*, pages 300–314, 1998.

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *EUROCRYPT*, pages 307–323, 1992.

[C00] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[CDSMW09] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Simple, black-box constructions of adaptively secure protocols. In *TCC*, pages 387–402, 2009.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996. Longer version at `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/nce_abs.html`.

[CHK05] R. Canetti, S. Halevi, and J. Katz. Adaptively-secure, non-interactive public-key encryption. In *TCC*, pages 150–168, 2005.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[DN00] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.

[DP92] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS*, pages 427–436, 1992.

[FF02] M. Fischlin and R. Fischlin. The representation problem based on factoring. In *CT-RSA*, pages 96–113, 2002.

[G04] O. Goldreich. *Foundations of Cryptography: Volume II, Basic Applications*. Cambridge University Press, 2004.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GM04] A. Granville and G. Martin. Prime number races, 2004. http://arxiv.org/abs/math/0408319.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GWZ09] J. A. Garay, D. Wichs, and H.-S. Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO*, 2009. To appear. Also, Cryptology ePrint Archive, Report 2008/534.

[H99] S. Halevi. Efficient commitment schemes with bounded sender and unbounded receiver. *J. Cryptology*, 12(2):77–89, 1999.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[JL00] S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *EUROCRYPT*, pages 221–242, 2000.

[K02] A. Kalai. Generating random factored numbers, easily. In *SODA*, pages 412–412, 2002.

[KO04] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[RS94] M. Rubinstein and P. Sarnak. Chebyshevs bias. *Experiment. Math*, 3(3):173–197, 1994.

[S96] C.-P. Schnorr. Security of 2^t-root identification and signatures. In *CRYPTO*, pages 143–156, 1996.

[Y82] A. C.-C. Yao. Theory and applications of trapdoor functions. In *FOCS*, pages 80–91, 1982.