MUTUALLY INDEPENDENT COMMITMENT

Moses Liskov, Anna Lysyanskaya, Silvio Micali, Leo Reyzin, Adam Smith
mliskov@, anna@, silvio@, reyzin@, asmith@theory.lcs.mit.edu
MIT Lab for Computer Science

Abstract

We describe a new kind of commitment scheme in which two parties commit to values in a commitment stage, at the end of which we are assured that the values they have committed to cannot be correlated to one another. We call this new primitive *mutually independent commitments.* We present three mutually independent commitment schemes which handle single bit commitments, and which are computationally hiding and perfectly binding.

1 Introduction

COMMITMENT SCHEMES. A commitment scheme consists of a method C which produces "commitments" to an input x based on some randomness. The commitment is denoted $C(x)$ though we note that C may not be a traditional function, rather $C(x)$ may be arrived at through a protocol. (For the immediate discussion, we will consider only non-interactive commitments.) During this protocol, the committing party learns some side information p, which together with x can verify that the commitment $C(x)$ is indeed a commitment to x. Commitment schemes need the following two properties in order to be secure.

- *Hiding* It should be hard, seeing only the output $C(x)$, to learn anything about x.
- *Binding* It should be hard to find a quadruple (x, x', p, p') such that $y = C(x)$ can be shown to be a commitment to x with proof p, and can be shown to be a commitment to x' with proof p' where $x \neq x'$.

A SIMPLE CORRELATION ATTACK. These properties are not sufficient to infer that anyone seeing a commitment cannot extract any useful information from it. Indeed, someone seeing a commitment cannot extract information about the secret value x, but information can be gleaned in other ways. As a simple example, suppose Alice and Bob each wish to commit to two values, and Alice will commit to hers first. However, Bob may secretly want to commit to the same thing Alice committed to. Then, if $c_a = C(a)$ is Alice's commitment, Bob can just give c_a as his own commitment. Now, the value Bob has committed to is the same as the value Alice has committed to. Later, when the commitments are opened, if Alice goes first, she reveals her proof p and her secret value a. Then, Bob can claim that p was also *his* proof and that he committed to a as well.

A STRONGER CORRELATION ATTACK. Some may not find this attack very compelling, because Bob's commitment is exactly the same as Alice's, which means that checking for this would be very easy. So to motivate the problem further, consider the following scheme for committing to a bit [P91]:

Public Parameters: p, q, g such that $p = 2q + 1$, p and q are both prime, and $g \in \mathbb{Z}_q^*$, and separate random value $h \in \mathbb{Z}_q^*$ (where the discrete logarithm of h to the base g is not known to anyone).

Commitment Phase: When a is a bit, Alice computes $C(a, r) = g^r h^a \bmod p$.

Revealing Phase: Alice reveals a and r. Anyone can check that $g^r h^a$ was her published commitment.

In this scheme, suppose that Bob sees that Alice's commitment is x. Bob can commit to the same value as Alice by computing a random value r' and publishing as his commitment the value $xg^{r'}$. Now, when Alice reveals a and r, Bob can reveal a and $r + r'$. Moreover, Bob's copied commitment looks like a totally random commitment.

MOTIVATION. It is not clear immediately that this is an important problem, so we consider the following example. Suppose Alice and Bob are bidding in an auction, but in this auction they each only get one chance to bid, and a commitment scheme is used to prevent them from changing their bids later. The rules are that whoever bids more for the item buys it at their committed-to price, and anyone refusing to open their commitment pays a large penalty. If Bob, seeing only Alice's commitment to her bid, can produce a commitment to a bid of one cent more than Alice's commitment, then he will certainly win the auction, and do so at the lowest possible price.

AN INADEQUATE SOLUTION. The straightforward solution to this problem is somewhat inadequate for protocol design. That solution is that during commitment, Alice commits first and Bob commits second, but during revealing, Bob reveals *first* and Alice reveals second. Once Bob opens his commitment, we know that whatever value he committed to is independent of Alice's committed value. (Otherwise, the hiding property of the commitment scheme would be violated.) This protocol, taken all at once, forms a scheme for producing mutually independent *announcements*. That is, once the reveal stage is completed, the revealed values are independent of one another. However, at the end of the commitment stage, there is no such guarantee.

In protocol design, however, it may be important to have a guarantee that, if the commitment stage is successfully completed, the committed-to values are guaranteed to be independent at that point, regardless of whether the revealing protocol is ever run.

Note that any mutually independent commitment scheme gives a mutually independent announcement protocol; once the committed-to values are known to be independent, the parties simply reveal their commitments. By the binding property, these plaintexts are the same as the committed-to values, and so they must also be mutually independent.

1.1 Structure of this paper

First, we describe the definition of a mutually independent commitment scheme. In this paper, we are concerned only with the (1) schemes for committing to a single bit and (2) schemes which are perfectly binding / computationally hiding. Finally, we are of course concerned with a communication model that does not allow perfect synchronization (or the problem could be trivially solved by having each party publish their commitments simultaneously).

Next, we go on to describe a mutually independent commitment scheme based on the assumption of dense, semantically secure public key cryptosystems with a 3-round commitment protocol.[1]

Next, we describe a stronger property that may be desirable, and present two mutually independent commitment schemes with that property. The first of these schemes will be based on the discrete logarithm assumption and will have a 4-round commitment protocol, while the other will be based on the existence of one-way permutations but will have a 7-round commitment protocol.

Finally, we give a preliminary discussion about the issues that arise when

[1]In our schemes, there will be two separate phases: the commitment phase and the reveal phase. We will measure the efficiency of the protocol by the number of rounds needed in the commitment protocol, since in many cases the reveal protocol is very simple, and since we may sometimes have commitments which are never opened.

we try to extend these schemes to ones which allow commitments to longer strings.

2 Prior Work

3 Acknowledgements

4 Definitions

A *mutually independent commitment scheme* is a scheme involving two parties, A and B, which will be modeled by interactive probabilistic polynomial-time Turing machines. The scheme consists of two protocols, *commitment* and *revealing.*

For ease of notation, we will use the notation A' to denote an arbitrary PPT TM taking the place of machine A in the protocol, and similarly B' for B.

Let us say that $(T, p_A, p_B) = AB(\text{`commit}', a, b)$ [2] be the output of the commitment protocol, where T is the transcript, a public output, and p_A and p_B are private outputs of A and B, respectively.

Let us say that $(c_A, c_B, v_A, v_B) = AB((\text{`reveal}', T), (a, p_A), (b, p_B))$ is the output of the revealing protocol. Here, v_A and v_B are the revealed values that A and B committed to, respectively, and c_A and c_B are boolean values that say whether A (or B) accepts B's (or A's) value, respectively.

The scheme should have the following properties:

- *Completeness:* If A and B are both honest, then the commitment protocol produces an output, and if $(T, p_A, p_B) = AB(\text{`commit}', a, b)$, the probability that if $(c_A, c_B, v_A, v_B = AB((\text{`reveal}', T), (a, p_A), (b, p_B))$, then with probability 1, $c_A = c_B = 1$ and $v_A = a$ and $v_B = b$.

- *Perfect A-Binding:* No TM A' can succeed in producing a transcript T with the honest B such that in the reveal stage with the honest B, A' can cause B to accept either of two different values. Formally,

  $\forall A', \forall \mathcal{B}, \Pr[b \leftarrow \mathcal{B}, (T, p_A, p_B) \leftarrow A'B(\text{`commit}', \cdot, b), (c_A, c_B, v_A, v_B) \leftarrow A'B((\text{`reveal}', T), (p_A, 0), (b, p_B)), (c'_A, c'_B, v'_A, v'_B) \leftarrow A'B((\text{`reveal}', T), (p_A, 1), (b, p_B) : v_A = 0 \wedge v'_A = 1 \wedge c_B = c'_B = 1] = 0$

[2]Our notational convention here will be that the output of $M_1M_2(P, S_1, S_2)$ will be the outputs of the protocol run between M_1 and M_2, two (randomized) TMs, where P is input given to both M_1 and M_2, S_1 is private input given only to M_1, and S_2 is similarly given only to M_2.

- *Perfect B-Binding:* Similarly for the above definition, but B being the dishonest party.
- *Computational A-Hiding:* For all B', if after participating in a commitment protocol with A, then produces a value z, the probability that $a = z$ is at worst negligibly larger than $1/2$.
- *Computational B-Hiding:* Similarly for the above definition, but with A being the dishonest party.
- *Non-A-correlation:* For all binary relations R on pairs of bits, for all distributions $\mathcal{A}$, and for all B', if a is generated from $\mathcal{A}$, and (T, p_A, p_B) is the output of $AB'(\text{`commit}', a, \cdot)$ and $b$ is a bit such that $(c_A, c_B, v_A, b)$ is the output of $AB'((\text{`reveal}', T), p_A, p_B)$ then the probability that $R(a, b)$ and $c_a = 1$ is only negligibly better than the probability that $R(a', b)$ where a' is generated independently of a from the distribution $\mathcal{A}$.
- *Non-B-correlation:* As for Non-A-correlation, but with A being the dishonest party instead of B. We should note that in our protocols, Non-B-correlation is trivial to show since A will commit to their value in the commitment protocol before B does.

Note that the schemes we present are only set up for commitments to single-bit values, but the defintions we present are flexible and are capable of handling larger values.

5 First Protocol

For this protocol, we require a semantically secure public key cryptosystem which is *dense*,[3] that is, that a randomly chosen string of the right length is a valid public key.

The commitment protocol consists of three rounds:

STEP 1. B generates a random value R_1 and sends a commitment to R_1 to A.

STEP 2. A sends a commitment $C(a)$, and a random value R_2.

[3]What we want, specifically, is that any string of the appropriate length be a valid public key, and moreover that the distribution on public keys chosen through the key generation protocol be the uniform distribution on strings of the appropriate length. This seems like a definition that would have come up before, but we have not found it in the literature. It is similar to the definition given in [DP92], but different.

STEP 3. B sets $PK_B = R_1 \oplus R_2$, and sends R_1 and $E_B(b)$ to A.

The revealing protocol simply consists of B opening his commitment $E_B(b)$, providing the random bits used, and of A and B opening their commitments $C(a)$ and $C(R_1)$.

In order to show the security of this scheme, we need only assume the security of the cryptosystem and the (perfectly binding) commitment scheme used in it.

That this scheme is complete, binding, and hiding is fairly straightforward. Furthermore, Non-B correlation is trivial. To show that it has Non-A correlation, suppose there was a B' that succeeded in creating a commitment to a value correlated to a, then we can create an algorithm for breaking the GM security of the cryptosystem. The primary trick in the proof is that this algorithm C interacts with B until it sees R_1, then it rewinds the protocol and sends a different R_2 so that $R_1 \oplus R_2$ is a public key for which C knows the secret key. Then, C simply decrypts the value $E_B(b)$, and uses this information to recover information about a, thus violating the hiding property of the commitment scheme used by A.

However, this protocol lacks one property which we may sometimes wish to have. That is, when the commitment stage of the protocol completes, we know that the values A and B committed to are independent of one another, but we don't know for sure whether A and B know how to actually open them. If they proceed honestly they will actually succeed in the reveal stage. However, if they are dishonest, they might not. Thus, we propose additional security properties which we may desire.

- *A-Extractibility:* There exists an extractor E_A such that for any A', A' and E_A engaging together in the commitment protocol (with E_A playing the part of B, and with E_A capable of "rewinding" A',) produce a transcript T and private output p_B such that (1) T is distributed just as it would be if A' were interacting with an honest B, and that (2) with overwhelming probability, the value p_B is such that if A' chooses to open its commitment at all in a valid way, p_B is the value it can open to.
- *B-Extractibility:* Similar to A-Extractibility, except that we use an extractor E_B in place of the A party, and a dishonest B' in place of B.

We call any mutually independent commitment scheme with these two properties a *mutually independent and aware commitment scheme*, since these properties ensure that the parties are aware of their commitments.

6 Second Protocol

We now exhibit the first of the two mutually independent and aware commitment schemes. The first scheme requires only a 4-round commitment stage, but relies on the hardness of the discrete logarithm problem.

Public parameters: a large prime $p = 2q + 1$ where q is also prime, and a generator g of Z_q^*.

STEP 1. A privately generates a value $k \in Z_q$ such that if A wants to commit to 0, $k \in \{1, \ldots, (q-1)/2\}$ and otherwise, $k \in \{(q+1)/2, \ldots, q-1\}$. Then, A generates n values $r_{A_1}, \ldots, r_{A_n}$, and sends the $2n$ values $g^{k-r_{A_1}}, g^{r_{A_1}}, \ldots, g^{k-r_{A_n}}, g^{r_{A_n}}$. Call these values $\alpha_{1,0}, \alpha_{1,1}, \ldots, \alpha_{n,0}, \alpha_{n,1}$.

STEP 2. B similarly generates a value k' and n values $r_{B_1}, \ldots, r_{B_n}$, and generates $\beta_{1,0}, \ldots, \beta_{n,1}$ as A does in step 1. During this step, B also produces a challenge bit string of length n: $S_{B_1} S_{B_2} \ldots S_{B_N}$.

STEP 3. A checks that for all $i \in [1, n]$, the value $\beta_{i,0}\beta_{i,1}$ is the same. A also produces a challenge string S_A of length n, and for each $i \in [1, n]$ produces the discrete logarithm of $\alpha_{i,S_{B_i}}$, and sends these to B.

STEP 4. B checks that for all $i \in [1, n]$, the value $\alpha_{i,0}\alpha_{i,1}$ is the same, and verifies the discrete logarithms provided by A, and answers A's challenge similarly.

STEP 5. A checks that B answered the challenge correctly.

In the revealing protocol, A and B reveal their secret values k and k' and these are verified, and v_A and v_B are calculated based on k and k'.

The idea here is that whether the secret discrete logarithm of $\alpha_{i,0}\alpha_{i,1}$ is in the top or bottom half of Z_q is a hard-core predicate of the discrete logarithm function, which provides the hiding property and the noncorrelation properties. The discrete logarithm problem itself provides the property of perfect binding we require. Finally, the "cut-and-choose" structure of the protocol makes the A- and B-extractibility fairly easy to demonstrate.

One issue we will bring to light in the extractibility proof is this. Suppose that A' is just like A except that it will never reveal the discrete logarithm of $\alpha_{i,j}$ for some specific i, j (until the revealing protocol, if A' chooses to open her commitment). Then with probability 1/2, A' makes it through the protocol, and the canonical extractor E_A would run into problems when the second time around, A' refused to open the challenge. However, E_A could just try new random challenges until it got A' to respond to its challenge and the challenge is different in even one position. Then, E_A can reconstruct k, which reveals a.

This protocol is efficient (only 4 rounds in the commitment phase, since step 5 does not necessarily have to take place, as it can be verified by anyone), but relies on a fairly strong security assumption. Next, we will demonstrate a protocol which relies on weaker assumptions, but which requires more rounds.

7 Third Protocol

This protocol is a bit more complicated than the previous two. It involves, in the commitment protocol, a zero knowledge proof of knowledge. The best scheme we know of to give a zero-knowledge proof of knowledge in constant rounds depends on the existence of a perfectly hiding commitment scheme, so we assume that in addition to the existence of one-way permutations (which is sufficient for perfectly binding commitments). This zero-knowledge proof of knowledge requires 5 rounds, and the prover speaks first. For brevity, we will call the rounds of communication in the proof $< BA >_1, < BA >_2$, et cetera. In [FS89] the authors show how to give a zero knowledge argument of knowledge in 5 rounds based on one-way functions. The difference between a "proof" and an "argument" is that in a proof, the prover may be unbounded computationally, while in an argument, the prover need not be. Since in our scheme we assume both players are computationally bounded, it suffices to rely on a zero-knowledge argument.

Here is the protocol.

STEP 1. A generates $2n$ values $a_{1,0}, \ldots, a_{n,1}$ such that for all $i \in [1, n]$, $a_{i,0} \oplus a_{i,1} = a$ and computes $\alpha_{i,j} = C(x_{i,j})$, where C is the perfectly hiding commitment scheme.

STEP 2. B publishes a commitment $C(b)$ to b. B and A start the zero knowledge proof of knowledge that B knows his commitment b and knows how to open it. B sends $C(b)$ and $< BA >_1$ to A.

STEP 3. A sends $< BA >_2$ to B.

STEP 4. B sends $< BA >_3$ to A.

STEP 5. A sends $< BA >_4$ to B.

STEP 6. B sends $< BA >_5$ to A and sends a challenge s to A, where s is a bit string of length n.

STEP 7. A opens for B the values a_{i,s_i} for each $i \in [1, n]$.

STEP 8. B checks that A's opened values from step 7 were valid.

In the revealing protocol, A opens all the remaining commitments and B opens his commitment $C(b)$.

The completeness and binding properties for this protocol are clear. The hiding properties are not so obvious: to prove that A's value is hidden, we employ a hybrid argument. To prove that B's value is hidden, we use the simulator from the zero-knowledge proof. Extractibility is fairly easy; to extract B's value we simply use the extractor from the proof of knowledge. To extract A's value, we need only rewind B's challenge and provide a different random one. With a good probability, A will provide both $a_{i,0}$ and $a_{i,1}$ and then we can XOR these to get a. Proving that A cannot correlate his value to B's is trivial. Proving that B cannot correlate to A's is done by using his ability to do this with a hybrid argument to construct a machine that breaks the hiding property of the underlying commitment scheme.

Invalid commitments. One important difference between the discrete logarithm protocol (the "second protocol") and this proocol is that in this protocol it is possible for A to get through the commitment stage with an invalid commitment. In this protocol, A's commitment is only valid if for every i, $a_{i,0} \oplus a_{i,1}$ produces a single value a. If some produce one value and some produce another, the commitment is invalid. This possibly undesirable property of the protocol can be fixed by having A simply commit to $C(a)$ and perform a ZKPOK that A knows how to decommit that value. It is not obvious that we can interleave that proof with the other proof, so this would add rounds to the protocol.

However, we have an intuitive argument that suggests that this kind of behavior on the part of A is not really a matter for concern. We start by remarking that regardless of the scheme involved, either player always has the option of refusing to cooperate in the reveal stage of the scheme, which effectively gives the players three options: commit to 0 and reveal it, commit to 1 and reveal it, or refuse to reveal. If we add to this the possibility that a player can also commit in an invalid way and reveal it, we have four options, but we can treat an invalid commitment and the refusal to reveal as the same result for any player. The only problem with this is that it seems as if the two situations are different, but in a sense, every situation boils down to this: each player is asked to pick a value, 0 or 1, and commit to it, and then open it later. Any deviation from this is a refusal to cooperate in the protocol, and furthermore is only detected during the reveal stage. Thus, if we just treat the two situations the same, no honest player is losing any of their options, and we are not relaxing the ability of the scheme to notice this lack of cooperation.

8 Conclusion

We have described a protocol for mutually independent commitments which has a 3-round commitment phase, the security of which is based on the existence of a secure public key cryptosystem.

We have also described two protocols for mutually independent and aware commitments, one which is a four-round protocol based on the discrete logarithm assumption, and one which is a seven-round protocol based only on a perfectly binding commitment scheme and general zero-knowledge proofs of knowledge.

9 References

[DP92] A. De Santis and G. Persiano. Zero Knowledge Proofs of Knowledge Without Interaction (Exended Abstract). In FOCS 1992.
[FS89] U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. In *Advances in Cryptology – CRYPTO '89*, 1990.
[P91] T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology – CRYPTO '91*, 1991.