13. Commitment Schemes

Commitment schemes constitute a core cryptographic primitive for building larger protocols. They consist of two subprotocols, a *commit* protocol and an *open* protocol: In the commit protocol, one participant commits to a value, i.e., it fixes it so that it cannot be changed later, while still keeping the value hidden from the other participants. In the open protocol, the value is shown to the other participants. Typically, there is only one other participant, i.e., both commit and open constitute two-party protocols. In addition to the obvious correctness property, commitment schemes should satisfy two security properties: They should be *binding*, i.e., the committer cannot change the value after the commit protocol, and *hiding*, i.e., the recipient does not learn anything about the value during the commit protocol.

A common non-digital example of commitments is the following: Alice writes something on a piece of paper (without allowing Bob to see what is written), puts the paper inside an envelope and puts the envelope on the table, which completes the commit protocol. Now she is bound to the written value, but Bob cannot look inside the envelope. Later they can open the envelope and verify the value. A special case of commitment schemes are bit commitment schemes which only allow for committing to one bit. Indeed the early constructions only worked bit by bit; now, however, many schemes are known for committing to longer messages more efficiently than by proceeding bit by bit; this is sometimes called string commitment.

13.1 Definition of Commitment Schemes

Commitment schemes are of the following structure: The parameters are a message space $\mathcal{M}$ and a security parameter n. There are two roles: a so-called *committer* and a *recipient*. There are two subprotocols: The *commit* protocol, where the committer program obtains one input $m \in \mathcal{M}$, the message to be committed to. The recipient program obtains no input, except possibly for a start signal. Neither program makes an output except for a value $acc \in \{ok, \downarrow\}$ indicating whether the commit protocol terminated correctly. (But they store some values for the later opening protocol.) In the *open* protocol the same committer and recipient as in a previous commit protocol take part. The programs do not need inputs, but the recipient program will output either $(accept, m^*)$ for a message $m^* \in \mathcal{M}$ or *reject*. This means that it either accepts that the commitment contained the message m^*, or decides that the committer did not open the commitment correctly. There are three requirements. The first one is simply that everything should work in the faultless case; the other two are the real security requirements.

(a) **Correctness:** If both the committer and the recipient are honest and execute commit and open in succession, then $m^* = m$. More precisely, both programs output $acc = ok$ in the commit protocol, and the output of the recipient program in open is $(accept, m)$, where m is the message that was input to the committer program in the commit protocol.

(b) **Binding property** (or "committing property"): Even if the committer is dishonest, he cannot open one commitment in two different ways. More formally, the notions of "can open" can be represented as follows: One first executes the commit protocol with the cheating committer and the correct recipient. If the recipient program outputs $acc = ok$, the opening protocol is executed twice, both times starting from the state after commit. The cheating committer gets an additional bit $b = 0$ the first time and $b = 1$ the other time to distinguish how it should open. We say that the cheating committer was successful in a particular run if the recipient makes outputs $(accept, m)$ and $(accept, m')$ for two different messages $m \neq m'$ in the two executions of *open*.

(c) **Hiding property** (or "secrecy"): The commit protocol should not give the recipient any information about the message m. This is further formalized in the same way as for an encryption scheme.

The binding and hiding property can each be required with either information-theoretic or computational security, i.e., for computationally unbounded adversaries, or for adversaries that are constraint to running in probabilistic polynomial-time in the security parameter. The binding property is a requirement of the recipient, the hiding property a requirement of the committer. Consequently, the recipient is assumed to be honest in (b), and the committer is assumed to be honest in (c), whereas the other party may potentially deviate from the protocol to gain an advantage.

Some Special Cases and Notation

- Some constructions additionally have a subprotocol *initialization* that can be executed once for many commitments to improve the efficiency of the actual commit protocol.
- A commitment scheme is said to be with *non-interactive opening* if only one message is sent in the open protocol (from the committer to the recipient).
- A scheme is said to be with *public verification* if the final decision of the recipient whether he accepts a message, and if yes which, only depends on the traffic, i.e., the messages sent and received, and not on internal secrets. In this case, other parties who saw the traffic can also decide whether the recipient should accept.
- A commitment scheme is called *non-interactive* if committing is non-interactive in the same sense as opening. The one message sent in commit is then called the commitment.

  In this case a committer can, e.g., make a commitment and broadcast it, so that he is committed with respect to many recipients.

Almost all well-known commitment schemes are with non-interactive opening and public verification. Non-interactive committing is not so prevalent, but many people nevertheless tend to think primarily of non-interactive schemes.

13.2 An Impossibility Result

All the following protocols are either information-theoretically binding or information-theoretically hiding, but not both. It is not hard to see that one cannot construct commitment schemes that satisfy both binding and hiding in an information-theoretic manner.

Theorem 13.1 *No protocol can fulfill both hiding and binding in an information-theoretic manner.* □

Proof. (Sketch) The intuitive reason is that after an execution of commit, there are either two possibilities to open the commitment, then it is not information-theoretically binding, or only one, then it is not information-theoretically hiding. In the non-interactive case, this intuitive argument can simply be formalized by considering the messages that the committer can send in open. In the general case, "the commitment" is modeled by the traffic in the commitment protocol. We only need to consider resulting values *traffic* from correct executions of the commit protocol. Information-theoretic secrecy of the message m implies in particular that it should be impossible to guess m significantly better than with a-priori probability from *traffic* (because *traffic* is part of the recipient's view). In other words, this means that any message m' could lead to the same traffic in an execution of the commit protocol, and the probability for this is equal for all messages. Now we consider a cheating committer exploiting this fact: He can first execute the commit protocol correctly with one message m, then choose another message m' and find an execution that is consistent with both m' and the traffic from the first execution. By executing open starting with his final state from either the m-execution or the m'-execution, he can then (by the correctness) get the recipient to accept either m or m'. ■

13.3 Information-Theoretically Binding Schemes

We will see three well-known commitment schemes in the following, two information-theoretically binding ones in this section and an information-theoretically hiding one in the next section.

13.3.1 Commitment Schemes from Asymmetric Encryption

Any CPA-secure asymmetric encryption scheme can be used to build a commitment scheme in the following way. We assume that the message space of the encryption scheme comprises the desired one of the commitment scheme. Note that for this scheme, the random bits used by the key generation algorithm Gen and by the encryption algorithm E are important, thus we equip these functions with an additional argument representing this random string. Note that then the encryption scheme is secure only if the randomness was chosen uniformly and if it does not become known to the adversary.

- In *commit*, the committer generates a key pair

  $$(sk, pk) := \mathsf{Gen}(n; r_1),$$

  where Gen is the key generation algorithm of the given encryption scheme, n the same security parameter as in the commitment scheme, and r_1 a random string. He encrypts his message m into a ciphertext

  $$c := \mathsf{E}(pk, m; r_2),$$

  using a second random string r_2, and sends

  $$com := (pk, c)$$

  to the recipient.

- In *open*, the committer sends
$$(m, r_1, r_2)$$
to the recipient. The recipient regenerates the key pair (sk, pk) using r_1 and recomputes the ciphertext $c := \mathsf{E}(pk, m; r_2)$. He verifies that his result (pk, c) equals the commitment com, and that m belongs to the message space.

Now we show that this scheme fulfills the three requirements.[1]

Theorem 13.2 *The commitment scheme just described is information-theoretically binding, and it is computationally hiding if the underlying asymmetric encryption scheme is CPA-secure.* □

Proof. Correctness of the above scheme is obvious, and the *hiding property* follows immediately from the CPA-security of the encryption scheme.

To see that the *binding property* holds we assume that a cheating committer can send values (m, r_1, r_2) and (m', r_1', r_2') such that recipient accepts in both cases for the same commitment $com = (pk, c)$. The recipient's first test guarantees that key generation using r_1 gives a key pair (sk, pk). The correctness of the encryption scheme now guarantees that
$$\mathsf{D}(sk, c) = \mathsf{D}(sk, \mathsf{E}(pk, m, r_2)) = m,$$
but also
$$\mathsf{D}(sk, c) = \mathsf{D}(sk, \mathsf{E}(pk, m', r_2')) = m'.$$
This is a contradiction. ■

Note that the encryption scheme must guarantee CPA-security, in particular if the message space of the commitment scheme is small. However, no security against active attacks is necessary, because each key is only used once.

13.3.2 Quadratic Residues

The scheme described in the subsequent section is a special case of the previous one. We present it nevertheless, as it has a homomorphism property which allows many extensions and applications that are not possible (at least not efficiently) for the aforementioned, generic scheme. The scheme relies on the so-called Quadratic Residuosity Assumption (QRA), which is stronger than the factoring assumption (i.e., if someone can factor, he can also break QRA). Once given the QRA, the scheme is quite simple, but to get there, we need a bit more number theory. The basic question is what numbers are squares modulo a potentially composite number N.

Squares and Roots Generally

Recall that we already defined square roots for groups of prime order in an earlier chapter. We now define square roots for arbitrary, potentially composite values N as
$$QR_N := \{x \in \mathbb{Z}_N^* \mid \exists y \in \mathbb{Z}_N^* : y^2 = x \bmod N\}$$
for any natural number N. The set QR_N are the *quadratic residues*, and the complement set $QNR_N := \mathbb{Z}_N^* \setminus QR_N$ the *quadratic non-residues*.

[1] In the proof, we will also see why the recipient recomputes the encryption instead of simply decrypting: It is a general way of checking that something is a possible correct ciphertext. However, one can save communication by not sending m, and instead letting the recipient first decrypt c and then re-encrypt the resulting m.

Lemma 13.1 (Basic Facts Squares and Roots) *A few simple rules hold for squares and roots modulo arbitrary N:*

(a) If y is a root of x modulo N, then so is $-y$.

(b) The set QR_N constitutes a subgroup of $\mathbb{Z}_N^*$, i.e., a multiplicative group.

(c) If $x_1 \in QR_N$ and $x_2 \notin QR_N$, then $x_1 x_2 \notin QR_N$.

□

Proof. The proof of part (a) follows directly from $(-1)^2 = 1$.

For proving part (b), let x_1 and x_2 be elements of QR_N and y_1 and y_2 respective roots. Then $y_1 \cdot y_2$ and y_1^{-1} are roots of $x_1 \cdot x_2$ and x_1^{-1}, respectively, because $(y_1 \cdot y_2)^2 = y_1^2 \cdot y_2^2 = x_1 \cdot x_2 \bmod n$ and $(y_1^{-1})^2 = (y_1^2)^{-1} = x_1^{-1} \bmod N$.

Part (c) holds for any group: If $x_1 x_2$ were in QR_N, then x_2 would also have to be in QR_N because it can be expressed as $(x_1 x_2) \cdot (x_1)^{-1}$. ■

However, some other rules that are well-known from $\mathbb{N}$ or $\mathbb{R}$ are not always true in residue rings for composite moduli. In particular, a number can have more than two roots modulo N, for example for $N = 8$ we have $1 = 1^2 = 3^2 = 5^2 = 7^2 \bmod 8$.

Squares and Roots modulo N with Known Factors

Now we consider squares and roots modulo numbers $N = pq$ as in the usual setting for cryptographic systems that rely on the hardness of factoring. As usual, we use the Chinese Remainder Theorem to exploit the facts and algorithms known for the prime factors p and q.

Lemma 13.2 (Squares and Roots mod pq) *Let $N = pq$ with p, q prime and $p \neq q$. Then the following holds:*

(a) For all $x \in \mathbb{Z}_N^*$, we have

$$x \in QR_N \Leftrightarrow x \in QR_p \wedge x \in QR_q.$$

Note that the two conditions on the right-hand side can be tested efficiently with Euler's criterion if one knows p and q.

(b) Every $x \in QR_N$ has exactly 4 square roots (except if p or q equals 2).

□

Proof. Recall that from the Chinese Remainder Theorem we know that a congruence modulo N holds exactly if it holds modulo both p and q, i.e., for all $x, y \in \mathbb{Z}_N^*$:

$$x = y \bmod N \Leftrightarrow x = y \bmod p \ \wedge \ x = y \bmod q. \quad (*)$$

(a) "⇒": Let $x \in QR_N$ be given, and let y be a root, i.e., $y^2 = x \bmod N$. Then $y^2 = x \bmod p$ and $y^2 = x \bmod q$ follow immediately from (*), and therefore $x \in QR_p$ and $x \in QR_q$.
"⇐": Now let $x \in QR_p$ and $x \in QR_q$. This means that x has a square root modulo each of p and q, say $y_p^2 = x \bmod p$ and $y_q^2 = x \bmod q$. We cannot immediately apply (*) because the two congruences are different. However, by the Chinese Remainder Theorem itself, there exists a (unique) value y modulo N with

$$y = y_p \bmod p \text{ and } y = y_q \bmod q.$$

This implies

$$y^2 = y_p^2 = x \bmod p \text{ and } y^2 = y_q^2 = x \bmod q.$$

Now we can apply (*) and obtain

$$y^2 = x \bmod N.$$

(b) We know from (a) that x belongs to QR_p and QR_q. We know that it has two roots in each group, say $\pm y_p$ and $\pm y_q$. By the proof of part (a), each of the 4 possible combinations can be combined to a root of x with the Chinese Remainder Algorithm.

■

We now extend the Legendre symbol to the so-called *Jacobi symbol* modulo $N = pq$. First this is once again pure notation:

$$\left(\frac{x}{N}\right) := \left(\frac{x}{p}\right) \cdot \left(\frac{x}{q}\right)$$

More generally, the Jacobi symbol is defined for arbitrary $N \in \mathbb{N}$ in the same way: If $N = p_1^{e_1} \cdot \ldots \cdot p_t^{e_t}$ is the prime factorization of N, then

$$\left(\frac{x}{N}\right) := \left(\frac{x}{p_1}\right)^{e_1} \cdot \ldots \cdot \left(\frac{x}{p_t}\right)^{e_t}.$$

The Jacobi symbol is always 1 or -1 (except for values $x \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$, for which the Jacobi symbol is sometimes defined to be 0; we excluded these values by definition). However, the Jacobi symbol does not denote exactly whether x is a quadratic residue modulo N! Therefore we also introduce notation for all the elements with a certain Jacobi symbol:

$$\mathbb{Z}_N^{(+1)} := \{x \in \mathbb{Z}_N^* \mid \left(\frac{x}{N}\right) = 1\}$$

and $\mathbb{Z}_N^{(-1)}$ analogously.

Lemma 13.3 (Jacobi Symbol) *Let $N = pq$ with p, q prime and $p \neq q$.*

(a) We have

$$\left(\frac{x}{N}\right) = 1 \Leftrightarrow (x \in QR_p \wedge x \in QR_q) \text{ or } (x \notin QR_p \wedge x \notin QR_q).$$

Note that the first conjunction means $x \in QR_N$. Thus all squares have the Jacobi symbol 1, but not all numbers with Jacobi symbol 1 are squares.

(b) The Jacobi symbol is multiplicative, i.e., for all $x, y \in \mathbb{Z}_N^*$:

$$\left(\frac{xy}{N}\right) = \left(\frac{x}{N}\right) \cdot \left(\frac{y}{N}\right).$$

In other words, it is a homomorphism from $\mathbb{Z}_N^*$ to $\{1, -1\}$.

(c) $\mathbb{Z}_N^{(+1)}$ is a multiplicative subgroup of $\mathbb{Z}_N^*$.

□

Proof.

(a) The product in the definition of the Jacobi symbol is 1 if either both factors are 1 or both are -1. Now the claim follows immediately from the definition of the Legendre symbol.

(b) This follows immediately from the definition of the Jacobi symbol and the multiplicativity of the Legendre symbol, which easily follows from Euler's theorem.

(c) This follows immediately from Part (b): In the language of algebra, one can simply say that $\mathbb{Z}_N^{(+1)}$ is the kernel of the homomorphism. Otherwise, it is clear that for $x, y \in \mathbb{Z}_N^{(+1)}$ also $xy \in \mathbb{Z}_N^{(+1)}$, and one can easily derive that $\left(\frac{x^{-1}}{N}\right) = \left(\frac{x}{N}\right)^{-1}$ with $\left(\frac{x^{-1}}{N}\right) \in \mathbb{Z}_N^{(+1)}$.

■

Special Case $p, q = 3 \bmod 4$

If $N = pq$ for two primes $p \neq q$, and if additionally both primes are congruent to 3 modulo 4, we obtain the following lemma.

Lemma 13.4 *For $N = pq$ with p, q prime, $p \neq q$, and $p = q = 3 \bmod 4$:*

(a) The number -1 is a quadratic non-residue modulo N with Jacobi symbol 1, i.e.,

$$-1 \notin QR_N \wedge \left(\frac{-1}{N}\right) = 1.$$

(b) The non-squares with Jacobi symbol 1 are exactly the negatives of the squares:

$$\mathbb{Z}_N^{(+1)} \setminus QR_N = -QR_N.$$

(c) Anyone who knows p and q can compute square roots of a value $x \in QR_N$ by applying the algorithm from Section 6.1.3 to obtain roots y_p and y_q modulo p and q, respectively, and then applying the Chinese Remainder Algorithm to all 4 combinations of $\pm y_p$ with $\pm y_q$.

□

Proof.

(a) Obviously -1 is not a square modulo either p or q, as $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = (-1)^{2k+1} = -1 \bmod p$ for $p = 4k + 3$, and similar for q. Thus $-1 \notin QR_N$ and $\left(\frac{-1}{N}\right) = 1$.

(b) By Lemmas 13.2 (a) and 13.3 (a), a value x lies in $\mathbb{Z}_N^{(+1)} \setminus QR_N$ exactly if $x \notin QR_p$ and $x \notin QR_q$, i.e., if $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$. This is equivalent to $\left(\frac{-x}{p}\right) = \left(\frac{-x}{q}\right) = 1$ by the multiplicativity and the proof of part (a), and thus to $-x \in QR_N$. This in turn is equivalent to $x \in -QR_N$.

(c) Clear by the proof of Lemma 13.2.

■

Squares and Roots Modulo N Without Knowledge of the Factors

The basic use of all the algorithms for testing quadratic residuosity and extracting square roots in cryptography is that one hopes that some of these tasks are hard for someone who does not know the factorization of N. We summarize some known facts in the following lemma:

Lemma 13.5 *For numbers $N = pq$ with p, q prime and $p \neq q$:*
(a) Extracting square roots is infeasible under the factoring assumption.
(b) Anybody can efficiently choose elements of QR_N randomly and uniformly.
(c) Anyone can efficiently compute Jacobi symbols modulo N. (This holds for all N.)

□

Proof.

(a) This will be proved on the exercise sheet.

(b) One can simply choose a random element of $\mathbb{Z}_N^*$ and square it. This gives a uniform distribution because every element of QR_N has exactly 4 roots.

(c) The algorithm is not very complicated and its efficiency quite similar to that of the Euclidean algorithm. However, the proof uses the so-called quadratic reciprocity law which is a fairly hard theorem in elementary number theory that we do not present it here.

■

It is not known whether deciding quadratic residuosity is as hard as factoring, thus we have to make a separate assumption for it. The assumption must in particular take into account that it is not hard to compute the Jacobi symbol for arbitrary N (Lemma 13.5). This means that one can efficiently see that certain numbers x are not squares, namely those with $\left(\frac{x}{N}\right) = -1$, whereas the remaining numbers x with $\left(\frac{x}{N}\right) = 1$ might either be squares, or non-squares modulo both p and q. Moreover, the assumption cannot state that no polynomial-time algorithm can find out whether $x \in QR_N$ with non-negligible probability, because mere random guessing will already succeed with probability 1/2. Finally, we will immediately make the assumption for the class of numbers we need. The assumption is therefore formulated as follows:

Definition 13.1 (QRA Challenger) *The QRA challenger for security parameter n is defined as follows:*

- *Firstly, it randomly chooses n-bit primes p, q with $q \neq q$ and sets $N := pq$.*
- *Secondly, it chooses a value x as follows, depending on the bit b:*
  - *If $b = 0$ let $x \leftarrow_{\mathcal{R}} \mathbb{Z}_N^{(+1)} \setminus QR_N$,*
  - *If $b = 1$ let $x \leftarrow_{\mathcal{R}} QR_N$.*
- *Finally, it outputs (N, x).*

◇

An adversary wins the game against the QRA challenger if it is able to deduce the bit b significantly better than by pure guessing. In the following, $Exp_{\mathsf{A}}^{\mathsf{QRA}}(b)$ denotes the experiment where the adversary A interacts with the QRA challenger with input bit b. Furthermore $Exp_{\mathsf{A}}^{\mathsf{QRA}}(b) = 0$ and $Exp_{\mathsf{A}}^{\mathsf{QRA}}(b) = 1$ denote the event that the adversary outputs 0 and 1 in the respective experiment. The advantage is defined as usual.

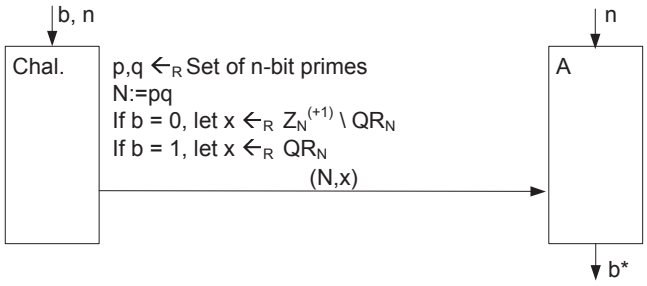

Figure 13.1: The QRA Game

Definition 13.2 (QRA Advantage) *The advantage of an adversary* A *against the QRA challenger is defined as follows (as a function of the security parameter* n *again):*

$$Adv^{\mathsf{QRA}}[\mathsf{A}](n) := \left|\mathsf{Pr}\left[Exp^{\mathsf{QRA}}_{\mathsf{A}}(0) = 1\right] - \mathsf{Pr}\left[Exp^{\mathsf{QRA}}_{\mathsf{A}}(1) = 1\right]\right|.$$

◇

Assumption 1 (Hardness of the QRA Problem) *The QRA problem is conjectured to be hard, i.e., it is conjectured that* $Adv^{\mathsf{QRA}}[\mathsf{A}](n)$ *is negligible in the security parameter* n *for all efficient adversaries* A*.*

More precisely, we will only use primes congruent 3 modulo 4 in the following, thus we might need a similar assumption 3-QRA where this additional restriction is made on the choice of p and q. However, 3-QRA follows from QRA because of Dirichlet's prime number theorem, which states that approximately half of all primes are congruent 3 mod 4. Thus about $1/4$ of all pairs (p, q) are of this form, and if an attacker A could guess quadratic residuosity with significant probability P for these, the success probability of A on arbitrary numbers would be at least $P/4$.[2]

13.3.3 The Quadratic-Residuosity Commitment Scheme

The following scheme is a bit commitment scheme. Essentially, one commits to 0 by a quadratic residue and to 1 by a quadratic non-residue. The scheme can be seen as a special case of the construction using an encryption scheme we have seen earlier, using a certain encryption scheme from Goldwasser and Micali, which encrypts 0 as a quadratic residue and 1 as a quadratic non-residue. In detail, the construction looks as follows:

- In *commit*, the committer generates n-bit primes p and q with $p = 3 \bmod 4$ and $q = 3 \bmod 4$ randomly and uniformly. (This can be guaranteed within the normal prime generation algorithm, also note that approximately half of all primes fulfill this property.) He verifies that $p \neq q$ (if not, he repeats the generation of q), and sets $N := pq$. He now chooses $y \leftarrow_{\mathcal{R}} \mathbb{Z}_N^*$ and computes
$$x := (-1)^m y^2 \bmod N,$$

[2]This was of course a sketch. One would need to state the "approximately" in Dirichlet's theorem precisely and proceed from there.

where m is his message bit. In other words, this part is y^2 or $-y^2$ depending on m. He sends the recipient

$$com := (N, x).$$

- In *open*, the committer sends (m, p, q, y) to the recipient. The committer verifies that N was generated correctly, i.e., that p and q are primes with $p = 3 \bmod 4$ and $q = 3 \bmod 4$ and $p \neq q$, and that $N = pq$. Then he verifies the main commitment, i.e., that

$$y \in \mathbb{Z}_N^* \text{ and } x = (-1)^m y^2 \bmod N.$$

Theorem 13.3 *The quadratic-residuosity commitment scheme just described is information-theoretically binding, and it is computationally hiding under the Quadratic Residuosity Assumption.* □

Proof. The binding property holds information-theoretically because no number x can be both a quadratic residue and a quadratic non-residue. We have to verify that the recipient's tests guarantee that opening x to $m = 0$ or 1 is in fact only possible if $x \in QR_N$ or $x \notin QR_N$, respectively. The former is clear. The verification of N guarantees that Lemma 13.4 applies and therefore $-y^2 \notin QR_N$.

The hiding property follows from the Quadratic Residuosity Assumption because this assumption says that one cannot distinguish quadratic residues and quadratic non-residues. More precisely, we have to check whether the commitments are chosen precisely as in the assumption.

For the modulus N this is clear, at least under the assumption 3-QRA, which was mentioned to follow from QRA.

If $m = 0$, then x is uniformly distributed in QR_N by the proof of Lemma 13.5(b).

If $m = 1$, then x is uniformly distributed in $-QR_N$, which equals $\mathbb{Z}_N^{(+1)} \setminus QR_N$ by Lemma 13.4(b). Thus if m is chosen randomly, the resulting x is a uniformly random element of $\mathbb{Z}_N^{(+1)}$. (In both cases for m, each of the possible values x occurs for 4 values y.) Thus we have shown that if m is chosen randomly, secrecy is exactly equivalent to the QRA. For our special case where the message space has only 2 elements, this immediately implies CPA-security: The attacker cannot find two messages m_1, m_2 such that he can distinguish their encryptions (here commitments) if the honest participant selects one of m_1 and m_2 randomly and encrypts it. In our case, m_1 and m_2 can only be 0 and 1. ■

If one wants to commit to several bits, it seems intuitively clear that one can commit to each bit separately with the given construction, and that secrecy of each bit will imply secrecy of the entire message. With the computational definition of secrecy, this statement nevertheless needs a proof, which we omit here.

Improved multi-bit commitment scheme For a commitment to a message m that is treated as a whole, i.e., whose bits will all be opened at the same time, the committer can choose one modulus N that is valid for all bits and only compute a separate value x_i for each bit m_i. Note that the committer cannot use the same modulus N for messages m and m' that will be opened at different times because revealing p and q would open both commitments. We can later extend this by using zero-knowledge proofs for the partial correctness of N. Then the choice of N and the proof of its correctness could be a subprotocol initialization that is used for many commitments, and the individual commitments would be opened by only showing y.

Special Properties

The quadratic-residuosity commitment scheme is obviously non-interactive and with public verification. Several other nice properties make the scheme especially useful in larger protocols. These are captured in he following lemma, whose proof will be omitted.

Lemma 13.6 *The following properties hold for the improved multi-bit commitment scheme, where several bits are committed to using the same modulus N.*

- *Once N is known, anyone (i.e., not only the committer) can make commitments with respect to this N. The committer can open them all.*
- *The commitments are homomorphic: Given two bit commitments x_1 and x_2 with respect to the same modulus N, their product $x_1 \cdot x_2$ is a commitment to $m_1 \oplus m_2$, where m_i is the content of x_i. If one assumes that correctness of N is shown in other ways, the product commitment can be opened without revealing additional information about the individual contents.*
- *Once N is known, anyone can blind commitments, i.e., transform a given commitment into a random commitment to the same bit.*

□

13.4 Information-Theoretically Hiding Schemes

This section is dedicated to an information-theoretically hiding commitment scheme based on discrete logarithms that can be used to commit to many bits efficiently. Committing is interactive. This is natural because the binding property should hold under a computational assumption, i.e., something should be infeasible for a computationally restricted committer. Thus someone else must choose the particular instance of the hard problem, e.g., a number that a cheating committer hopefully cannot factor.

13.4.1 Commitment Schemes based on Discrete Logarithms

The following scheme can be used in any family of groups of prime order where computing discrete logarithms is conjectured to be infeasible. For concreteness, we use subgroups G_q of $\mathbb{Z}_p^*$, making the same discrete-logarithm assumption as in Chapter 7.

The Discrete-Logarithm Commitment Scheme

We start immediately with the construction:

- As for the ElGamal encryption scheme we assume a function $n_p \colon \mathbb{N} \to \mathbb{N}$ which determines the size of the group $\mathbb{Z}_p$ containing the subgroup G_q. The message space is $\mathbb{Z}_q$.
- In commit, the recipient randomly chooses an n-bit prime q and an $n_p(n)$-bit prime p with $q|(p-1)$. The recipient also selects an elements g of order q, and a second generator h of $\langle g \rangle$. The recipient sends
$$pk_{Rec} := (p, q, g, h)$$
to the committer. The committer verifies that p and q are prime and that g and h are of order q. The committer chooses an integer $k \leftarrow_{\mathcal{R}} \mathbb{Z}_q$ and sends
$$com := g^m h^k \bmod p.$$

to the recipient.

- In open, the committer sends (m, k), and the recipient verifies that $com = g^m h^k \bmod p$.

Theorem 13.4 *The discrete-logarithm commitment scheme just described is information-theoretically hiding, and it is computationally binding under the Discrete Logarithm assumption for the chosen family of groups.* □

Proof. For the information-theoretical hiding property, we essentially show that k serves as a one-time pad on m.

First note that the committer verifies that g and h generate the same group of order q, where q is prime. Hence each element of G_q has exactly one representation as h^k with $k \in \{0, \ldots, q-1\}$. Thus h^k is a uniformly random element of G_q. Multiplying by h^k is therefore a one-time pad operation in the group G_q and hides g^m perfectly, and thus also m.

Now assume for contradiction that a cheating committer could break the binding property, using a probabilistic polynomial-time algorithm C. Thus C gets an input (p, q, g, h) and should output values (com, m, k, m', k') with $m \neq m'$ and $com = g^m h^k = g^{m'} h^{k'} \bmod p$. From such outputs of C, one can easily compute $\mathsf{Dlog}_g(h) \bmod p$ as follows:

$$\begin{aligned} g^m h^k = g^{m'} h^{k'} \bmod p \quad &\Rightarrow \quad g^{m-m'} = h^{k'-k} \bmod p \\ &\Rightarrow \quad g = h^{(k'-k)(m-m')^{-1}} \bmod p. \end{aligned}$$

The inverse in the exponent is taken in the group $\mathbb{Z}_q^*$: The precondition $m \neq m'$ implies $m - m' \neq 0 \bmod q$, and congruences mod q can be applied in the exponents. Thus $(k' - k)(m - m')^{-1} = \mathsf{Dlog}_g(h) \bmod p$. Hence we can use C as a subprogram to construct an almost equally efficient algorithm C^* that computes discrete logarithms with the same success probability as that of C. This contradicts the discrete-logarithm assumption. ∎

Note that this commitment scheme is similarly efficient as other asymmetric cryptographic primitives like RSA (in contrast to the quadratic-residue scheme): To commit to an n-bit message, one essentially only needs two exponentiations with exponents of length n. This means two multiplications per message bit. Moreover, the commitment is only one number modulo p.

Improved scheme for many commitments The recipient can generate pk_{Rec} once and for all in a subprotocol initialization. Then many committers can commit to many messages for this recipient non-interactively, always using this pk_{Rec}. Each of them verifies the correctness of pk_{Rec} before he uses it for the first time. One can easily see that the security proof still holds in this scenario.

Every recipient really needs his own pk_{Rec} (or needs to trust another recipient or a center for its generation) because he is only convinced that the commitments are binding if he is sure that nobody has generated g as h^x so that he would know $\mathsf{Dlog}_g(h)$.

Special Properties

The discrete-logarithm commitment scheme is obviously with non-interactive opening and public verification.[3] Similar to the quadratic-residuosity commitments, it has another nice property that

[3] Note that public verification does not contradict the fact that recipients cannot trust commitments made for other recipients, which was mentioned earlier.

is useful in larger protocols.

Lemma 13.7 *The discrete-logarithm commitment scheme is homomorphic if several messages are committed to using the same value pk_{Rec}: If a committer has made two commitments com_1 and com_2 with respect to the same value $pk_{Rec} = (p, q, g, h)$, he can open the product $com_1 \cdot com_2$ as a commitment to $m_1 + m_2 \bmod q$. Opening the product commitment does not reveal additional information about the individual contents.* □

Note that the homomorphism property was formulated in a different way than in Lemma 13.6: Now we said "he can open the commitment as" instead of talking about the content of a commitment. This is unavoidable with information-theoretically hiding commitments: The notion "the content" is not defined for such commitments by themselves because of the fact that they can have any content — this is exactly the property of being information-theoretically hiding. What the "actual" content is lies only in the knowledge of the committer, in our case which of the possible pairs (m, k) the committer knows.[4]

Proof. The committer knows values k_1 and k_2 such that $com_1 = g^{m_1} h^{k_1}$ and $com_2 = g^{m_2} h^{k_2} \bmod p$. This implies

$$com_1 \cdot com_2 = (g^{m_1} h^{k_1})(g^{m_2} h^{k_2}) = g^{m_1+m_2} h^{k_1+k_2} \bmod p. \qquad (*)$$

Thus the committer can open the product commitment by sending $(m_1 + m_2, k_1 + k_2)$. Both sums are taken modulo q.

We now show that after the product commitment has been opened, any pair of messages (m'_1, m'_2) with the correct sum is still equally likely. This is now an information-theoretic property and therefore simpler to prove than in Lemma 13.6: We show that for each such pair, there is exactly one pair (k'_1, k'_2) that the committer could have chosen in committing such that it fits both the secret messages (m'_1, m'_2) and the view of the attacking recipient. As the "pads" k_1 and k_2 are chosen uniformly, this is sufficient for secrecy.

Uniqueness: If such a pair (k'_1, k'_2) exists at all, then $com_1 = g^{m'_1} h^{k'_1}$ and $com_2 = g^{m'_2} h^{k'_2} \bmod p$ must hold. This uniquely defines $h^{k'_1}$ and $h^{k'_2} \bmod p$ and thus also k'_1 and k'_2 modulo q because h is a generator of G_q.

Existence: The recipient's view consists of the two commitments and the values (m_1+m_2, k_1+k_2) that opened the commitments. Thus we have to show that the only possible values k'_1 and k'_2 fulfill $k'_1 + k'_2 = k_1 + k_2 \bmod q$. We know that

$$com_1 com_2 = (g^{m'_1} h^{k'_1})(g^{m'_2} h^{k'_2}) = g^{m'_1+m'_2} h^{k'_1+k'_2} \bmod p.$$

We combine this with equation (*) and use that $m'_1 + m'_2 = m_1 + m_2 \bmod q$: Thus we get

$$h^{k_1+k_2} = h^{k'_1+k'_2} \bmod p.$$

This implies $k'_1 + k'_2 = k_1 + k_2 \bmod q$, which finishes the proof. ■

Remark: One can easily generalize this from sums to arbitrary linear combinations, i.e., if a_1, a_2 are two openly known numbers, the committer can open $com_1^{a_1} com_2^{a_2}$ as a commitment to $a_1 m_1 + a_2 m_2$ without revealing additional information about the individual contents.

[4] For the same reason, we cannot state an analog of Lemma 13.6 (c). Moreover, nobody can efficiently open commitments that he has not made, in contrast to Lemma 13.6 (a).