Mutually Independent Commitments

Moses Liskov[1], Anna Lysyanskaya[1], Silvio Micali[1],
Leonid Reyzin[2], and Adam Smith[1]

[1] Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{mliskov,anna,asmith}@theory.lcs.mit.edu
[2] Boston University
Department of Computer Science
Boston, MA 02215, USA
reyzin@bu.edu

Abstract. We study the two-party commitment problem, where two players have secret values they wish to commit to each other. Traditional commitment schemes cannot be used here because they do not guarantee independence of the committed values. We present three increasingly strong definitions of independence in this setting and give practical protocols for each. Our work is related to work in non-malleable cryptography. However, the two-party commitment problem can be solved much more efficiently than by using non-malleability techniques.

1 Introduction

We consider the scenario in which two players have some private values in mind, and want to commit these values to one another. In these circumstances, simply using commitment schemes on each side does not provide sufficient security. While this approach guarantees that the two commitments will each be hiding and binding, it does not guarantee their independence.

For example, if Alice is selling something to Bob, and commits to a (her lowest price) by publishing $c(a)$, then Bob can commit to a as his highest bid (without knowing the value), by copying $c(a)$. Thus, Bob will force Alice to always sell at her lowest price. Though this is an obvious and easily preventable attack, more sophisticated ones exist. For example, if the commitment scheme being used is that of Pedersen [Ped91], Bob could, without risking detection, copy (or indeed add an arbitrary constant to) Alice's value.

Independence of committed values is quite fundamental to secure two-party protocols. Indeed, in any protocol to which both parties have inputs which they are unwilling to reveal at the outset, the inputs must be committed (so that the parties cannot change their minds later) and independent (so that each party's influence of the outcome is limited to the choice of its own input).

The two-party commitment problem. In our setting, Alice and Bob have secret values a and b, respectively. They want to commit their values to each other. Informally, we want the following security properties to hold:

- *Hiding:* A dishonest party cannot discover the honest party's value.
- *Binding:* A dishonest party cannot open his or her commitment in more than one way.
- *Non-correlation:* A dishonest party cannot commit to a value that is in some significant way correlated to the honest party's value.

We formalize the last property in three increasingly stronger definitions.

- *Mutually independent announcement*: Non-correlation is guaranteed given that the parties open their commitments.
- *Mutually independent commitment*: Non-correlation is guaranteed once the commitments are exchanged and accepted.
- *Mutually independent and aware commitment*: Each party is guaranteed to *know* his or her own value once the commitments are exchanged and accepted. This property, combined with the hiding property of the commitment, actually guarantees non-correlation.

We also give practical protocols that satisfy these definitions. Specifically, we give a two-round[1] mutually independent announcement protocol based on the existence of one-way permutations. We give two mutually independent commitment protocols: a two-round protocol based on the assumption that subexponentially hard one-way permutations exist, and a three-round protocol based on the assumption that dense cryptosystems exist. Finally, we give a seven-round mutually independent and aware commitment protocol based on the discrete logarithm assumption. With the exception of an eleven-round mutually independent and aware protocol we present to elucidate the definitions, all the protocols we present are efficient enough to be useful in practice.

1.1 Mutual Independence Versus Other Notions

Mutually independent commitments provide a new approach to an important cryptographic problem: how ensure that *secret* and *committed* values are *independent*. This problem has been addressed before in other settings.

Independence in the Multi-party Setting. In protocols involving more than two parties, it has long been recognized that independence of committed values is fundamental to the very notion of security: a player who can correlate his input to those of other players (without necessarily knowing them) may be able to change the outcome of the protocol in his favor. In that setting, the problem of independence was introduced by Chor, Goldwasser, Micali and Awerbuch [CGMA85], who solved it using verifiable secret-sharing protocols.

[1] The round complexity we refer to is the number of rounds required for the commit stage of the protocols. All the protocols we present have one-round reveal stages, except the mutually independent announcement protocol, which has a two-round reveal stage.

Subsequent impovements to their solution were made by Chor and Rabin [CR87] and by Gennaro [Gen95].

Our two-party setting, while similar to the multi-party setting at first glance, is actually quite different: the multi-party protocols assume that a majority of players are honest. This allows "committed information" to actually be distributed among multiple players. Because we have only two players, we cannot assume an honest majority without trivializing the problem. Thus, each player in our setting will have *all* the committed information from the other player.

Non-malleability of Commitment Schemes. It has also long been recognized that the hiding property of a commitment scheme (carried out between two parties, a sender and a receiver) does not prevent an adversary from committing to a value related to someone else's commitment. In the two-party commitment setting, the notion of non-malleability was introduced to address this problem.

Defined by Dolev, Dwork and Naor [DDN00], non-malleability for commitment schemes captures the following intuitive notion: if Alice commits a value to Bob, and Bob commits a value to Charlie (using the same commitment scheme), then Bob's committed value should be independent of Alice's. Thus, this is a setting with two honest parties (Alice and Charlie) who are unaware of each other, and one adversary (Bob). Because Alice and Charlie are unaware of each other, Bob can arbitrarily vary the timing of the two interactions in which he is involved. This, in particular, implies that Bob can always just copy Alice's committed value by simply being a "transparent intermediary." Copying committed values is, in fact, explicitly permitted in the definiton of [DDN00].

In our setting there are three crucial differences. First, Alice and Charlie are, in a sense, the same person. (This, in particular, prevents Bob from arbitrarily scheduling the exections of the two commitment protocols and thus copying the committed value.) Second, we are not restricted to using the same commitment scheme for the two commitments. Finally, either party in our setting can be the adversary, and independence needs to be ensured both ways.

While, as we describe in the next section, non-malleable commitment schemes may be used to provide mutually independent commitments, the mutually independent commitment problem can be solved more efficiently in other ways.

1.2 Relevance of Prior Solutions

As we pointed out above, solutions in the multiparty setting seem inapplicable to our setting, because they assume an honest majority of players. Non-malleable commitments, on the other hand, can address our problem.

In fact, any commitment protocol non-malleable *with respect to commitment* (i.e., in which non-malleability is assured even if the adversary never sees Alice's decommitted value) can be used to provide mutually independent commitments: simply run two copies of the protocol in parallel, one from Alice to Bob and the other from Bob to Alice. Either party can detect if the other is copying the transcript, and thus prevent it from copying the commitment[2]. However, only one non-malleable commitment protocol is known that does not require extra set-up assumptions: the one of [DDN00]. It is quite impractical, requiring a non-constant number of rounds (it will, however, achieve mutually independent and *aware* commitments in our setting). In contrast, we present simple constant-round protocols that solve the problem.

A number of much simpler non-malleable commitment schemes (constructed using either non-malleable encryption [DDN00,CS98,Sah99] or directly [DIO98,FF00,DKOS01,CF01]) are known, all requiring trusted public file to be set up ahead of time. Because we are interested in a two-party scenario, we are unwilling to assume the existence of trusted public parameters.

Moreover, some of the above commitment schemes [DIO98,FF00,DKOS01] achieve only a weaker security notion called non-malleability *with respect to opening*. That is, it may be possible for Bob to commit to a value related to Alice's, but he won't know how to open it. Using such a protocol in our setting will achieve mutually independent *announcement*, but not necessarily *commitment* (in particular, some of the schemes are perfectly hiding, and then it is unclear how the committed value can be defined prior to opening).

1.3 Applicability of Mutually Independent Commitments

The protocols we present are for the two-party model. We intend our notions to be useful as essential building blocks in secure two-party computation protocols.

As we have discussed, the question of mutual independence also naturally arises in the multi-party setting, and has been previously studied. By focusing on the two-party case exlusively, we obtain protocols that are more efficient and conceptually simpler. Applicability of our techniques in other settings is a subject of further study.

2 Definitions

2.1 Notation

Negligible functions. The expression negl(k) is used to denote any function f that is negligible in k; that is, for any positive polynomial q, $f(k) = o(1/q(k))$.

Probability .[3] If S is a probability space, then "$x \leftarrow S$" denotes assigning to x an element randomly selected according to S. If F is a finite set, then the notation "$x \leftarrow F$" denotes the algorithm that chooses x uniformly from F.

[2] The original definition of [DDN00] did not prevent Bob from copying the committed value while *not* copying the transcript. However, any non-malleable commitment scheme can be modified to preclude this possibilty [KOS], thus leaving Bob only one way to copy the commitment: by copying the transcript exactly.

[3] This notation closely follows that of [BDMP91] and [GMR88].

If p is a predicate, the notation $\Pr[x \leftarrow S; y \leftarrow T; \cdots : p(x, y, \cdots)]$ denotes the probability that $p(x, y, \cdots)$ will be true after the ordered execution of the algorithms $x \leftarrow S$; $y \leftarrow T; \cdots$. The notation $[x \leftarrow S; y \leftarrow T; \cdots : (x, y, \cdots)]$ denotes the probability space over $\{(x, y, \cdots)\}$ generated by the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, \cdots$.

Protocols. The schemes discussed in this paper are protocols $P = (A, B)$ run between two parties, A and B. Both A and B are probabilistic polynomial-time interactive Turing machines (ppITMs). Given (1) a *security parameter* 1^k, which is available to both parties; (2) *inputs* (a, b), where a is private to A and b is private to B; and (3) *random tapes* (r_A, r_B), where r_A is private to A and r_B is private to B, protocol P computes in a sequence of rounds, alternating between A-rounds and B-rounds. In an A-round (respectively, B-round) only A (only B) is active and sends a string that will become an available input to B (to A) in the next B-round (A-round). We will divide P into two stages: the *commit* stage $P_C = (A_C, B_C)$, and the *reveal* stage $P_R = (A_R, B_R)$ (state information for A and B is saved between the stages). At the end of the commit stage, A_C and B_C will each output "accept" or "reject." At the end of the reveal stage, A_R will output the value β that B revealed to it, which is a string or a special symbol "reject", and B_R will similarly output the value α. For notational convenience, we will assume that if the output of the commit stage is "reject," then so is the output of the reveal stage—i.e., if a party did not accept the commitment, it will not accept its revealing, either. The terms "output of A" and "output of B" shall mean "output of A_R" and "output of B_R."

We will also consider the situation in which one of the two parties is dishonest. The dishonest party, denoted by A' or B', is also a ppITM. A dishonest party can, of course, simply stop the protocol before the other party produces an output. In such a case, for notational convenience, we will consider the honest party's output to be "reject."

Transcripts, views, and outputs.[4] Letting E be an execution of a protocol (A, B) on inputs $(1^k, a, b, r_A, r_B)$, we make the following definitions:

- The *transcript* of E consists of the sequence of messages exchanged by A and B, and is denoted by $\mathrm{TRANS}^{A,B}(1^k, a, b, r_A, r_B)$ (for notational convenience, we will include the outputs of A and B into the transcript);
- The *view of* A consists of the triplet $(1^k, a, r_A, t)$, where t is E's transcript, and is denoted by $\mathrm{VIEW}_A^{A,B}(1^k, a, b, r_A, r_B)$;
- The *output of* A is denoted by $\mathrm{OUT}_A^{P_A,P_B}(a, b, r_A, r_B)$;
- The output and view of B are defined similarly and denoted by $\mathrm{VIEW}_B^{A,B}(a, b, r_A, r_B)$ and $\mathrm{OUT}_B^{P_A,P_B}(a, b, r_A, r_B)$;
- We use the symbol $\cdot$ in place of r_A and r_B in the above notation to denote the distribution induced on the transcripts, views and outputs when r_A or r_B is selected at random. Thus, for example, $\mathrm{TRANS}^{A,B}(1^k, a, b, \cdot, \cdot)$, is a

[4] We borrow much of our protocol notation from [BMM99] and [GMR85].

probability space of transcripts, with probabilities induced by selecting r_A and r_B at random and executing (A, B) on $(1^k, a, b, r_A, r_B)$.

The output of each party is, of course, computed based solely on that party's view. Therefore, we denote by $\mathrm{OUT}_A(1^k, a, r_A, t)$ the output of A as computed on the particular view (note that A is not assumed to be interacting with anyone in this case). We use similar notation for the output of B.

Finally, we will denote the transcript for the commit stage by t_C, the transcript for the reveal stage by t_R, and the combined transcript by $t = t_C \circ t_R$.

2.2 Mutually Independent Announcement

A protocol (A, B) is a *mutually independent announcement* if the following properties hold:

- *A-completeness.* If A and B are honest, then A can can commit and reveal her value successfully:

  $$\begin{array}{l} \forall a, b \\ \Pr[\alpha \leftarrow \mathrm{OUT}_B^{A,B}(1^k, a, b, \cdot, \cdot) : \\ \quad \alpha = a] = 1 - \mathrm{negl}(k) \end{array}$$

- *A-soundness.* This property prevents a dishonest B' from influencing which value A commits to. That is, if the honest A is interacting with a dishonest B'_C during the commit stage and outputs "accept," then A would only reveal a during the reveal stage, at least with an honest B_R.

  $$\begin{array}{l} \forall t_C, t_R, a, b, r_A, r_B, \\ \mathrm{OUT}_{A_C}(1^k, a, r_A, t_C) = \text{“accept”} \wedge \\ \mathrm{OUT}_{B_R}(1^k, b, r_B, t_C \circ t_R) = \alpha \Rightarrow \\ \alpha = a \end{array}$$

- *Computational A-hiding.* No adversary B', interacting only with A_C, the commit stage of A, can break the GM-security [GM84] of A's commitments:

  $$\begin{array}{l} \forall (a_0, a_1)\ \forall B' \\ \Pr[v \leftarrow \{0, 1\}; \\ \quad z \leftarrow \mathrm{OUT}_{B'}^{A_C,B'}(a_v, (a_0, a_1), \cdot, \cdot) : \\ \quad z = v] < 1/2 + \mathrm{negl}(k) \end{array}$$

- *Perfect A-binding.* If the commit stage B_C of B outputs "accept," then the reveal stage B_R will accept only one revealed value; moreover, this value depends only on the transcript of the reveal stage, not on the private input of B:

  $$\begin{array}{l} \forall t_C, b, b', r_B, r'_B, t_R, t'_R, \alpha, \alpha', \\ \mathrm{OUT}_{B_C}(1^k, b, r_B, t_C) = \text{“accept”} \wedge \\ \mathrm{OUT}_{B_C}(1^k, b', r'_B, t_C) = \text{“accept”} \wedge \\ \mathrm{OUT}_{B_R}(1^k, b, r_B, t_C \circ t_R) = \alpha \wedge \\ \mathrm{OUT}_{B_R}(1^k, b', r'_B, t_C \circ t'_R) = \alpha' \Rightarrow \\ \alpha = \alpha'. \end{array}$$

- *A-non-correlation at opening.* To define non-correlation at opening, we use techniques similar to those used in defining commitment schemes that are non-malleable with respect to opening ([DDN00,DIO98,FF00]). The definition essentially states that for any polynomial-time relation R, any adversary B' that engages in a protocol with $A(1^k, a, r_A)$ and then opens his committed value as β, has no more chance of achieving $R(a, \beta)$ than a simulator who does not engage in any interaction with A at all. Note, of course, that because we are defining non-correlation at opening (rather than at commitment), B' may already know a before revealing β, and thus may simply refuse to reveal depending on a. If B' refuses to reveal, A will output "reject." There is no getting around the fact that B' can correlate "reject" to a. Thus, as for non-malleability, we explicitly require that $R(a,\text{"reject"}) = 0$, so that forcing A to reject is not considered correlating to a better than a simulator. We call such polynomial-time relations *allowable.* Note that, unlike the definitions of non-malleability, we do not require that the relation be non-reflexive—that is, our definitions do not even allow B' to copy the commitment of A.

  $\forall B' \; \exists S \; \forall$ allowable $R \; \forall$ efficiently sampleable $\mathcal{D}$
  $\Pr[a \leftarrow \mathcal{D};$
  $\quad \beta \leftarrow \mathrm{OUT}_A^{A,B'}(1^k, a, -, \cdot, \cdot)$:
  $\quad R(a, \beta) = 1] <$
  $\Pr[a \leftarrow \mathcal{D};$
  $\quad \beta \leftarrow S(1^k, \mathcal{D})$:
  $\quad R(a, \beta) = 1] + \mathrm{negl}(k)$

- *B-completeness, B-soundness, computational B-binding, perfect B-binding, B-non-correlation at opening.* Defined the same way as for the above, with A and B reversed.

2.3 Mutually Independent Commitment

Mutually independent commitments are defined the same way as mutually independent announcements, except for the non-correlation property, which is defined as follows.

- *A-non-correlation at commitment.* Because our commitments are perfectly B-binding, at the end of the commit stage, there is at most one value β that a (dishonest) B' can reveal. Moreover, this value is determined uniquely by the transcript t_C of the commit stage, provided that A outputs "accept." Let $U_B(t_C)$ be this value, or $\perp$ if no such value exists or if A outputs "reject" (recall that we included A's output into the transcript, by definition). Note that, by B-hiding, $U_B(t_C)$ is not efficiently computable, at least when B is honest. A-non-correlation at commitment requires that $U_B(t_C)$ be not correlated to a, for any polynomial-time relation R. Unlike the case for non-correlation at opening, we do not require that $R(a, \perp) = 0$ —that is, B' should not even be able to correlate to a the fact that no valid decommitment exists. This stronger requirement is justified in this case, because B' does not get to see a before deciding whether a decommitment should exist.

$$\begin{array}{l}
\forall B'\ \exists S\ \forall \text{ poly-time } R\ \forall \text{ efficiently samplable } \mathcal{D} \\
\Pr[a \leftarrow \mathcal{D}; \\
\quad t_C \leftarrow \text{TRANS}^{A_C, B'_C}(1^k, a, \mathcal{D}, \cdot, \cdot) : \\
\quad R(a, U_B(t_C)) = 1] < \\
\Pr[a \leftarrow \mathcal{D}; \\
\quad \beta \leftarrow S(1^k, \mathcal{D}) : \\
\quad R(a, \beta) = 1] + \text{negl}(k)
\end{array}$$

- *B-non-correlation at commitment.* Defined similarly, with A and B reversed.

2.4 Mutually Independent and Aware Commitment

In addition to the properties of mutually independent commitments defined above, we want to capture the strong notion that B, if he accepts the commitment stage, is assured that A "knows" the value she committed to. We mean "knowledge" in the sense of the existence of a knowledge extractor E, in the tradition of the definitions of a proof of knowledge [TW87,FFS88]. Note, however, that unlike proofs of knowledge, where an NP-witness y is being extracted for a predetermined statement x, in our case, no predetermined statement exists. What is being extracted—the commited-to value—is determined only by the transcript t_C of the commitment stage. Thus, our definition gives E the view of the dishonest party (which includes the transcript of the conversation), and E has to extract, given oracle access to the dishonest party, the committed value.

- *A-awareness.* Similarly to the definition of A-non-correlation at commitment, given a transcript t_C of the commitment stage, let $U_A(t_C)$ be the unique value α that A' can reveal, or $\perp$ if no such value exists or if B output "reject" at the end of the commitment stage. Because the view $V_{A'_C}$ of A includes t_C, we will use $U_A(V_{A'_C})$ to mean $U_A(t_C)$.

$$\begin{array}{l}
\exists E\ \forall b\ \forall A' \\
\Pr[V_{A'_C} = \text{VIEW}^{A'_C, B_C}_{A'_C}(1^k, -, a, \cdot, \cdot); \\
\quad a \leftarrow E^{A'}(V_{A'_C}); \\
\quad \alpha = U_A(V_{A'_C}) : \\
\quad a = \alpha] > 1 - \text{negl}(k)
\end{array}$$

- *B-awareness.* Defined similarly, with A and B reversed.

Note that A-awareness, combined with B-hiding, implies B-non-correlation at commitment, because we can simply use E in place of the simulator S. Therefore, aware commitments are automatically mutually independent.

3 Protocols

3.1 Mutually Independent Announcement

Theorem 1. *If one-way permutations exist, then there exists a protocol for mutually independent announcements with a two-round commit stage and a two-round reveal stage*[5].

Proof sketch. The protocol is simplicity itself. Let c be a perfectly binding non-interactive commitment scheme, which can be constructed based on any one-way permutation [GL89]. The commit stage consists of two rounds:

1. Alice sends $c(a)$ to Bob
2. Bob sends $c(b)$ to Alice

The reveal stage likewise consists of two rounds:

1. Bob opens his commitment
2. Alice opens her commitment

The completeness, soundness, binding, and hiding properties are easy to see. It is clear that A non-correlation at opening holds, since after step 1 of the reveal stage, Bob still cannot understand Alice's commitment, so if he could correlate then he would break the hiding property of c. On the other hand, B non-correlation at opening holds, since Alice commits first. Thus, her only option is to refuse to open her commitment, which, be definition, can only hurt her chances of being correllated. □

It is worth noting that while non-malleable commitments "with respect to opening" can be used for mutually independent announcement, the solution they offer is far more complex than this. This elegantly illustrates the point that the problem we solve requires less security than the problem that non-malleable commitments solve.

3.2 Mutually Independent Commitment

All the remaining protocols we present have a one-round reveal stage under the imperfect synchronization assumption. That is, each player sends only one message in the reveal stage, and the order does not matter: we allow the honest players to not wait to receive a message before sending one. Note that we do not assume that the messages are actually sent simultaneously: dishonest players can always wait for receipt of a message before sending theirs.

[5] This protocol can be modified to be based only on one-way functions, but this requires a 3-round commit stage: we simply use the construction of Naor [Nao91] to make a commitment scheme based on one-way functions, which requires a round to set up.

From this point on, when we refer to the number of rounds a protocol requires, we mean only the rounds in the commitment stage.

We present two protocols for mutually independent commitment: a two-round protocol based on the assumption that subexponentially hard one-way permutations exist, and a three-round protocol based on the assumption that 'dense' cryptosystems exist.

Two Round Protocol. A subexponentially hard one-way permutation is one for which there exists an $\epsilon > 0$ such that the permutation remains one-way even against adversaries that run in time 2^{n^ϵ}, where n is the security parameter. We note that, based on the current state of the art in factoring and discrete logarithm techniques, it is reasonable to assume that both RSA and exponentiation in a large prime-order subgroup of Z_p^* are subexponentially hard one-way permutation with some $\epsilon \leq 1/3$ (because the best known attacks against them take time $2^{O(n^{1/3}(\log n)^{2/3})}$).

Theorem 2. *If subexponentially hard one-way permutations exist, then there exists a two-round mutually independent commitment protocol.*

Proof sketch. Let c be a subexponentially secure non-interactive commitment scheme: i.e., one that is semantically secure against adversaries that run in time 2^{n^ϵ} for some $\epsilon > 0$, where n is the security parameter (such a commitment scheme can be constructed based on subexponentially hard one-way permutations). Assume that, for security parameter n, a commitment can be forced open in time 2^{n^δ}, for some $\delta > 0$ (this must be true for some δ, because one should be able to simply enumerate all the possible decommitment strings).

Let k be the security parameter for our scheme, and $K = k^{2\delta/\epsilon}$. The protocol is, again, very simple:

1. Alice commits to a using c with security parameter K.
2. Bob commits to b using c with security parameter k,

In the reveal stage, Alice and Bob reveal their values. It is clear that this scheme is complete, sound, hiding and binding. It is also clear that Alice cannot correlate her value to Bob's, since Alice is bound to her value before Bob commits to his. On the other hand, if Bob could correlate his value to Alice's, we could force open his commitment in time $2^{k^\delta} = 2^{K^\epsilon/2}$, and then use b to break the subexponentially strong semantic security of Alice's commitment in time $2^{K^\epsilon/2}$, which is a contradiction, because Alice's security parameter is K. □

Three Round Protocol. This protocol assumes the existence of dense, perfectly faithful cryptosystems. Following [DP92], a *δ-dense* cryptosystem is defined by modifying the definition of a secure cryptosystem [GM84] as follows: first, we add the requirement that a public key generated by the key generation algorithm is distributed uniformly over $\{0,1\}^{p(k)}$, for some polynomial p in the security parameter k; and second we require security for only a δ-fraction of public keys. It was observed by [DDP00] that the assumption of the existence of δ-dense cryptosystems, for a non-negligible δ, is equivalent to the existence of $(1-\epsilon)$-dense cryptosystems, for any negligible ϵ. We will actually need the latter. They can be constructed based on the ElGamal cryptosystem, for example.

Theorem 3. *If dense cryptosystems exist, then there exists a three-round mutually independent commitment scheme.*

Proof sketch. Let ϵ be a negligible function, and let (G, E, D) be a $(1-\epsilon)$-dense cryptosystem. Let $p(k)$ be the length of the public key for a security parameter k. Let c be a perfectly binding non-interactive commitment scheme. The commit stage is as follows:

1. Alice generates a random $p(k)$-bit string, R_A, and sends $c(R_A)$ to Bob.
2. Bob sends $c(b)$ to Alice, and sends a random $p(k)$ bit string R_B to Alice.
3. Alice computes $\mathrm{PK} = R_A \oplus R_B$, $C = E(\mathrm{PK}, a)$, and sends PK, C to Bob. Note that Alice does *not* open her commitment $c(R_A)$ at this step.

In the reveal stage, Bob opens his commitment to b, and Alice opens her commitment to R_A and reveals her value a and the random bits used to come up with C. Bob checks if R_A was revealed correctly, if PK indeed equals $R_A \oplus R_B$, and if the random bits were correct. If any of these checks fail, Bob rejects.

Completeness, soundness, binding, and B-hiding are easy to prove. A-hiding is proved as follows. Suppose B' is able to break the semantic security of the commitment of A. Then we will build a machine to break the semantic security of the dense cryptosystem. The machine will be given, as input, a public key PK and a ciphertext C. The machine will simulate A to B': it will commit to a random string R_A in the first round, and receive $c(b)$ and R_B in the second round. In the third round, it will ignore the first two rounds, and simply send the PK and C that were input to it. Note that B' should not be able to tell that $\mathrm{PK} \neq R_A \oplus R_B$—otherwise, it would be violating the hiding property of $c(R_A)$. Therefore, B' will "behave the same way" as with the true A, and thus would break the semantic security of the ciphertext C.

A-non-correlation at commitment is simple to prove: B' has no information about a at the time it has to commit to b.

B-non-correlation at commitment is proved as follows. Suppose A' can correlate to b. Then we will build a machine M that breaks the hiding property (semantic security) of $c(R_B)$, as follows. M receives a commitment c to some unknown value b. It will generate a key pair $(\mathrm{PK}', \mathrm{SK}')$ for the encryption scheme, and run A', simulating B to it by sending it c and a random string R_B in the second round. In the third round, A' will send PK to M. M will compute $R = \mathrm{PK} \oplus R_B$ (note that, if A' computed PK faithfully, then $R = R_A$), and $R'_B = \mathrm{PK}' \oplus R$. M will then rewind A' to the end of the first round, run it again, this time sending c and R'_B to A'. If A' again computes the public key faithfully, then it will encrypt a with PK', for which M knows the corresponding secret key SK$'$. This will allow M to recover a, which is correlated to the unknown committed value b, and thus will allow M to break the semantic security of c.

Of course, when M runs A' in this manner and A' does not compute the public key faithfully, then M will fail. However, if A' computes the public key faithfully with only a negligible probability, then the commitment of A' is invalid with all but a negligible probability, so A' is not correlating to b any better than a simulator who just outputs $\perp$ all the time. If, on the other hand, A' computes the public key faithfully with probability better than negligible, then M will break the semantic security of c with probability better than negligible, as well. □

3.3 Mutually Independent and Aware Commitment

We present two protocols for mutually independent and aware commitment. The first protocol, previously known in the folklore, uses non-interactive perfectly binding commitments and general zero-knowledge arguments of knowledge (arguments, as opposed to proofs, are sound only if the prover is computationally bounded, which suffices for our case). Specifically, to minimize the number of rounds, we use the 5-round protocol of [FS89], which is based on one-way permutations.

This protocol is not practical, because it uses general zero-knowledge proofs of NP statements. We present it here for didactic purposes: it clearly illustrates the notion of mutually independent and aware commitments.

Theorem 4. *If one-way permutations exist, then there exists an 11-round mutually independent and aware commitment protocol.*

Proof sketch. Let c be a commitment scheme.

The commit stage proceeds as follows:

1. Alice publishes a commitment $c(a)$ to her value.
2. Bob publishes a commitment $c(b)$ to his value.
3. Bob uses the [FS89] ZK argument of knowledge to prove to Alice that he knows how to open his commitment.
4. Alice uses the [FS89] ZK argument of knowledge to prove to Bob that she knows how to open her commitment.

This takes eleven rounds, since Bob can send $c(b)$ and the first round of his proof in the same message.

It is easy to show that this protocol is complete, binding, and sound. If it is not (say) A-hiding, then whatever B' breaks A-hiding can break the commitment scheme, because the zero-knowledge argument of knowledge can be simulated. Awareness follows simply by using the extractor for the proof of knowledge. □

The second protocol is much more efficient than the first. It requires just seven rounds, each of which takes only a few modular exponentiations. It relies on the hardness of discrete logarithms. In its simplest version, it assumes that there exists an easily indexable sequence of "safe" primes and generators $(p_1, g_1), (p_2, g_2), \ldots, (p_k, g_k), \ldots$, one pair for every value of the security parameter k, such that $p_i = 2q_i + 1$ (where q_i is a prime), g_i is a generator of the subgroup of order q_i in $Z_{p_i}^*$, and discrete logarithm is hard in that subgroup.[6] To simplify notation, we will assume the security parmater k is fixed, and will simply use p, q, g in place of p_k, q_k, g_k when describing our protocol.

With a loss of efficiency, our protocol can be modified to be based on general assumptions rather than the hardness of discrete logarithms.

Theorem 5. *Assuming the hardness of discrete logarithms, there exists a seven-round mutually independent and aware commitment scheme.*

Proof sketch. For clarity, we will present our protocol for commitments to single-bit messages first, and then explain how it can modified for longer messages. Let H be a hardcore predicate for discrete log (in particular, [BM84] prove that the sign of the exponent minus $(p-1)/2$ is hardcore).

Let C denote a perfectly hiding trapdoor commitment scheme based on the discrete logarithm assumption. To be specific, we use the scheme of Pedersen [Ped91], in which one has two bases (generators), g and $h = g^\alpha$, and commits to a value v by publishing $g^v h^r$, for a random r. The scheme is binding because decommitting in two different ways allows one to find α. On the other hand, the scheme is trapdoor because knowing α allows one to decommit to any v'.

The commit stage of our protocol proceeds as follows.

1. Alice randomly generates an element g_a of order q and $\alpha \in Z_q$, and computes $h_a = g_a^\alpha$. She sends (g_a, h_a) to Bob, to be used by him as bases for the Pedersen commitment scheme.
2. (a) Bob likewise generates g_b, β and h_b, which he sends to Alice to be used by her as bases for the Pedersen commitment scheme.
   (b) Bob generates k_b, k_b' such that $H(k_b) \oplus H(k_b') = b$, and sends g^{k_b} and $g^{k_b'}$ to Alice.
   (c) Bob generates a random $r_b \in Z_q$, computes g^{r_b}, and then commits to g^{r_b} using Pedersen commitments with bases g_a and h_a. He sends the resulting commitment $C_a(g^{r_b})$ to Alice.
3. (a) Alice generates k_a and k_a' such that $H(k_a) \oplus H(k_a') = a$, and sends $g^{k_a}, g^{k_a'}$ to Bob.
   (b) Alice generates a random r_a, and, just like Bob, commits to g^{r_a} using Pedersen commitments with bases g_b and h_b. She sends the resulting commitment $C_b(g^{r_a})$ to Bob.
   (c) Alice generates and sends to Bob a random $c_a \in Z_q$.
4. (a) Bob generates and sends to Alice a random c_b in Z_q.
   (b) Bob decommits g^{r_b} from $C_a(g^{r_b})$ and sends the decommitment to Alice.

[6] This assumption can be relaxed by having the parties provide the parameters to each other; moreover, we do not need primes of the form $2q_i + 1$; primes of the form $k_i q_i + 1$, for sufficiently long prime q_i, would suffice. For the sake of clarity, however, we do not present our protocol that way.

(c) Bob computes $d_b = c_a k_b + r_b$, and sends d_b to Alice.

5. (a) Alice checks the decommitment of g^{r_b} and verifies that $g^{d_b} = (g^{k_b})^{c_a} g^{r_b}$. If the checks fail, she outputs "reject" and stops.
   (b) Alice decommits g^{r_a} from $C_b(g^{r_a})$ and sends the decommitment to Bob.
   (c) Alice computes $d_a = c_b k_a + r_a$, and sends d_a to Bob.
   (d) Alice sends α to Bob.
6. (a) Bob checks the decommitment of g^{r_a} and verifies that $g_a^d = (g^{k_a})^{c_b} g^{r_a}$. If the checks fails, he outputs "reject" and stops.
   (b) Bob also checks that $g_a^\alpha = h_a$. If not, he outputs "reject" and stops.
   (c) Bob sends k'_b and β to Alice.
7. (a) Alice checks that $g_b^\beta = h_b$. If not, she outputs "reject" and stops.
   (b) Alice checks k'_b received from Bob against $g^{k'_b}$ that was sent to her in step 2. If they do not agree, she outputs "reject" and stops. Otherwise, she outputs "accept."
   (c) Alice sends k'_a to Bob.
8. Bob checks k'_a against $g^{k'_a}$ that was sent to him in Step 3. If the agree, he outputs "accept." Otherwise, he outputs "reject."

In the reveal stage, Alice reveals k_a and Bob reveals k_b. The value a is then calculated as $H(k_a) \oplus H(k'_a)$. b is calculated similarly.

Intuition. The following may help explain what is happening in this protocol with respect to Alice's commitment (Bob's commitment is, of course, similar). In step 3(a), Alice commits to her bit a by splitting it into two parts, k_a and k'_a. In steps 3(b), 4(a) and 5(c), she proves knowledge of k_a using a Schnorr [Sch89] three-round proof of knowledge for discrete logarithms. The only difference from the Schnorr proof is that the initial message of Alice's proof of knowledge is committed using the trapdoor commitment that Bob set up for Alice in step 2(a), and Alice reveals that message in step 5(b), at the end of the proof of knowledge. Bob reveals the trapdoor for the commitment scheme in step 6(c). Only after Alice gets the trapdoor does she reveal k'_a in step 7(c).

The reason for not using Schnorr's proof of knowledge directly is that it is not known to be simulatable. However, if the simulator knows Bob's trapdoor, then we can simulate the proof. As we explain in detail below, Bob can refuse to reveal the trapdoor, but then the simulator does not have to reveal k'_a. (We note that the idea of revealing the trapdoor to allow the simulation to go through has been used before in a number of protocols, and seems to have first appeared in [CDM00].)

Security. It should be clear that the protocol is complete, sound, and binding. Awareness is fairly easy to show: the extractor (say, for Bob) need only run the protocol through where c_a is sent by Alice, and repeatedly try substituting different challenges. If Bob ever answers two different challenges, k_b can be recovered. Then, since Bob reveals k'_b in the protocol, b can be determined.

The hiding property is a little more difficult to demonstrate. Suppose there is a B' which can find A's secret values. There are two cases.

case i: In this case, with a non-negligible probability, when Bob does not give the correct β, he still can still distinguish whether A was committing to a 0 or a 1. In this case, we can break the hard-core predicate. Suppose we are given $z = g^x$ and we are asked to find $H(x)$ with good probability. Then we first of all randomly choose k_a and run the protocol as Alice faithfully, except that we give z in place of $g^{k'_a}$. If Bob returns the correct β we output a coin flip. Otherwise, we get Bob's guess a and output $a \oplus H(k'_a)$. Note that since we either output a coin flip, or Bob doesn't return the correct β, we never actually reach step 7, so it doesn't matter that we don't know x.

case ii: In this case, when Bob does not give the correct β, he can only distinguish with negligible probability. Thus, if Bob does give β correctly, he must be able to distinguish. So, since he is able to distinguish with non-negligible probability, he must give β with non-negligible probability. Thus, we run honestly until Bob gives β. If it is incorrect, we output a coin flip. Otherwise, we rewind and give z in place of g^{k_a} and fake the proof of knowledge (which we can do now that we have the trapdoor.) If Bob then completes the protocol, we take his guess a and output $a \oplus H(k'_a)$. Otherwise, we output a coin flip. This will give us an edge in guessing the most significant bit of the discrete logarithm of z.

Longer messages. In order to extend this protocol to longer messages, there are two techniques. The first is the obvious one: run the protocol many times in parallel (though we can collapse some rounds together: for instance, only one pair g_a, h_a is needed). We can do better than this by relaxing our assumptions and assuming that the discrete logarithm problem has more hardcore bits. That is, if A's secret is n bits long and $H_n(x)$ returns n hardcore bits x, then we simply modify the protocol so that Alice generates k_a and k'_a such that $H_n(k_a) \oplus H_n(k'_a) = a$. □

4 Acknowledgements

We would like to thank the anonymous referees for their detailed comments.

Moses Liskov was supported by a grant from the Merrill Lynch corporation. Anna Lysyanskaya was supported by an NSF graduate fellowship, the Lucent Technologies GRPW program, and the Merrill-Lynch grant given to R.L.Rivest. The work of Leonid Reyzin was performed while at MIT, and was supported by a grant from the NTT corporation. Adam Smith was supported by U.S. Army Research Office Grant DAAD19-00-1-0177.

References

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, December 1991.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, November 1984.

[BMM99] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer-Verlag, 15–19 August 1999.

[CDM00] Ronald Cramer, Ivan Damgård, and Philip MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Public Key Cryptography (PKC 2000)*, pages 354–372. Springer-Verlag, 2000.

[CF01] R. Canetti and M. Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, Lecture Notes in Computer Science. Springer-Verlag, 19–23 August 2001.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th IEEE Symposium on Foundations of Computer Science*, pages 383–395, 1985.

[CR87] Benny Chor and Michael Rabin. Achieving independence in logarithmic number of rounds. In *Principles of Distributed Computing (PODC 87)*, pages 260–268. ACM, 1987.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, 23–27 August 1998.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM*, 30:391–437, 2000.

[DDP00] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all np relations. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Automata Languages and Programming: 27th International Colloquim (ICALP 2000)*, volume 1853 of *Lecture Notes in Computer Science*, pages 451–462. Springer-Verlag, July 9–15 2000.

[DIO98] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, Dallas, Texas, 23–26 May 1998.

[DKOS01] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. In Birgit Pfitzmann, editor, *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 40–59. Springer-Verlag, 6–10 May 2001.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[FF00] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In Mihir Bellare, editor, *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*. Springer-Verlag, 20–24 August 2000.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–545. Springer-Verlag, 1990, 20–24 August 1989.

[Gen95] Rosario Gennaro. Achieving independence efficiently and securely. In *Principles of Distributed Computing (PODC 95)*, pages 130–136. ACM, 1995.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. Knowledge complexity of interactive proofs. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, 6–8 May 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[KOS] J. Katz, R. Ostrovsky, and A. Smith. Personal Communication.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1992, 11–15 August 1991.

[Sah99] Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, New York, October 1999. IEEE.

[Sch89] C. P. Schnorr. Efficient identification and signatures for smart cards. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *Lecture Notes in Computer Science*, pages 688–689. Springer-Verlag, 1990, 10–13 April 1989.

[TW87] Martin Tompa and Heather Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *28th Annual Symposium on Foundations of Computer Science*, pages 472–482, Los Angeles, California, 12–14 October 1987. IEEE.