9 Quantum cryptography beyond key distribution

While QKD is arguably the most celebrated cryptographic application of quantum communication, many others are known. In this chapter and the next we look at a variety of other settings where quantum information provides an advantage. As before we focus on applications involving two parties, namely our usual suspects Alice and Bob. In contrast to earlier chapters however, in what follows Alice and Bob no longer trust each other. As such, our objective is not to protect Alice and Bob from a third entity — Eve — but instead we would like to ensure that an honest Alice (or Bob) will be protected against a dishonest Bob (or Alice).

9.1 Coin flipping

Week 8, Lecture 8.5,Lecture 1: Coin flipping
Week 8, Lecture 8.5,Lecture 2: Coin flipping over the phone

As a warmup let us start with a problem, introduced by Blum in the 1980s, that is deceptively simple: coin flipping. Imagine that Alice and Bob jointly won a single laptop as a prize in a quantum programming competition. They agree to flip a coin $c \in \{0, 1\}$ to determine which one of them can take home the laptop. Unfortunately for our protagonists, Alice is in Europe and Bob is in North America. So they need to perform this coin flip over the phone (or the internet), by exchanging classical (or quantum!) communication. Can we find a good protocol to help them solve this task?

To think through this, let us first try and define a bit more carefully what we mean by "their task," which is to produce a coin flip c (see Fig. 9.1). As usual, we would first like the protocol to be correct: If Alice and Bob are both honest, then they should eventually agree on the outcome c of the coin flip in order to avoid further arguments. Ideally, when they are both honest, the coin should also be fair, in the sense that the probability of obtaining an outcome $c = 0$ (say, heads), is the same as the probability to produce the outcome $c = 1$ (say, tails).

Finding a protocol that works if both of them are guaranteed to be honest is indeed an easy task. For example, they could simply agree that Alice flips a coin locally to obtain c, and then she tells Bob the outcome c on the phone. But if the stakes are high (remember that the coin flip outcome will determine who gets the laptop!) then Bob may not trust Alice to perform the local coin flip. In this case the trivial protocol we just described no longer provides any guarantees: Alice has full control over the final outcome, and this is clearly not satisfactory.

If Alice and Bob do not trust each other, then they would like the protocol not only to be correct, but also *secure*. What does security mean in the context of coin flipping? One possible definition of security demands that neither one of the two parties can bias the coin too much in either direction. That is, if Alice is honest, she is guaranteed that whatever Bob does, the probability of obtaining either heads or tails is close to uniform, i.e. $p(c = 0) \approx p(c = 1) \approx 1/2$. This form of coin flipping is known as *strong coin*

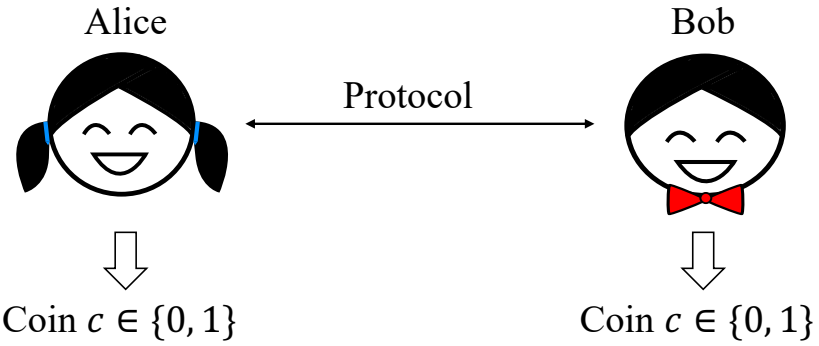

Fig. 9.1 Coin flipping: Alice and Bob engage in a protocol in order to produce a coin flip $c \in \{0,1\}$. After the protocol, Alice and Bob should both agree on the value of c, where ideally both outcomes of c should be equally likely. Moreover, neither Alice nor Bob should be able to bias the coin in a specific direction.

flipping, where the word strong refers to the fact that the dishonest party cannot bias the coin in either direction. This leads to the following (informal) definition.

Definition 9.1.1 (Strong coin flipping) Strong coin flipping *is a two-party task between Alice and Bob. The goal is for both parties to output the same value* $c \in \{0,1\}$ *such that the following properties hold.*

- *Correctness: if both Alice and Bob are honest, then* c *is uniformly distributed:* $p(c=0) = p(c=1) = 1/2$.
- ε*-secure: if Alice (or Bob) is honest, then Bob (or Alice) cannot bias the coin by more than* ϵ*:*

$$\frac{1}{2} - \varepsilon \leq p(c=0),\ p(c=1) \leq \frac{1}{2} + \varepsilon\ ,$$

where $p(c)$ *denotes the probability that the honest party outputs the value* c.

The smallest ε *for which a protocol is* ε*-secure is called the (strong coin flipping)* bias *of the protocol.*

What if both Alice and Bob are dishonest? In this case all bets are off, and there are no guarantees required for the protocol. Indeed, in any multi-party protocol in which some of the parties may be dishonest we never need to worry about writing down security guarantees protecting the dishonest party. Our objective is only to design protocols that protect the honest partie(s) from the dishonest one(s). This is because it is impossible to write down a security definition that makes guarantees for the dishonest parties: one strategy of the dishonest parties could always be to produce a random output, or any other output of their choice, just to make sure that our definition is not satisfied.

Thinking back to the laptop example that we gave earlier, we can see that we are overshooting the goal a little by asking for a strong coin flipping protocol. After all, we can reasonably assume that both Alice and Bob want to obtain the laptop, otherwise they could simply give up. So let's say that Alice and Bob agree that $c = 0$ means that Alice gets the laptop, and $c = 1$ indicates that Bob obtains the laptop. A protocol that would assure us that Alice cannot force $p(c = 0) \geq 1/2 + \varepsilon$, and Bob cannot force

$p(c = 1) \geq 1/2 + \varepsilon$ for some (hopefully small!) error parameter $\varepsilon \geq 0$ would evidently be sufficient to solve our problem. This motivates the definition of a weaker cryptographic primitive.

Definition 9.1.2 (Weak coin flipping) Weak coin flipping *is a two-party task between Alice and Bob. Neither party has an input. The goal is for both parties to output the same value $c \in \{0, 1\}$ such that the following properties hold.*

- *Correctness: If both Alice and Bob are honest, then c is uniformly distributed: $p(c = 0) = p(c = 1) = 1/2$.*
- *ε-secure: If Alice is honest, then $p(c = 1) \geq 1/2 + \varepsilon$. If Bob is honest, then $p(c = 0) \geq 1/2 + \varepsilon$.*

As before, the smallest ε for which a protocol is ε-secure is called the (weak coin flipping) bias *of the protocol.*

While this second definition is clearly less demanding than the first, it is not immediately clear that either definition can be satisfied at all. Is secure coin flipping possible?

Quiz 9.1.1 *A strong coin flipping protocol immediately implies a weak coin flipping protocol with the same bias. However, we can also derive a strong coin flipping protocol from a weak one. A simple way to do this is to make the modification that whoever wins the weak coin flip gets to flip its own 50-50 coin (if acting honestly) and announce the final outcome of the protocol. What is the bias of this strong coin flipping protocol, if the weak coin flipping protocol had bias ϵ?*

a) $\frac{3}{8} + \frac{\epsilon}{4}$
b) $\longrightarrow \frac{1}{4} + \frac{\epsilon}{2}$
c) $2(\epsilon + \frac{1}{2}) - \frac{1}{2}$
d) ϵ

Solution: For the proposed strong coin flipping protocol, if the final outcome is given by c', then, the cheating probability is $P = p(c' = 0) = p(c' = 1) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \left(\frac{1}{2} + \epsilon\right)$; and the bias is $P - \frac{1}{2}$.

9.1.1 Classical coin flipping

Let us first have a look at whether we might be able to construct a classical protocol for coin flipping. Classical means that the resources that Alice and Bob have at their disposal is a classical communication channel connecting them. No other assumptions are made: in particular Alice and Bob have unlimited computational resources, and we have no guarantees, neither about the locations of Alice and Bob, nor the classical communication channel other than that it will eventually deliver a message. Maybe a protocol a bit more sophisticated than the trivial protocol discussed earlier would work? Instead of just one party flipping a coin, a natural idea is to have both of them do it simultaneously, and then announce their choice via the classical communication channel.

Protocol 1 (Blum coin-flipping)

1 Alice flips a random bit $a \in \{0, 1\}$ and sends it to Bob.
2 Bob flips a random bit $b \in \{0, 1\}$ and sends it to Alice.
3 Both parties output the coin flip $c = a \oplus b$.

As usual, our first goal is to check if the protocol is correct: indeed, it is easy to see that if both parties are honest, then c is uniformly distributed. In fact, it seems that it might be sufficient that only one party is honest: as long as a or b is random then $a \oplus b$ will be random. Is this right? Note that the protocol forces

us to specify an order in which the parties exchange their messages (indeed, it is never wise to attempt to speak simultaneously over the phone). Here we made Alice go first, and Bob second. So Bob receives Alice's message a before he sends her his choice of b. But this makes it possible for him to cheat! Bob can easily force any outcome $c = b'$ of his choice by choosing b' first and then selecting $b = b' \oplus a$ at step 2. Thus this protocol does not even fulfill the security requirement of a weak coin flipping protocol.

At first glance you might hope that it will be possible to find a more sophisticated classical protocol. Unfortunately, it turns out that there exists *no* classical protocol for coin flipping that is secure without making any additional assumptions: no value of $\varepsilon < 1/2$ can be achieved for security. In other words, whenever one party cannot completely bias the outcome of the protocol to a certain value, then the other party can: there is always at least one of Alice or Bob who can cheat perfectly (in the protocol above, it is Bob). Very intuitively, the reason is precisely the same as what made the Blum protocol insecure: one can argue that, whatever the outcome c of the protocol will be, it has to be determined at *some* point in the protocol. By considering the messages during the course of the protocol (which can involve many rounds of interaction) one can determine the message before which the outcome c was not yet determined, but once the message is sent, the outcome becomes determined. In the case of the Blum protocol above, this message is Bob's message to Alice. However, given that c is not yet determined before that message, the next message sent effectively will determine a value for c. The party who sends that message thus has the ability to bias the coin as they desire.

If we allow ourselves to make a few more assumptions then a slight variation of the Blum protocol *can* work. For example, if the channel connecting Alice and Bob features a guaranteed message delivery time t that cannot be influenced by either Alice or Bob (e.g. via special relativity if Alice and Bob's locations are fixed), and they have synchronized time slots (possibly by using the message delivery times), then we could modify the Blum protocol by asking that Alice and Bob both send a and b simultaneously. Any message that arrives after time t is immediately rejected by the recipient. This way, we can be sure that none of the two parties can base their choice of which bit to send on their knowledge of the other's message, and the protocol becomes secure.

9.1.2 Quantum strong coin flipping

Luckily, our impossibility argument does not apply to quantum protocols. Can you see why? Try to run the argument in your head — what part breaks down? This is a bit subtle: the main reason is that for a quantum protocol the notions of a transcript, and the outcome being "determined", are not well-defined, because everything can happen in superposition. So we can't really talk of a specific message in the protocol when the outcome becomes determined: the determining message could in fact be a superposition of messages exchanged at different times.

As is often the case, there is a good reason for a classical argument not to extend to the quantum setting: strong coin flipping actually *is* possible using quantum information, for some non-trivial values of the bias $\varepsilon < 1/2$. Let us see an example such protocol. To describe the protocol we introduce the term "qutrit", which is used to refer to a quantum state in the space $\mathbb{C}^3$, i.e. a state of the form $\alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle$.[1]

Protocol 2 (Quantum weak coin-flipping) *For $a, x \in \{0,1\}$ define the qutrit*

$$|\phi_{a,x}\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + (-1)^x |a+1\rangle \big) .$$

1 Alice selects $x \in \{0,1\}$ and $a \in \{0,1\}$ uniformly at random and sends $|\phi_{a,x}\rangle$ to Bob.

[1] One way to realize qutrits is by using two qubits, e.g. by identifying $|0\rangle \leftarrow |00\rangle$, $|1\rangle \leftarrow |01\rangle$, and $|2\rangle \leftarrow |10\rangle$ (and we make sure that $|11\rangle$ always has zero amplitude).

2 *Bob selects $b \in \{0,1\}$ uniformly at random and sends b to Alice.*
3 *Alice sends a and x to Bob.*
4 *Bob verifies that the state he received from Alice in step 1 is $|\phi_{a,x}\rangle$ (e.g. by measuring in any orthonormal basis containing $|\phi_{a,x}\rangle$). If it is not the case then he declares that Alice has been cheating and aborts the protocol.*
5 *Both return the outcome $c = a \oplus b$.*

Note the similarity between this protocol and the Blum protocol from the previous section. Here as well Alice and Bob each choose "half" of the outcome c: Alice chooses a, Bob b, and they return $c = a \oplus b$. However, Alice does not fully reveal a to Bob in her first message: instead, she provides him with some form of "weak commitment" to a in the form of the state $|\phi_{a,x}\rangle$. Because the four states $|\phi_{a,x}\rangle$ are not orthogonal, it is impossible for Bob to completely discover the value of a without first learning x, which only happens after he had to make his choice of b.

Exercise 9.1.1 Compute the reduced density matrices $\rho^B_{|a=0}$ and $\rho^B_{|a=1}$ associated with Bob's view of the protocol after Alice's first message has been sent, for a uniformly random choice of $x \in \{0,1\}$ and $a = 0$ or $a = 1$ respectively. Show that the trace distance between these two matrices is $1/2$. Conclude that the probability with which Bob can force an outcome c of his choice is at most $3/4$.

The exercise shows that the maximum bias that a cheating Bob can force in the protocol is $\varepsilon = 1/4$. Security for cheating Alice is a bit harder to argue, because we have to consider the possibility for her to prepare an arbitrary state in the first step, which may be entangled with some information she keeps on the side. She could then subsequently use all of these, together with the value b received from Bob, to determine her message in the third step of the protocol. We will not give the details here; the main result one can show is the following.

Theorem 9.1.1 *The quantum coin-flipping protocol, Protocol 2, is correct and ε-secure for $\varepsilon = 1/4$.*

Can we do even better? Unfortunately it turns out that perfectly secure strong coin-flipping is also impossible for quantum protocols: Kitaev showed that the smallest bias any protocol could achieve is $\varepsilon = (\sqrt{2}-1)/2 \approx 0.207$. Kitaev's proof is an extension of the classical impossibility argument, based on an ingenuous representation of transcripts for quantum protocols. If you are interested in Section 9.5 at the end of the chapter we sketch an argument that is in some sense "dual" to Kitaev's.

9.1.3 Quantum weak coin flipping

If we relax our requirements to *weak* coin-flipping now, it turns out that using quantum protocols this is possible with *arbitrarily small* (but non-zero) bias ε. The best protocol known for achieving this, however, is very complex! In particular the protocol requires a large number of rounds of interaction, that scales exponentially with $1/\varepsilon$. Using this protocol, it is possible to show that there exists quantum strong coin flipping protocols with bias $(\sqrt{2}-1)/2 + \varepsilon$ for any $\varepsilon > 0$, nearly matching Kitaev's lower bound.

9.2 Two-Party cryptography

Coin flipping is an example of a two-party task. In this task, the parties have no input at all, and they each produce a single bit of output (the coin flip). More generally, we can imagine two parties (Alice and

Bob?) who each have some classical input, x for Alice and y for Bob, and each would like to compute a certain function of both their inputs, $f_A(x, y)$ for Alice and $f_B(x, y)$ for Bob.[2] To do this they can communicate over a classical, or even quantum, channel. An easy solution would be for Alice and Bob to exchange their respective inputs and perform the computation locally. Unfortunately they don't trust each other: neither party wants to reveal more information about his or her input than is absolutely necessary. How do they do it?

A concrete example of such a task is "Yao's millionaire's problem". Here x and y represent Alice and Bob's respective fortune. The functions $f_A(x, y) = 1_{x>y}$ and $f_B(x, y) = 1_{y>x}$, where $1_{x>y}$ is 1 if $x > y$ and 0 otherwise, tell Alice and Bob if their fortune is larger than the other's. Can they decide who is the richest without announcing their actual fortune? (It turns out they can, but it's not so easy: we'll see how to do it later.)

9.2.1 Secure Function Evaluation

Week 8, Lecture 8.1,Lecture 1: Secure function evaluation

The general task we just introduced is called Secure Function Evaluation (SFE). Let's formalize it (see also Figure 9.2).

Definition 9.2.1 *Secure function evaluation (SFE) is a task involving two parties, Alice and Bob. Alice holds an input $x \in \mathcal{X}$ and Bob holds an input $y \in \mathcal{Y}$. Alice and Bob interact over a communication channel, and output an $a \in \mathcal{A}$ and $b \in \mathcal{B}$ respectively. We say that a given protocol is a secure protocol computing a pair of functions $(f_A : \mathcal{X} \times \mathcal{Y} \to \mathcal{A}, f_B : \mathcal{X} \times \mathcal{Y} \to \mathcal{B})$ if it satisfies the following properties:*

- Correctness: *If both Alice and Bob follow the protocol (we say that they are* honest*) then $a = f_A(x, y)$ and $b = f_B(x, y)$.*
- Security against cheating Bob: *If Alice is honest, then Bob cannot learn more about her input x than he can infer from $f_B(x, y)$.*
- Security against cheating Alice: *If Bob is honest, then Alice cannot learn more about his input y than she can infer from $f_A(x, y)$.*

Note that the definition does not guarantee anything in case Alice and Bob are both dishonest. In this case there is nothing we can do! The goal is only to protect the honest parties. The definition is intuitive, and it is rather informal — for example, what does it mean that "Bob cannot learn more about Alice's input x than he can infer from $f_B(x, y)$"? It turns out that this requirement is very delicate to make precise! We will return to it in a moment. First let's consider some examples of Secure Function Evaluation tasks.

Example 9.2.1 *Alice and Bob are contemplating going to a movie. Here, $x, y \in \{0, 1\}$ where '0' denotes "no" and '1' denotes "yes". The function they wish to compute is*

$$f(x, y) = f_A(x, y) = f_B(x, y) = x \text{ AND } y.$$

Let us see what security means here. If $f(x, y) = 1$, then it must be that $x = y = 1$ and both parties learn the other's input. Alice and Bob go to the movies. If $f(x, y) = 0$, then it must be that either $x = 0$ or $y = 0$ (or both). If Alice input is $x = 1$, i.e., she would like to go to a movie, and the output is

[2] If we are precise, coin flipping requires a randomized output, so it does not strictly fall in the standard framework for two-party cryptography which requires f_A and f_B to be deterministic functions. For this reason it is best to treat it separately.

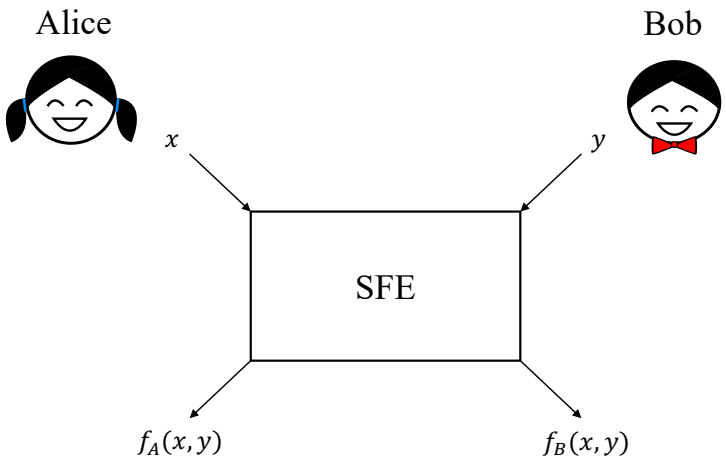

Fig. 9.2 Secure Function Evaluation (SFE): Alice has an input x, Bob has an input y. After the protocol is completed Alice should learn $f_A(x,y)$ for some function f_A, and Bob should learn $f_B(x,y)$. Neither of them should gain any further information.

$f(x,y) = 0$, then Alice can infer $y = 0$, but Bob will never learn whether $x = 0$ or $x = 1$. So a party only learns the others' input if they themselves declared that they wanted to go to a movie. ■

Example 9.2.2 *Alice (a customer) wants to identify herself to Bob (an ATM). Here, x is the password honest Alice should know, and y the password (for Alice) that the honest ATM should have stored in its database. The function f is the equality, that is, $f(x,y) = 1$ if and only if $x = y$, and $f(x,y) = 0$ otherwise. Security means that if Alice is dishonest (she might not know x but is still trying to break through the ATM's authentication system), then Bob should have the guarantee that Alice will never learn anything more about his input y than she can infer from $f(x,y)$ — that is, whatever x she tries, that $x \neq y$! (Unless she happens to be lucky of course.) Similarly, if Bob is a fraudulent ATM who is out to steal passwords from the users, the best he can do is guess a y and see whether it worked. No more information is revealed.* ■

Example 9.2.3 *Alice wants to sell a book to Bob. Here, x is Alice's asking price, and y is Bob's bid. The function they wish to compute is $f(x,y) = (ok, y)$ if $y \geq x$, and $f(x,y) = (no, 0)$ if $y < x$. If $f(x,y) = (ok, y)$, Alice can proceed to sell the book to Bob. Bob pays what he offers, and Alice gets at least her asking price. Security means that dishonest Bob can never learn what the asking price actually was, only that it was less or equal than his bid. If $f(x,y) = (no, 0)$, then Alice will not sell her book. Security means that Alice will never learn exactly what Bob's bid actually was, only that it was lower than her asking price. Similarly, Bob will only learn that Alice's asking price was higher than his bid.* ■

Quiz 9.2.1 *True or False: In a SFE protocol Bob learns nothing at all about Alice's input.* ***Solution: False***

Quiz 9.2.2 *True or False: Consider a secure function evaluation protocol that outputs $f_A(x,y) = f_B(x,y) = x \oplus y$. Then a malicious Bob can learn with certainty Alice's input.* ***Solution: True***

9.2.2 The simulation paradigm for security

The key ideas to formalize the intuition behind our informal definition of security for SFE are the concepts of an *ideal functionality* and a *simulator*.

Definition 9.2.2 (Ideal functionality) *Let $(f_A : \mathcal{X} \times \mathcal{Y} \to \mathcal{A},\ f_B : \mathcal{X} \times \mathcal{Y} \to \mathcal{B})$ be a pair of functions corresponding to an SFE task. The* ideal functionality *is a trusted device which takes as input $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and returns $f_A(x, y)$ and $f_B(x, y)$.*

Thus the ideal functionality does precisely what a protocol solving the SFE task is supposed to achieve: it directly and honestly returns the values of each of the two functions. It is "ideal" in the sense that it does not require any interaction between the two parties: you should picture a "black box," similar to the box marked SFE in Figure 9.2, which takes the inputs and provides the outputs, no questions asked. Informally, we will then say that a protocol for SFE is secure if, provided one of the parties is honest, whatever the other party does there is nothing more they can obtain that they could not have obtained by interacting with the ideal functionality.

Example 9.2.4 *Consider again the millionaire's problem. Here the ideal functionality takes as input x from Alice and y from Bob, and returns $f_A(x, y) = 1_{x>y}$ to Alice and $f_B(x, y) = 1_{y>x}$ to Bob. Suppose we are given a protocol for this problem, and suppose a malicious Bob was able to infer Alice's fortune x through his interaction with her. Then the simulation paradigm dictates that, if the protocol is secure, he should be able to do the same through an interaction with the ideal functionality. But the ideal functionality just takes any y' of Bob's choice and returns to him $f_B(x, y') = 1_{y'>x}$. Since only one interaction is allowed, the best Bob can do is find out if $x < y'$ for a single y' of his choice, which for this SFE task is unavoidable.* ∎

Now suppose that we have a candidate protocol for the millionaire problem. This protocol states what Alice and Bob's actions should be in the protocol (if they are honest). How do we prove that the protocol is secure in the simulation paradigm? According to the discussion above we need to show that, whatever Bob (or Alice) can do in the real protocol, he "should be able to do the same through an interaction with the ideal functionality." So how do we show that this is the case? The idea to do this is to define a *simulator*. A simulator is simply an algorithm that interacts with the ideal functionality on the one hand and with Bob on the other. We use the simulator to show that, for any dishonest Bob that obtains some information in the real protocol, there is a "simulated Bob," obtained by inserting the simulator between the real Bob and the ideal functionality, that obtains the same information as the dishonest Bob. If the simulated Bob could obtain the information from the ideal functionality, without even involving Alice, then by definition this contains no more information about Alice's input than is already revealed in Bob's output. More formally, we make the following definition.

Definition 9.2.3 (Security against cheating Bob) *A protocol for an SFE task (f_A, f_B) is secure if for any malicious Bob interacting with an honest Alice in the protocol, there exists a* simulator *which, by controlling Bob in an interaction with the ideal functionality, is able to generate a distribution on outputs that is indistinguishable from the distribution produced by malicious Bob in the real interaction.*

Of course, a symmetric definition can be given for security against cheating Alice. The definition refers to the output distributions being "indistinguishable" from one another. The strongest notion of indistinguishability is called "statistical indistinguishability". In this case it means that the two distributions, the one produced by Bob and the one produced by the simulator, cannot be differentiated by any adversary

except with very small probability. As we know, this is equivalent to the distributions having small total variation distance.

A weaker notion is "computational indistinguishability". In this case it is only required that the distributions cannot be distinguished by any *efficient* algorithm. Here "efficient" means that the distinguisher has to run in time that is polynomial in some parameter of the problem, that is usually referred to as the "security parameter". Here we focus on finding certain tasks for which statistical indistinguishability can be achieved by quantum protocols.

If the notion of simulator remains somewhat hazy to you it's ok, we only want you to be familiar with the terminology and get the flavor of it. We won't worry too much about giving formal proofs of security. Nevertheless, you should be aware that doing so can be quite tricky. In fact, many *wrong* proofs of security of quantum protocols have been given in the past by relying on "intuitive" arguments rather than the simulation paradigm. We will point out one such example later.

9.3 Oblivious Transfer

Week 8, Lecture 8.2,Lecture 1: Oblivious transfer
Week 8, Lecture 8.2,Lecture 2: A quantum protocol for OT?

Let us discuss an important example of an SFE task, known as *Oblivious Transfer* (OT). Here $\mathcal{X} = \{0,1\}^\ell \times \{0,1\}^\ell$ and $\mathcal{Y} = \{0,1\}$. That is, Alice receives as input two ℓ-bit strings $x = (s_0, s_1)$, and Bob receives as input a single bit y. The goal is for Bob to obtain $f_B(x,y) = s_y$, while Alice will obtain nothing, $f_A(x,y) = \perp$. Thus a protocol will implement this task securely if, first of all it is correct, and second, malicious Alice will not obtain any information at all about Bob's input bit (since the ideal functionality never returns anything to Alice), and malicious Bob will at best learn one of Alice's strings, but never more. Figure 9.3 gives the ideal functionality for this task.

As an example scenario where this task could be useful, imagine that Alice is a database which contains two entries s_0 and s_1. Bob would like to retrieve one of them, but does not want Alice to know which one he retrieved, preserving his privacy. Alice can also be sure that he does not retrieve her entire database.

What makes OT interesting is the fact that it is *universal* for two-party cryptography. That is, if we can build a secure protocol for 1-2 OT then we can solve *any* SFE problem by just using 1-2 OT multiple times. In this respect 1-2 OT is analogous to a universal gate in computing. This universality property can be shown in different ways; see the chapter notes for pointers to how this can be done.

Unfortunately it is also known that OT cannot be implemented securely without computational assumptions... in the classical world. What about quantum protocols? Given all that we know about encoding classical information in quantum states, the following protocol is a natural first attempt.

Protocol 3 (Quantum OT protocol) *Alice has input $x = (s_0, s_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$ and Bob has input $y \in \{0,1\}$. Their goal is to compute $(f_A(x,y) = \perp, f_B(x,y) = s_y)$.*

1 *Alice selects uniformly random $x \in \{0,1\}^{2\ell}$ and $\theta \in \{0,1\}^{2\ell}$. She prepares BB'84 states $|x_j\rangle_{\theta_j}$ for $j = 1, \ldots, 2\ell$ and sends them to Bob.*
2 *Bob measures each of the qubits he received from Alice in a random basis $\tilde{\theta}_j$, obtaining outcomes $\tilde{x}_1, \ldots, \tilde{x}_\ell \in \{0,1\}$. He notifies Alice that he is done with his measurements.*
3 *Alice reveals her choice of bases $\theta_1, \ldots, \theta_{2\ell}$ to Bob.*

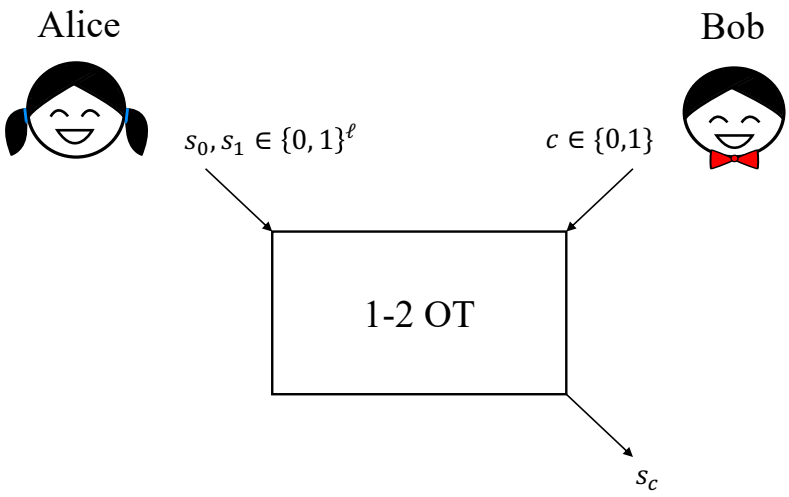

Fig. 9.3 1-2 Oblivious Transfer. Alice has two inputs $s_0, s_1 \in \{0,1\}^\ell$. Bob has a choice bit $c \in \{0,1\}$, according to which he receives the desired string s_c as output. Alice has no output, although it is often implicitly assumed that Alice obtains a notification that the 1-2 Oblivious Transfer is completed (i.e. Bob made a choice for c, and received the corresponding output).

4 *Bob sets* $I = \{i : \theta_i = \tilde{\theta}_i\}$, $I_y = I$ *and* $I_{1-y} = \{1, \ldots, 2\ell\} \backslash I$. *(For simplicity, assume that* $|I_0| = |I_1| = \ell$*.) Bob sends* (I_0, I_1) *to Alice.*

5 *Alice sends* $t_0 = s_0 \oplus x_{I_0}$ *and* $t_1 = s_1 \oplus x_{I_1}$ *to Bob.*

6 *Alice outputs* $\perp$*, and Bob outputs* $t_y \oplus \tilde{x}_{I_y}$.

Let's first check that this protocol is correct. This is clear: whenever $j \in I_y$, by definition $\theta_j = \tilde{\theta}_j$, therefore $x_j = \tilde{x}_j$ and $(s_y \oplus x_{I_y}) \oplus \tilde{x}_{I_y} = s_y$.

Is it secure? Security against cheating Alice is not hard to verify. Indeed, the only information she gets from an honest Bob are two sets (I_0, I_1). If Bob is honest, even if Alice sent him arbitratry states in the first step, and misleading basis information in the third, since Bob's choice of $\tilde{\theta}_j$ is uniformly random the sets I_0, I_1 will be a uniformly random partition of $\{1, \ldots, 2\ell\}$ that contains no information at all about his input y. So anything a dishonest Alice could do in this protocol can be simulated by an interaction with the ideal functionality, where the simulator would replace Bob's message (I_0, I_1) (which is not provided by the ideal functionality) with a uniformly random choice.

How about security against cheating Bob? The idea is supposed to be that, given Bob's basis choices are random, he can at best learn roughly half of Alice's inputs $\tilde{x}_j$. Of course he could lie about which half he learned, but in any case he will only be able to recover about half of the bits of Alice's input $x = (s_0, s_1)$. Note he could still, for example, learn half of s_0 and half of s_1 (instead of the whole s_0 or s_1 and nothing about the other). This could be prevented by adding in a layer of privacy amplification (see Chapter **??**) to the protocol, so let's assume it is not a serious issue.

You might already have noticed that there is a more worrisome hitch. The protocol requires Bob to "measure each qubit he received from Alice", and then to "notify Alice that he is done with his measurements". But what if Bob is malicious — what if he stores Alice's qubits in a large quantum

memory, without performing any immediate measurement, and lies to her by declaring that he is done? Alice would then naively reveal her basis information, and Bob could measure all the qubits he stored using $\tilde{\theta}_j = \theta_j$. He would thus obtain outcomes $\tilde{x}_j = x_j$ for all j, and he could recover both $s_0 = t_0 \oplus \tilde{x}_{I_0}$ and $s_1 = t_1 \oplus \tilde{x}_{I_1}$.

So the protocol we gave is not at all secure! There are two ways to get around the problem. One possibility is to make certain physical assumptions on the capacities of cheating Bob. For example, that Bob has a bounded quantum memory, in which case he wouldn't be able to store all of Alice's qubits. We will explore this assumption in the next chapter. Another possibility would be to somehow force Bob to *commit* to a choice of basis $\tilde{\theta}_j$, and outcomes $\tilde{x}_j$ that he obtained, *before* Alice would accept to reveal her θ_j. Of course, to avoid reversing the difficulty it should be that Alice cannot learn any information about the $\tilde{\theta}_j$ just from Bob's commitments. The task we're trying to solve is called *bit commitment*, and it is another fundamental primitive of two-party cryptography. Let's explore it next.

Quiz 9.3.1 *A cryptographic primitive related to the 1-2 OT protocol is Rabin's OT. In Rabin's OT, Alice's input is a message. She sends it to Bob who receives it with probability* $1/2$*, while Alice remains oblivious as to whether the message was received or not.*

True or False: Rabin's OT can be constructed from 1-2 OT. ***Solution: True. Alice's and Bob's respective inputs can be*** $x = (s_0, s_1); y = \{0, 1\}$ ***where*** $s_0 = message; s_1 =$***a random string of the same length as the message; and value of bit*** y ***is chosen uniformly at random.***

9.4 Bit commitment

Week 8, Lecture 8.3,Lecture 1: Bit commitment

Let's move on to our second fundamental example of a two-party task, *bit commitment*. The idea of a bit commitment protocol (Figure 9.4 is that it should allow Alice to *commit* to an unbreakable promise, without revealing any information about the promise to Bob until Alice decides to *open* her promise — without being allowed to change her mind in-between the "commit" and "open" phases. You can think of a commitment as a *cryptographic safe*: in the commit phase, Alice places her bit inside the safe and locks the door; in the open phase she reveals the key to the safe.

Definition 9.4.1 (Bit Commitment (BC)) *Bit commitment is a task involving two parties, Alice (the committer) and Bob (the receiver). The input to Alice is a single bit* $b \in \{0, 1\}$*, and she has no output. Bob has no input, and his output is a bit* b'*. A protocol for bit commitment has two phases, the* commit phase *and the* open phase*, and it should satisfy the following properties:*

1. *(Correctness) If both Alice and Bob are honest then at the end of the protocol Bob outputs a bit* $b' = b$*.*
2. *(Hiding) For any malicious Bob, the state of Bob at the end of the commit phase (including all his prior information and information received from Alice during the commit phase, classical or quantum) is independent of* b*.*
3. *(Binding) For any three possible malicious behavior* A*,* A_0 *and* A_1*, the probabilities* p_b *that Bob outputs* $b' = b$ *after interacting with* A *in the commit phase and* A_b *in the open phase satisfy* $p_0 + p_1 \leq 1$*.*

The hiding property is clear: it states that, after the commit phase, Bob still has no information at all about the bit b that honest Alice committed to. From his point of view, it could really go either way. The

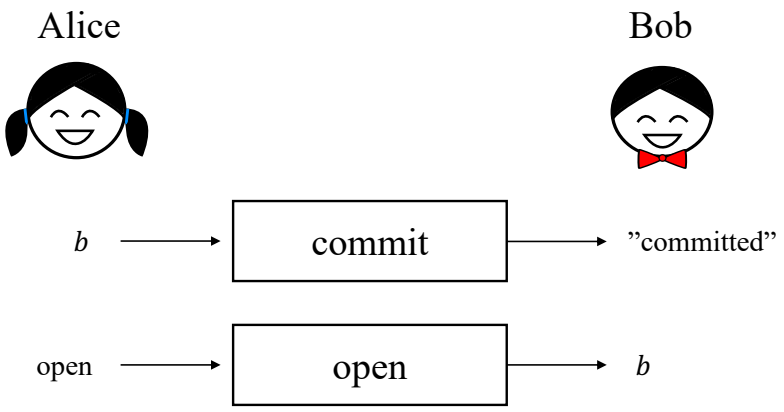

Fig. 9.4 Bit Commitment is a two-party primitive with two phases: commit and open. During the commit phase, Bob should learn that Alice is indeed committed, but gain no further information about b. When Alice decides to initiative the open phase, Bob learns b, where Alice cannot change her mind about the bit b defined by the open phase. Protocols typically also allow abort.

binding property is more subtle. Intuitively, what it is trying to capture is that once Alice has committed to a specific value b (this is the role of A in the definition), then she shouldn't be able to come up with two possible different behavior (A_0 and A_1) such that she has a strictly higher than $1/2$ chance of being able to convince Bob that $b = 0$ (she would run A_0) *or* that $b = 1$ (she would run A_1).

Bit commitment is a good example of a cryptographic task for which it is crucial to define security as precisely as possible, especially in the quantum setting. Consider the following "intuitive" definition of the binding property: "It should be impossible for malicious Alice to convince honest Bob that $b = 0$ *and* $b = 1$ with probability strictly larger than 1". Do you see the difference? I wouldn't blame you if you didn't — the pioneers of quantum information and cryptography didn't either! In 1991 Brassard et al. famously proposed an "unconditionally secure" quantum protocol for bit commitment, that satisfied the above intuitive notion of security. However, their protocol was later completely broken! (Indeed, as we will soon see, perfectly secure bit commitment is impossible both in the classical and the quantum world.) Their "mistake" is that they interpreted the italicized "and" in the intuitive definition above in a strong sense: they show that, in their protocol, it wouldn't be possible for a malicious Alice to *simultaneously* convince Bob that $b = 0$ and $b = 1$, by assuming that, if this where the case, the two final quantum states of the protocol associated with the outcomes "Bob returns $b' = 0$" and "Bob returns $b' = 1$" would exist simultaneously. However, as we know very well by now, quantum information is subtle, and the fact that Alice can "change her mind" after the commit phase does *not* imply that she can generate *both* the $b' = 0$ and $b' = 1$ states for Bob from the same state at the end of the open phase; only that she can generate either of them.

Quiz 9.4.1 *Consider the following protocol for bit commitment: Alice prepares* $|\psi_{00}\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ *if she commits to* $x = 0$*, or she prepares* $|\psi_{01}\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$ *if she commits to* $x = 1$*. Then she sends the register* B *to Bob. Finally, in the open phase, Alice sends Bob her register* A*, so that Bob can*

perform a measurement in the Bell basis on the two qubits in registers A and B to learn Alice's bit. Is this protocol correct and secure?

a) Yes
b) $\longrightarrow$ ***No***

9.4.1 Universality of bit commitment

Bit commitment is an important task in quantum multiparty cryptography, because – just as 1-2 OT – it is known to be universal. This is demonstrated by a small modification to the protocol for OT that we gave in the previous section: as we discussed, the protocol by itself is not secure; however if one has access to a secure protocol for bit commitment then it can be turned into a secure OT protocol as well. The essential idea is to enhance step 2 of the protocol by asking Bob to commit to the measurement outcomes. This allows Alice to gain confidence that Bob has measured the qubits after step 2 of the protocol, avoiding the attack described above.

Since OT itself is universal for multiparty computation we deduce that bit commitment is universal. However, note that the protocol for OT based on bit commitment we gave is quantum, even if bit commitment is implemented using a classical protocol. Interestingly, this this is unavoidable: indeed, bit commitment is *not* universal for *classical* multiparty computation! Nevertheless, it suffices as a building block for many useful protocols — such as the millionaire problem, as the next exercise asks you to show.

Exercise 9.4.1 Give a secure protocol for Yao's millionaire's problem, assuming you have access to a protocol securely implementing bit commitment.

Let's see how the reverse can be accomplished, using OT as a building block to achieve bit commitment. For this, we will consider an approximate version of bit commitment, in which Alice can change her mind with some small error probability ε. That is, we say that the protocol is ε-binding if the requirement $p_0 + p_1 \leq 1$ is relaxed to $p_0 + p_1 \leq 1 + \varepsilon$.

The following protocol takes 1-2 OT and turns it into bit commitment. In the protocol we invert the use of 1-2 OT: Bob is now the sender, and Alice the receiver.

Protocol 4 (Bit commitment from 1-2 OT) *Alice's input is $b \in \{0,1\}$. Bob has no input.*

1 *Commit phase: Bob chooses two strings $s_0, s_1 \in \{0,1\}^\ell$ uniformly at random. Bob and Alice execute a protocol for OT, with the role of the parties reversed: OT-Alice's input is (s_0, s_1) (provided by Bob), and OT-Bob's input is b (provided by Alice). Thus Alice receives s_b, and Bob receives $\perp$.*
2 *Open phase: Alice sends $\hat{b}$ and $\hat{s} = s_b$ to Bob. If $\hat{s} = s_{\hat{b}}$, then Bob accepts and concludes that Alice committed herself to $b = \hat{b}$. If $\hat{s} \neq s_{\hat{b}}$, then Bob rejects.*

Why does this give bit commitment? First of all, if both parties behave honestly the protocol is clearly correct. Now let's consider the hiding property. We need to show that, at the end of the commit phase, Bob has no information about b. This follows right away from the definition of OT, which guarantees that the sender never receives any information about the receiver's input.

It remains to show that the protocol is ε-binding. This again follows from the security of OT, for $\varepsilon = 2^{-\ell}$. Indeed, the ideal functionality for OT is such that the receiver can learn only *one* of the two strings. Suppose Alice has two possible strategies, one to open $\hat{b} = 0$ and the other to open $\hat{b} = 1$. Let p_0 be the probability that the first strategy succeeds, and p_1 the probability that the second succeeds. As a consequence, Alice can recover both of s_0 and s_1 with probability at least $p_0 + p_1 - 1$. By the security of the OT primitive, this can happen with probability at most the probability that a random guess

of the non-received string would succeed, i.e. $2^{-\ell}$. By taking ℓ large enough we can achieve any desired ε-security for the binding property.

If you have been reading carefully you may have noticed that in the argument we made a jump from "Alice can recover s_0 with probability p_0, and s_1 with probability p_1" to "Alice can recover both of s_0 and s_1 with probability at least $p_0 + p_1 - 1$". This is correct if Alice is classical, but if her strategies involve incompatible quantum measurements then the implication might no longer be true. Hence in case we allow the protocol implementing OT to be a quantum protocol, we have to be extra careful to show that the resulting protocol for bit commitment satisfies the required definition. This is possible (so the protocol described above *is* secure provided the implementation of OT is, whether classical or quantum) but one must take even greater care in making the right security definitions to ensure that they satisfy the stringent criteria of "universal composability".

9.4.2 Impossibility of bit commitment

Week 8, Lecture 8.4,Lecture 1: Impossibility of bit commitment

Since, as we argued, bit commitment implies OT (in the quantum world), but perfect OT is impossible, it must be that bit commitment is impossible as well! Let's see why. We give an informal argument, and refer you to the chapter notes for pointers.

Consider a protocol for bit commitment that is perfectly hiding, i.e. at the end of the commit phase Bob has absolutely no information about Alice's bit b. Let's show that in this case a malicious Alice can cheat *arbitrarily*: she is able to open any bit that she likes.

Suppose that the initial state of Alice and Bob is a pure state $|\psi\rangle_{AB}$, which in particular contains Alice's input. We can always assume that this is the case by considering a purification and giving the purifying system to Alice.

Now suppose Alice executes the bit commitment protocol with input b. At the end of the commit phase, the joint state of Alice and Bob can be described by some pure state $|\psi(b)\rangle_{AB}$. Since the bit commitment protocol is perfectly hiding, it must be the case that

$$\rho_B(0) = \mathrm{tr}_A(|\Psi(0)\rangle\langle\Psi(0)|_{AB}) \quad \text{and} \quad \rho_B(1) = \mathrm{tr}_A(|\Psi(1)\rangle\langle\Psi(1)|_{AB})$$

are absolutely identical, as otherwise there would be a measurement that Bob can make on his system to distinguish (even partially) between the two states, giving him some information about b.

Now is time to take out our quantum information theorist's toolbox and extract one of its magic tools: Uhlmann's theorem! The theorem implies that, if $\rho_B(0) = \rho_B(1)$, then necessarily there exists a unitary U_A on Alice's system such that $U_A \otimes \mathbb{I}_B \, |\Psi(0)\rangle_{AB} = |\Psi(1)\rangle_{AB}$. But this means that Alice can perfectly change her mind, thereby completely breaking the binding property for the protocol.

Rather unfortunately for the fate of quantum multiparty cryptography, it is possible to generalize this argument to show that any protocol for *any* task in multiparty quantum cryptography must be "totally insecure" in the following sense: if the protocol is perfectly secure against a malicious Bob, then it must be that a malicious Alice (interacting with honest Bob) can recover the value $f_A(x, y)$ associated with Bob's input y for *all* possible values of x, simultaneously! (To see why this indeed makes the protocol totally insecure, consider for example the millionaire's problem: Alice would learn if $x > y$ for any x, and could thus perform a quick binary search to learn Bob's fortune y exactly.)

Given such a strong impossibility result, we are left with two possibilities. The first possibility is that we could place limiting assumptions on any malicious party's abilities. In classical cryptography these are mostly computational assumptions, and we give an example in the next section. In quantum

cryptography we can also consider placing physical assumptions on the adversary, such as it having limited storage capabilities. We will explore such assumptions in detail in the next chapter.

The second possibility is to accept the impossibility of *perfect* protocols for multiparty cryptography and instead settling for protocols with a relaxed notion of security, where e.g. Bob can learn "some" information about both Alice's input strings s_0, s_1 in bit commitment, but not all. This is indeed possible, and can be quite useful in spite of the relaxed security condition; we saw an example of this when discussing coin flipping in Section 9.1.

9.4.3 Computationally secure commitments

Week 8, Lecture 8.4,Lecture 2: Computationally secure commitments

We have seen that it is impossible to perfectly implement bit commitment, whether we use quantum information or not. The fact that it is such a useful primitive, however, should encourage us to be creative. In many contexts we would be willing to put up with our usual requirement for perfect, information-theoretic security, and start making assumptions — of course, the fewer the better! For example, we can assume that the malicious party has bounded computational power. This is a very standard assumption in classical cryptography, as indeed very little can be achieved without it (in contrast to quantum cryptography). Of course, here we would only want to make assumptions that hold even if the malicious party has bounded *quantum* computational power. The weakest such assumption under which any interesting cryptographic task is made possible is the existence of *one-way functions*. Informally, a function is one-way if it is easy to evaluate the function on any input (there is an efficient algorithm to compute it), but it is hard to invert the function (given a point in the range of the function, find a pre-image).

There are many candidate constructions of one-way functions, including some that are believed to be hard to invert even for quantum computers. And it turns out that, assuming one-way functions exist, there is a simple protocol for bit commitment that is statistically binding ($p_0 + p_1$ can be made as close to 1 as desired by increasing the amount of communication required in the scheme), and computationally hiding (the hiding property holds as long as it can be assumed that the malicious party cannot invert the one-way function).

Since our focus in this book is on information-theoretic, not computational, security, we won't explain all the details of the protocol. Just for the fun though, let's see the flavor of it. What the protocol actually needs is a special object that can be constructed from a one-way function and is called a *pseudorandom generator* (PRG). For us a PRG is a family of functions $G_n : \{0,1\}^n \to \{0,1\}^{3n}$. These functions have two important properties: (i) for any $x \in \{0,1\}^n$, it is easy for anyone to compute $G_n(x)$, and (ii) it is "computationally infeasible" for anyone to make the difference between the following two distributions U_1 and U_2: U_1 is obtained by computing $G_n(x)$ for a uniformly random $x \in \{0,1\}^n$, and U_2 is obtained by directly returning a uniformly random $y \in \{0,1\}^{3n}$. Note that since U_1 has support size $2^n \ll 2^{3n}$, these distributions are very different; however the assumption is that, if we are only given a number of samples and an amount of time time that scales polynomially with n then it is impossible to make the difference between the two distributions.

PRGs are all we need, let's see the protocol! Recall that Alice has as input a bit $b \in \{0,1\}$, and Bob has no input at all. To commit,

1 Bob selects a uniformly random $r \in \{0,1\}^{3n}$ and sends it to Alice.
2 Alice selects a uniformly random $s \in \{0,1\}^n$ and sends $\sigma = (b \cdot r) \oplus G(s)$ to Bob, where $b \cdot r$ is the string 0^{3n} if $b = 0$ and the string r if $b = 1$.

We can already see that the protocol is *computationally* hiding, because whatever Bob does, Alice sends

him a sample from the distribution U_1 or the distribution $r \oplus U_1$. These distributions are indistinguishable because both are indistinguishable from U_2, by the PRG assumption.

Now to reveal her bit, Alice simply sends s to Bob, and Bob checks if $G(s) \oplus \sigma$ is 0^{3n} or r. Using the fact that the range of G has size 2^n, it is not too hard to check that, except with probability 2^{-n} over Bob's choice of r, Alice will not be able to reveal to a value $b' \neq b$. So the protocol is *statistically* binding, meaning that the chance that Alice violates the binding condition is exponentially small, independently of the amount of time or computational power that she has.

We won't go into the protocol in more detail here, but we hope that it gives you the flavor of how certain tasks can be securely implemented assuming that some problem is computationally hard. (Here, the hard problem is to distinguish U_1 from U_2.)

9.5 Kitaev's lower bound on strong coin flipping

In this section we give a proof of Kitaev's lower bound on the bias of secure protocols for strong coin flipping. The argument relies on simple notions of linear and semidefinite programming with which you may not already be familiar. If so we encourage you to read a bit about these techniques before proceeding. But if you don't have the time then you can skip this section: it is independent from the remainder of the book.

For any coin-flipping protocol, define p_{1*} as the probability that Alice outputs a 1, maximized over all possible (cheating) strategies for Bob. Define p_{*1} symmetrically. Then the condition for the protocol to be a secure strong coin-flipping protocol with bias ε is that both $p_{1*}, p_{*1} \in [1/2 - \varepsilon, 1/2 + \varepsilon]$. For example, for Blum's protocol we have $p_{1*} = p_{*1} = 1$ (make sure you understand why this is the case), and so the protocol is *not* secure.

Theorem 9.5.1 (Kitaev) *For any strong coin-flipping protocol, we have* $p_{1*}p_{*1} \geq \frac{1}{2}$.

Note that the condition in the theorem immediately implies that any strong coin flipping protocol has bias at least $(\sqrt{2}-1)/2$, as claimed. This is because if $x, y \in [0, 1]$ are such that $(1/2+x)(1/2+y) \geq 1/2$ then $\max(x, y) \geq (\sqrt{2} - 1)/2$; this is easy to show by contrapositive (assume both $x, y < (\sqrt{2} - 1)/2$ and reach a contradiction).

Kitaev's theorem applies to both classical and quantum protocols. We'll see the proof for classical protocols first, and then move to the quantum setting. Both proofs have the same structure: Bob's maximum cheating probability can be expressed as the optimum of a linear program (LP) (or semidefinite program (SDP) in the quantum case), and similarly for Alice's. We will show that any feasible solution to the duals of each LP provides an upper bound on the probability of success of the cheating strategy. The crucial insight is that the cheating probabilities need to be considered *together*, through the quantity $p_{1*}p_{*1}$: a good upper bound on this quantity expresses the fact that, either Alice can force Bob to output a 1, or, if she can't, then it must be that Bob can force her to produce a 1. We will obtain a bound on this quantity by taking the product of some of the LP (or SDP) constraints. Let's proceed with the details.

9.5.1 The bound on classical protocols

Fix a classical protocol for strong coin-flipping. Since we are proving a lower bound, i.e. an impossibility result, we need to find a way to represent the most general protocol possible. In general a classical procool can be thought of as a tree. Each node in the tree is labeled by a variable u which represents the transcript

that led to this node. So if we are in a node labeled by u, and Alice plays by sending a message a, then we arrive at node (u, a). The root of the tree, before any message has been sent, is simply labeled as $\emptyset$.

The honest protocol is then given by probabilities $p_A(a|u)$, $p_B(b|u)$, which are Alice's (resp. Bob's) transition probabilities. Suppose first that Alice is honest. Then Bob's maximum cheating probability can be expressed as a *linear program* LP_B. This can be seen as follows. For each node u in the tree introduce a variable $p_B(u)$ which represents the probability of reaching node u when Alice is honest and Bob cheats using some cheating strategy of his choice. Note that each cheating strategy of Bob leads to a $p_B(u)$, but not all $p_B(u)$ may be achievable, given that Alice is being honest. Recall that since we are considering p_{*1} we consider that Bob's goal is to maximize the probability of reaching a leaf labeled with a 1. Denote the set of all such leaves as L_1. We introduce constraints to express the fact that Bob can choose any distribution on edges of the tree when it is his turn to play, but he still has to follow Alice's distribution when it is her turn.

$$
\begin{aligned}
(\mathrm{LP}_B,\ \text{primal}) \qquad & \max \quad \sum_{u \in L_1} p_B(u) & \\
& p_B(u)p(a|u) = p_B(u,a) & \forall a,\ \forall u \text{ node for Alice} \\
& p_B(u) = \sum_b p_B(u,b) & \forall u \text{ node for Bob} \\
& p_B(0) = 1 & \\
& p_B(u) \geq 0 & \forall u
\end{aligned}
$$

If we could solve this linear program we would obtain the value of p_{*1}. However, without knowing the protocol, which determines all the $p(a|u)$, this seems hard. To get information about a linear program we know that it is always wise to compute the dual linear program. In order to do so, introduce variables $Z_A(u, a)$ for the first set of constraints and $Z_A(u)$ for the second set. With a little work the dual can be written in the form

$$
\begin{aligned}
(\mathrm{LP}_B,\ \text{dual}) \qquad & \min \quad Z_A(0) & \\
& Z_A(u) \geq \sum_a p(a|u) Z_A(u,a) & \forall u \text{ node for Alice} \\
& Z_A(u) \geq Z_A(u,b) & \forall b,\ \forall u \text{ node for Bob} \\
& Z_A(u) \geq 1 & \forall u \in L_1
\end{aligned}
$$

What to make of this? For each node u, we can interpret $Z_A(u)$ as the maximum probability with which Bob can cheat, starting at node u.

Now we can proceed in an exactly symmetric manner to introduce another linear program LP_A, this time for a cheating Alice. Taking the dual, we get variables $Z_B(u)$. If we accept to interpret these as maximum cheating probabilities from a given node then we are motivated to introduce the following quantity

$$
F_\ell = \mathrm{E}_{u \sim \ell}\big[Z_A(u) Z_B(u)\big]\ , \tag{9.1}
$$

where $u \sim \ell$ is shorthand for u being taken according to the probability distribution on nodes at depth ℓ which arises from the honest protocol (with both parties playing honestly). In this expression, $Z_A(u)Z_B(u)$ should be interpreted as the bias that cheating parties can achieve, if any of them starts cheating at node u.

Let Z_A, Z_B be optimal solutions to the duals of LP_B and LP_A respectively. The last constraint of the dual implies that without loss of generality we can assume that Z_A and Z_B are both exactly 1 at all leaves

labeled with a 1 (as if they were larger, a better solution to the LP could be obtained by scaling). Hence if n is the last level of the tree, then

$$F_n = p_{1,1} = 1/2 \; . \tag{9.2}$$

This is because we assume that the protocol is correct, and so for a random leaf (taken according to the distribution on leaves obtained by having both parties play honestly) the leaf is labeled with a 1 with probability $1/2$ and a 0 with probability $1/2$.

Now let's use strong duality: the optimum of the primal and dual form of LP_B are equal, and similarly for LP_A. This means that F_0, which is defined as the product of the two dual optimums, is also equal to the product of the two primal optimums, i.e.

$$F_0 = p_{1*} p_{*1} \; . \tag{9.3}$$

To conclude we multiply out the constraints of the two duals to show that for any $\ell \geq 0$, $F_\ell \geq F_{\ell+1}$. So then $F_0 \geq F_n$, which using (9.2) and (9.3) proves Theorem 9.5.1 for the case of classical protocols.

9.5.2 The bound on quantum protocols

The lower bound for quantum protocols is very similar to the lower bound for classical protocols, and if you understood how that works then you already have all of the main ideas in place. The main technical hurdle that we need to overcome is the following: how do we model a general quantum protocol? For a classical protocol we could list all the honest probabilities of sending a certain message m given that the parties are in a certain stage u of the protocol. However, for quantum protocols, there is not such a simple notion of sending a given message. In particular, for a fully general protocol the messages will be in a quantum state, and that quantum state could be entangled with the party's memory! It seems that we are facing an almost-impossible mess...

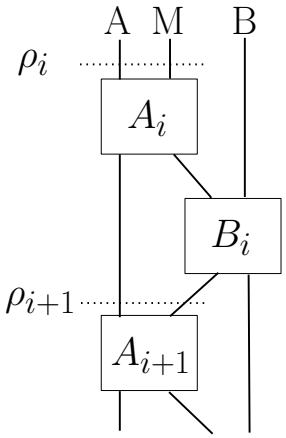

Fig. 9.5 Round i in a quantum coin-flipping protocol. A and B denote registers associated with Alice and Bob's private space respectively, and M with the quantum messages they exchange. At round i Alice applies operation A_i on her local view ρ_i. Then Bob applies B_i, leading to an updated local view ρ_{i+1} for Alice.

Luckily there is a very clean and principled way to go about modeling a general quantum protocol, and this is useful even outside of the present context of coin-flipping. First, let us make the assumption that at each step of the protocol, either Alice or Bob applies an unitary operation on the message space, as well

as their private space. This is without loss of generality because any measurement can be pushed to the end of the protocol using the principle of deferred measurement. We also assume that the parties start in the $|0\dots0\rangle$ state: if Alice or Bob wants to prepare a different state then they can do this as part of their first action. At the first step, Alice (or Bob, whoever plays first) applies a unitary A_1 on her private space and the message space, and she sends the message to Bob. Then Bob applies B_1 to his private space and the message space, which he then returns to Alice, etc. See Figure 9.5 for a pictorial representation of Alice's i-th and $(i+1)$-st actions, and Bob's i-th action, in the protocol. The density matrices ρ_i and ρ_{i+1} represent the joint state of Alice's private space, and the message space, right before she performs her i-th and $(i+1)$-st actions respectively.

At the end of the interaction, Alice measures her entire space using a projective measurement $\{\pi_A, \mathbb{I}-\pi_A\}$, and similarly Bob with $\{\pi_B, \mathbb{I}-\pi_B\}$. Each of them obtains a binary outcome, and this is what they return as their value for the coin flip.

Using this notation, let's first write what it means for the protocol to be correct. Simply writing out the party's actions in turn we see that

$$p_{1,1} = \|(\pi_A \otimes \mathbb{I}_M \otimes \pi_B) B_n A_n \cdots B_1 A_1 |0\dots0\rangle\|^2 ,$$

and we can compute $p_{0,0}$ symmetrically using $(\mathbb{I}-\pi_A), (\mathbb{I}-\pi_B)$ as the measurements. Correctness requires both of these quantities to equal $1/2$.

Proceeding analogously to the classical case, we now need to write an optimization problem that captures the maximum cheating probability for Bob, assuming that Alice plays honestly. The key idea is that instead of using probabilities $p_B(b|u)$ as the variables, we use density matrices ρ_i, where i indexes the messages from Bob to Alice in the protocol, that are supported on Alice's private space and the message space (see Figure 9.5). Since Alice is playing honestly, and Bob can act on the messages, but not on Alice private's space, we know that for any i, $\mathrm{Tr}_M(\rho_{i+1}) = \mathrm{Tr}_M(A_i \rho_i A_{i+1}^\dagger)$. Here we traced out the message register M because it can be affected by cheating Bob, and in general we do not know how. Putting everything together we get the following optimization problem, which places a limit on Bob's probability to cheat in the protocol:

$$\begin{aligned}
(\text{SDP}_B,\ \text{primal}) \qquad & \max \quad \langle \pi_B \otimes \mathbb{I}, \rho_n \rangle & \\
& \mathrm{Tr}_M(\rho_{i+1}) = \mathrm{Tr}_M(A_i \rho_i A_{i+1}^\dagger) & \forall i \\
& \rho_0 = |0\rangle\langle 0|_{A\otimes M} & \\
& \rho_i \geq 0 & \forall i
\end{aligned}$$

This is not a linear program because the last condition is a matrix positivity condition: it says that density matrices must be positive semi-definite. Instead, it is a semidefinite program. Semidefinite programs are a generalization of linear programs, that keep many of the same advantages, including that they can be solved in polynomial time and have a rich duality theory.

The dual of the semidefinite program can be computed mechanically, and we obtain the following formulation:

$$\begin{aligned}
(\text{SDP}_B,\ \text{dual}) \qquad & \min \quad \langle 0|Z_A(0)|0\rangle & \\
& Z_A(i) \otimes \mathbb{I}_M \geq A_{i+1}^\dagger (Z_A(i+1) \otimes \mathbb{I}_M) A_{i+1} & \forall i \\
& Z_A(n) = \pi_A & \\
& Z_A(i) = (Z_A(i))^\dagger & \forall i
\end{aligned}$$

Now let $|\Psi_\ell\rangle$ be the state of all registers, Alice's private space, the message register, *and* Bob's private

register, assuming both parties play honestly. Then the analogue of (9.1) is to define

$$F_\ell = \langle \Psi_\ell | \, Z_A(\ell) \otimes \mathbb{I}_M \otimes Z_B(\ell) \, | \Psi_\ell \rangle \tag{9.4}$$

Correctness then implies the condition $F_n = 1/2$, where n is the total number of rounds, and strong duality implies the condition $F_0 = p_{1*}p_{*0}$, exactly as before. As before it is not hard to show that the relations $F_\ell \geq F_{\ell+1}$ follows from the dual constraints. We then get that $F_0 \geq 1/2$, which is precisely Kitaev's lower bound.

9.6 Chapter notes

The quantum coin flipping protocol discussed in Section 9.1.2 is due to Aharonov et al. [**?**]. A proof of Theorem 9.1.1 can be found in [**?**].

Quantum weak coin flipping with arbitrarily small bias was first discovered in [**?**]. The proof was re-written and simplified by Aharonov et al. [**?**], who showed the existence of a protocol with a number of rounds that is exponential in $1/\varepsilon$, where ε is the bias. Subsequently an explicit description of the protocols was given in [**?**]. Miller [**?**] showed that a polynomial dependence of the number of rounds on $1/\varepsilon$ is necessary. It remains an open question what is the optimal round complexity of quantum weak coin flipping.

Chailloux and Kerenidis [**?**] showed that any weak coin flipping protocol with bias ε could be used to build a strong coin flipping protocol with bias $(\sqrt{2}-1)/2 + O(\varepsilon)$, thereby matching Kitaev's lower bound.

Good references to learn more on the topic of SFE and more generally multiparty cryptograhy include a survey by Goldreich [**?**] (especially Section 7) and one by Lindell [**?**]. For the case of quantum protocols, it is only recently that a satisfactory definition has been introduced; see [**?**] for the notion of universal composability or [**?**] for a weaker but perhaps more approachable definition.

The "unconditionally secure" bit commitment protocol of Brassard et al. appeared in [**?**]. The impossibility of (quantum) bit commitment is due to Mayers and Lo [**?**, **?**]. For an approachable, self-contained proof we recommend the detailed notes by Watrous [**?**]. The impossibility of perfectly secure two-party cryptography, even using quantum information, is shown in [**?**]. For much more on computational security and how to implement various cryptographic tasks classically under computational assumptions we recommend the freely available lecture notes [**?**]; see in particular Section 4.7 for a discussion of bit commitment.

The proof of Kitaev's lower bound on strong quantum coin-flipping given in Section 9.5 is due to Gutoski and Watrous [**?**].