5

Publicly Verifiable, Non-Interactive Arguments via Fiat-Shamir

Recall from Section 3.3 that in a public-coin interactive proof or argument, any coin tossed by the verifier $\mathcal{V}$ is made visible to the prover $\mathcal{P}$ as soon as it is tossed. These coin tosses are interpreted as "random challenges" sent by $\mathcal{V}$ to $\mathcal{P}$, and in a public-coin protocol they are, without loss of generality, the only messages sent from $\mathcal{V}$ to $\mathcal{P}$.[1]

The Fiat-Shamir transformation [111] takes any public-coin protocol $\mathcal{I}$ and transforms it into a non-interactive, publicly verifiable protocol $\mathcal{Q}$. To describe the transformation and analyze its security, it is helpful to introduce an idealized setting called the random oracle model.

5.1 The Random Oracle Model

The random oracle model (ROM) [29], [111] is an idealized setting meant to capture the fact that cryptographers have developed hash functions (e.g., SHA-3 or BLAKE3) that efficient algorithms seem totally unable to distinguish from random functions. By a random function R mapping some domain $\mathcal{D}$ to the κ-bit range $\{0,1\}^\kappa$, we mean the following: on

[1] In a public-coin protocol, any $\mathcal{V} \to \mathcal{P}$ messages other than $\mathcal{V}$'s random coin tosses can be omitted: they must be deterministic functions of $\mathcal{V}$'s coin tosses, and hence $\mathcal{P}$ can derive such messages on its own.

any input $x \in \mathcal{D}$, R chooses its output $R(x)$ uniformly at random from $\{0,1\}^\kappa$.

Accordingly, the ROM simply assumes that the prover and verifier have query access to a random function R. This means that there is an oracle (called a random oracle) such that the prover and verifier can submit any query x to the oracle, and the oracle will return $R(x)$. That is, for each query $x \in \mathcal{D}$ posed to the oracle, the oracle makes an independent random choice to determine $R(x)$ and responds with that value. It keeps a record of its responses to make sure that it repeats the same response if x is queried again.

The random oracle assumption is not valid in the real world, as specifying a random function R requires $|\mathcal{D}| \cdot \kappa$ bits—essentially one must list the value $R(x)$ for every input $x \in \mathcal{D}$—which is totally impractical given that $|\mathcal{D}|$ must be huge to ensure cryptographic security levels (e.g., $|\mathcal{D}| \geq 2^{256}$ or larger). In the real world, the random oracle is replaced with a concrete hash function like SHA-3, which is succinctly specified via, e.g., a small circuit or computer program that evaluates the hash function on any input. In principle, it may be possible for a cheating prover in the real world to exploit access to this succinct representation to break the security of the protocol, even if the protocol is secure in the random oracle model. However, protocols that are proven secure in the random oracle model are often considered secure in practice, and indeed no deployed protocols have been broken in this manner.[2]

[2]The relationship between security in the random oracle model and security in the real world has been the subject of considerable debate and criticism. Indeed, a series of works has established very strong negative results regarding the (lack of) implications of random oracle model security. In particular, various protocols have been constructed that are secure in the random oracle model but not secure when the random oracle is replaced with *any* concrete hash function [26], [91], [134], [139], [199]. However, these protocols and functionalities are typically contrived [170]. For two entertaining and diametrically opposed perspectives, the interested reader is directed to [129], [170].

5.2 The Fiat-Shamir Transformation

Recall that the purpose of the Fiat-Shamir transformation [111] is to take any public-coin IP or argument $\mathcal{I}$ and transform it into a non-interactive, publicly verifiable protocol $\mathcal{Q}$ in the random oracle model.

Some Approaches That Do Not Quite Work. The Fiat-Shamir transformation mimics the transformation described in Section 3.3 that transforms any interactive proof system with a deterministic verifier into a non-interactive proof system. In that transformation, the non-interactive prover leverages the total predictability of the interactive verifier's messages to compute those messages itself on behalf of the verifier. This eliminates the need for the verifier to actually send any messages to the prover. In particular, it means that the non-interactive proof can simply specify an accepting transcript from the interactive protocol (i.e., specify the first message sent by the prover in the interactive protocol, followed by the verifier's response to that message, followed by the prover's second message, and so on until the protocol terminates).

In the setting of this section, the verifier's messages in $\mathcal{I}$ are not predictable. But since $\mathcal{I}$ is public coin, the verifier's messages in $\mathcal{I}$ come from a known distribution; specifically, the uniform distribution. So a naive attempt to render the protocol non-interactive would be to ask the prover to determine the verifier's messages itself, by drawing each message at random from the uniform distribution, independent of all previous messages sent in the protocol. But this does not work because the prover is untrusted, and hence there is no way to force the prover to actually draw the verifier's challenges from the appropriate distribution.

A second approach that attempts to address the above issue is to have $\mathcal{Q}$ use the random oracle to determine the verifier's message r_i in round i of $\mathcal{I}$. This will ensure that each challenge is indeed uniformly distributed. A naive attempt at implementing this second approach would be to select r_i in $\mathcal{Q}$ by evaluating the random oracle at input i. But this attempt is also unsound. The problem is that, although this ensures each of the verifier's messages r_i are uniformly distributed, it does not ensure that they are independent of the prover's messages $g_1, \ldots, g_i$ from rounds $1, 2, \ldots, i$ of $\mathcal{I}$. Specifically, the prover in $\mathcal{Q}$ can learn all

of the verifier's messages $r_1, r_2, \ldots$ in advance (by simply querying the random oracle at the predetermined points $1, 2, \ldots$) and then choose the prover messages in $\mathcal{I}$ in a manner that depends on these values. Since the IP $\mathcal{I}$ is not sound if the prover knows r_i in advance of sending its ith message g_i, the resulting non-interactive argument is not sound.

The above issue can be made more concrete by imagining that $\mathcal{I}$ is the sum-check protocol applied to an ℓ-variate polynomial g over $\mathbb{F}$. Consider a cheating prover $\mathcal{P}$ who begins the protocol with a false claim C for the value $\sum_{x \in \{0,1\}^\ell} g(x)$. Suppose in round one of the sum-check protocol, before sending its round-one message polynomial g_1, $\mathcal{P}$ knows what will be the verifier's round-one message $r_1 \in \mathbb{F}$. Then the prover can trick the verifier as follows. If s_1 is the message that the *honest* prover would send in round one, $\mathcal{P}$ can send a polynomial g_1 such that

$$g_1(0) + g_1(1) = C \quad \text{and} \quad g_1(r_1) = s_1(r_1), \tag{5.1}$$

where recall (Equation (4.2)) that

$$s_1(X_1) := \sum_{(x_2,\ldots,x_v) \in \{0,1\}^{v-1}} g(X_1, x_2, \ldots, x_v).$$

Note that such a polynomial is guaranteed to exist so long as g_1 is permitted to have degree at least one. From that point on in $\mathcal{I}$, the cheating prover $\mathcal{P}$ can send the same messages as the honest prover, and thereby pass all of the verifier's checks. In the naive attempt at implementing the second approach to obtaining a non-interactive protocol above, the prover in $\mathcal{Q}$ will be able to simulate this attack on $\mathcal{I}$. This is because the prover in $\mathcal{Q}$ can learn r_1 by simply querying the random oracle at the input 1, and then choosing g_1 to satisfy Equation (5.1) above.

To prevent this attack on soundness, the Fiat-Shamir transformation ensures that the verifier's challenge r_i in round i of $\mathcal{I}$ is determined by querying the random oracle at an input that depends on the prover's i'th message g_i. This means that the prover in $\mathcal{Q}$ can only simulate the aforementioned attack on $\mathcal{I}$ if the prover can find a g_1 satisfying Equation (5.1) with r_1 equal to evaluation of the random oracle at the appropriate query point (which, as previously mentioned, includes g_1). Intuitively, for the prover in $\mathcal{Q}$ to find such a g_1, a vast number of queries

to the random oracle are required, because the output of the random oracle is totally random, and for each g_1 there are a tiny number of values of r_1 satisfying Equation (5.1).

Complete Description of the Fiat-Shamir Transformation. The Fiat-Shamir transformation replaces each of the verifier's messages from the interactive protocol $\mathcal{I}$ with a value derived from the random oracle in the following manner: in $\mathcal{Q}$, the verifier's message in round i of $\mathcal{I}$ is determined by querying the random oracle, where the query point is the the list of messages sent by the prover in rounds $1, \ldots, i$. As in the naive attempt above, this eliminates the need for the verifier to send any information to the prover—the prover can simply send a single message containing the transcript of the entire protocol (i.e., a list of all messages exchanged by the prover in the interactive protocol, with the verifier's random coin tosses in the transcript replaced with the random oracle evaluations just described). See Figure 5.1.

A Concrete Optimization. When applying the Fiat-Shamir transformation to many-round interactive protocols, it is often implemented using a technique called *hash chaining*. This means that, rather than choosing the round-i verifier challenge r_i in the interactive protocol to be the hash (or random oracle evaluation) of all preceding prover messages

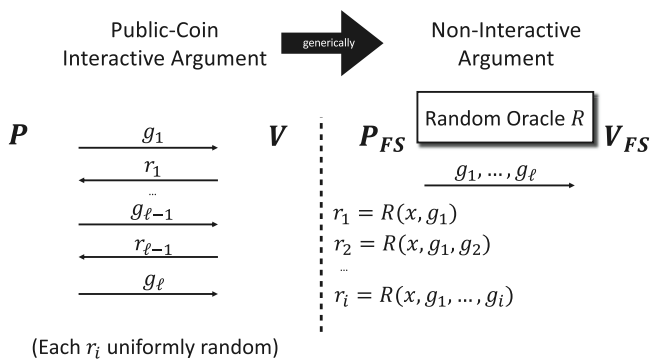

Figure 5.1: Depiction of the Fiat-Shamir transformation. Image courtesy of Ron Rothblum [218].

$g_1, \ldots, g_i$, one instead chooses r_i to be a hash only of (i, r_{i-1}, g_i). This reduces the cost of hashing in practice, because it keeps the inputs at which the hash function is evaluated short. This variant of Fiat-Shamir also shown secure in the random oracle model.

Avoiding a Common Vulnerability. For the Fiat-Shamir transformation to be secure in settings where an adversary can choose the input x to the IP or argument, it is essential that x be appended to the list that is hashed in each round. This property of soundness against adversaries that can choose x is called adaptive soundness. Some real-world implementations of the Fiat-Shamir transformation have missed this detail, leading to attacks [51], [146]. In fact, this precise error was recently identified in several popular SNARK deployments, leading to critical vulnerabilities.[3] Sometimes, the version of Fiat-Shamir that includes x in the hashing is called *strong* Fiat-Shamir, while the version that omits it is called *weak* Fiat-Shamir.

Here is a sketch of the attack if the adversary can choose x and x is not hashed within the Fiat-Shamir transformation. For concreteness, consider a prover applying the GKR protocol to establish that $\mathcal{C}(x) = y$. In the GKR protocol, the verifier $\mathcal{V}$ completely ignores the input $x \in \mathbb{F}^n$ until the final check in the protocol, when $\mathcal{V}$ checks that the multilinear extension $\widetilde{x}$ of x evaluated at some randomly chosen point r equals some value c derived from previous rounds. The adversary can easily generate a transcript for the Fiat-Shamir-ed protocol that passes all of the verifier's checks except the final one. To pass the final check, the adversary can choose any input $x \in \mathbb{F}^n$ such that $\tilde{x}(r) = c$ (such an input $\tilde{x}$ can be identified in linear time). The transcript convinces the verifier of the Fiat-Shamir-ed protocol to accept the claim that $\mathcal{C}(x) = y$. Yet there is no guarantee that $\mathcal{C}(x) = y$, as x may be an arbitrary input satisfying $\tilde{x}(r) = c$. See Exercise 5.2, which asks the reader to work through the details of this attack.

Note that in this attack, the prover does not necessarily have "perfect control" over the inputs x for which it is able to produce convincing

[3] https://blog.trailofbits.com/2022/04/13/part-1-coordinated-disclosure-of-vulnerabilities-affecting-girault-bulletproofs-and-PlonK/.

"proofs" that $\mathcal{C}(x) = y$. This is because x is constrained to satisfy $\tilde{x}(r) = c$ for some values r and c that depend on the random oracle. This may render the attack somewhat benign in some applications.[4] Nonetheless, practitioners should take care to avoid this vulnerability, especially since including x in the hashing is rarely a significant cost in practice.

5.3 Security of the Transformation

It has long been known that when the Fiat-Shamir transformation is applied to a *constant-round* public-coin IP or argument $\mathcal{I}$ with negligible soundness error,[5] the resulting non-interactive proof $\mathcal{Q}$ in the random oracle model is sound against cheating provers that run in polynomial time [211]. More quantitatively, if $\mathcal{I}$ consists of t rounds, any prover $\mathcal{P}$ for $\mathcal{Q}$ that can convince the verifier to accept input x with probability ε and runs in time T can be transformed into a prover $\mathcal{P}'$ for $\mathcal{I}$ that convinces the verifier to accept input x with probability at least $(\varepsilon/T)^{O(t)}$. If t is constant, this is $\text{poly}(1/\varepsilon, 1/T)$, which is non-negligible.[6] In fact, we prove this result at the end of this section for 3-message protocols $\mathcal{I}$ (Theorem 5.1). However, the runtime of $\mathcal{P}'$ grows exponentially with the number of rounds t in $\mathcal{I}$, and the IPs covered in this section all require at least logarithmically many rounds. Recently, a better understanding of the soundness of $\mathcal{Q}$ has been developed for such many-round protocols $\mathcal{I}$.

Specifically, it is now known that if a public-coin interactive proof $\mathcal{I}$ for a language $\mathcal{L}$ satisfies a property called *round-by-round soundness*

[4] An illustrative example: in some applications the only "sensible" inputs x are bit-vectors $x \in \{0,1\}^n$. The attack described above will efficiently identify an $x \in \mathbb{F}^n$ along with a convincing "proof" that $\mathcal{C}(x) = y$, but it may not be the case that all entries of x are in $\{0,1\}$. This may mean that the attacker is only able to generate "convincing proofs" for false statements about "nonsense vectors" $x \in \mathbb{F}^n \setminus \{0,1\}^n$.

[5] Throughout this monograph, negligible means any quantity smaller than the reciprocal of any fixed polynomial in the input length n or a security parameter λ. Non-negligible means any quantity that is *at least* the reciprocal of some fixed polynomial in n or λ. Computationally-bounded adversaries are assumed to run in time polynomial in λ and n.

[6] If $\mathcal{I}$ is an argument rather than a proof, then soundness of $\mathcal{Q}$ in the random oracle model will also inherit any computational hardness assumptions on which soundness of $\mathcal{I}$ is based.

then $\mathcal{Q}$ is sound in the random oracle model [44], [88]. Here, $\mathcal{I}$ satisfies round-by-round soundness if the following properties hold: (1) At any stage of any execution of $\mathcal{I}$, there is a well-defined state (depending on the partial transcript at that stage of the execution) and some states are "doomed", in the sense that once the protocol $\mathcal{I}$ is in a doomed state, it will (except with negligible probability) forever remain doomed, no matter the strategy executed by the prover in $\mathcal{I}$. (2) If $x \notin \mathcal{L}$, then the initial state of $\mathcal{I}$ is doomed. (3) If at the end of the interaction the state is doomed, then the verifier will reject.[7]

Canetti *et al.* [88] showed that the GKR protocol (and any other interactive proof based on the sum-check protocol) satisfy round-by-round soundness, and hence applying the Fiat-Shamir transformation to it yields a non-interactive proof that is secure in the random oracle model.[8]

Here is some rough intuition for why round-by-round soundness of the IP $\mathcal{I}$ implies soundness of the non-interactive proof $\mathcal{Q}$ in the random oracle model. The only way a cheating prover in $\mathcal{I}$ can convince the verifier of a false statement is to "get lucky", in the sense that the verifier's random coin tosses in $\mathcal{I}$ happen to fall into some small set of "bad" coin tosses B that eventually force the protocol into a *non-doomed* state. Round-by-round soundness implies that B is small. Because a random oracle is by definition totally unpredictable and unstructured, in $\mathcal{Q}$ roughly speaking all that a cheating prover can do to find an accepting transcript is to iterate over possible prover messages/transcripts for the IP $\mathcal{I}$ in an arbitrary order, and stop when he identifies one where the random oracle happens to return a sequence of values falling in B. Of course, this isn't quite true: a malicious prover in $\mathcal{Q}$ is also capable of executing a so-called *state-restoration* attack [44]

[7] For illustration, the canonical example of an IP with negligible soundness error that does *not* satisfy round-by-round soundness is to take any IP with soundness error 1/3, and sequentially repeat it n times. This yields a protocol with at least n rounds and soundness error $1/3^{-\Omega(n)}$, yet it is not round-by-round sound. And indeed, applying the Fiat-Shamir transformation to this protocol does *not* yield a sound argument system in the random oracle model [218]. See Exercise 5.1.

[8] In fact, Canetti *et al.* [88] also show that, using parallel repetition, *any* public-coin IP can be transformed into a different public-coin IP that satisfies round-by-round soundness.

(also sometimes called a *grinding attack*), which means that the prover in $\mathcal{Q}$ can "rewind" any interaction with the verifier of $\mathcal{I}$ to an earlier point of the interaction and "try out" sending a different response to the last message sent by the verifier in this partial transcript for $\mathcal{I}$ (see Section 5.3.1 for additional discussion of this attack). The prover may hope that by trying out a different response, this will cause the random oracle to output a non-doomed value. However, round-by-round soundness of $\mathcal{I}$ precisely guarantees that such an attack is unlikely to succeed: once in a doomed state of $\mathcal{I}$, no prover strategy can "escape" doom except with negligible probability.

In summary, applying the Fiat-Shamir Transformation to a public coin interactive protocol with negligible round-by-round soundness error yields a non-interactive argument in the random oracle model with negligible soundness error against efficient provers. The protocol can then be heuristically instantiated in the "real world" by replacing the random oracle with a cryptographic hash function.

As discussed next, there are nuances regarding what is an appropriate level of security for interactive protocols vs. the non-interactive arguments that result after applying the Fiat-Shamir transformation.

5.3.1 "Bits of Security": Statistical vs. Computational

Statistical, Computational, and Interactive Security. As we have seen (Section 4), interactive protocols can satisfy *statistical* (i.e., information-theoretic) security. The logarithm of the soundness error of an information-theoretically secure protocol is referred to as the number of *bits of statistical security*.

In contrast, the security level of a non-interactive argument is measured by the amount of work that must be done to find a convincing "proof" of a false statement. Similar to other cryptographic primitives like digital signatures and collision-resistant hash functions, the logarithm of this amount of work is referred to as the number of *bits of computational security*. For example, 30 bits of security implies that $2^{30} \approx 1$ billion "steps of work" are required to attack the argument system. This is inherently an approximate measure of real-world security

because the notion of one step of work can vary, and practical considerations like memory requirements or opportunities for parallelism are not considered.

Later in this text, we will see many examples of succinct *interactive* arguments. Although only computationally rather than statistically sound, many of these arguments have the property that adversaries that cannot break a cryptographic primitive (e.g., cannot find a collision in a collision-resistant hash function) also cannot convince the argument-system verifier to accept a false statement with probability more than 2^{-s} for some value s. In this situation, some practitioners refer informally to s as the number of *bits of interactive security* of the argument.

Appropriate Security Levels for Interactive vs. Non-Interactive Arguments. For reasons discussed shortly, non-interactive arguments are generally recommended to be deployed with at least 100 or 128 bits of computational security [261]. In contrast, it may be appropriate in some contexts to set statistical or interactive security levels lower.

The key difference is that, with statistical or interactive security, the cheating prover has to actually interact with the verifier in order to "attempt" an attack that will succeed with only tiny probability. This is because the cheating prover in an interactive protocol is hoping to get a "lucky" verifier challenge (i.e., one that leads the verifier to accept the prover's false claim), and the prover does not know whether or not the verifier's challenge will be lucky until after sending one or more messages to the verifier and receiving challenges in response.

For example, suppose that an interactive protocol is run at 60 bits of statistical or interactive security. This means that each attempted attack succeeds with probability at most 2^{-60}. So after, say, 2^{30} attempted attacks, the probability that *any* of the attempts succeed is at most $2^{-60} \cdot 2^{30} = 2^{-30}$. It is unlikely that a verifier will continue interacting with a prover that has tried and failed to convince her to accept false statements 2^{30} times. Moreover, due to round-trip delays involved in interactive protocols, executing a large number of attacks may take an infeasibly large amount of time. For example, if every attempted attack executes in one second, then 2^{30} attempts would take more than 30 years to execute. For these reasons, 60 bits of statistical or interactive

security may be sufficient in some contexts—specifically, those where a successful attack would not be totally catastrophic, and where, for the reasons above, attacks cannot be attempted too many times.

In contrast, with non-interactive arguments, a cheating prover can "silently" attack a protocol, without any interaction with the verifier. For example, if applying the Fiat-Shamir transformation to a 3-message interactive protocol as considered in Figure 5.2 below, the canonical "grinding attack" on the resulting non-interactive argument involves the prover attempting to "guess" a first message α that yields a "lucky" verifier message $R(x, \alpha)$, in the sense that the prover can efficiently find a response γ such that $(\alpha, R(x, \alpha), \gamma)$ is an accepting transcript.

Suppose the original protocol had only 60 bits of statistical or interactive security. Then a cheating prover executing a grinding attack on the non-interactive argument only needs to try about 2^{60} first messages α before it finds one such that $R(x, \alpha)$ is "lucky". When instantiating the Fiat-Shamir transformation with a concrete hash function, the computational bottleneck in this attack may be simply performing 2^{60} hash evaluations. This number of hash evaluations is entirely feasible for modern computers.[9] Indeed, in 2020, the cost of computing just shy of 2^{64} SHA-1 evaluations using GPUs was $45,000 [181]. As another data point, as of 2022, bitcoin's network hash rate was about 2^{67} hash evaluations per second, meaning bitcoin miners as a whole were performing 2^{80} SHA-256 evaluations every 2 hours. Of course, this very large number of hashes is due to vast investment in ASICs for bitcoin mining.

In summary, if one is applying the Fiat-Shamir transformation to render an interactive protocol non-interactive, the interactive protocol should be configured to over 80 bits of statistical or interactive security if one wishes to ensure that the canonical grinding attack on the resulting non-interactive protocol is out of the reach of modern hardware.

[9] More precisely, a grinding attack that tries $T < 2^{60}$ different values of α succeeds with probability roughly $T \cdot 2^{-60}$. This matches the lower bound shown in the proof of Theorem 5.1. Specifically, Theorem 5.1 shows that if the Fiat-Shamir transformation is applied to a 3-message interactive protocol with statistical soundness error 2^{-60}, then any attack on the Fiat-Shamir-ed protocol that treats the hash function as a random oracle, runs in time at most T, and succeeds with probability ϵ must satisfy $\epsilon/T < 2^{-60}$.

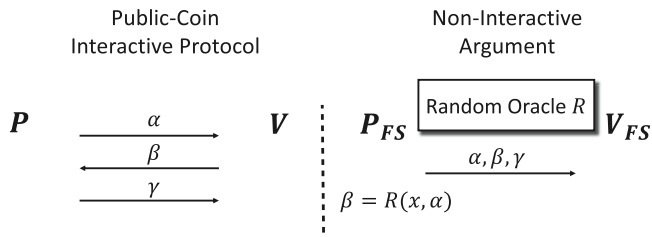

Figure 5.2: Depiction of the Fiat-Shamir transformation applied to a 3-message interactive proof or argument as in the proof of Theorem 5.1. Image courtesy of Ron Rothblum [218].

5.3.2 Soundness in the Random Oracle Model for Constant-Round Protocols

Theorem 5.1. Let $\mathcal{I}$ be a *constant-round* public-coin IP or argument with negligible soundness error, and let $\mathcal{Q}$ be the non-interactive protocol in the random oracle model obtained by applying the Fiat-Shamir transformation to $\mathcal{I}$. Then $\mathcal{Q}$ has negligible computational soundness error. That is, no prover running in polynomial time can convince the verifier in $\mathcal{Q}$ of a false statement with non-negligible probability.

Proof. For simplicity, we will only prove the result in the case where $\mathcal{I}$ is a 3-message protocol where the prover speaks first. See Figure 5.2 for a depiction of the Fiat-Shamir transformation in this setting and the notation we will use during the proof.

We will show that, for any input x, if $\mathcal{P}_{\mathcal{FS}}$ is a prover that runs in time T and convinces the verifier in $\mathcal{Q}$ to accept on input x with probability at least ϵ (where the probability is over the choice of random oracle), then there is a prover $\mathcal{P}^*$ for $\mathcal{I}$ that convinces the verifier in $\mathcal{I}$ to accept with probability at least $\varepsilon^* \geq \Omega(\varepsilon/T)$ (where the probability is over the choice of the verifier's random challenges in $\mathcal{I}$). Moreover, $\mathcal{P}^*$ has essentially the same runtime as $\mathcal{P}_{\mathcal{FS}}$. The theorem follows, because if ϵ is non-negligible and T is polynomial in the size of the input, then ε^* is non-negligible as well.

Handling Restricted $\mathcal{P}_{\mathcal{FS}}$ Behavior. The rough idea is that there isn't much $\mathcal{P}_{\mathcal{FS}}$ can do to find an accepting transcript (α, β, γ) with

$\beta = R(x, \alpha)$ other than to mimic a successful prover strategy $\mathcal{P}$ for $\mathcal{Q}$, setting α to be the first message sent by $\mathcal{P}$, setting β to be $R(x, \alpha)$, and setting γ to be $\mathcal{P}$'s response to the challenge β.[10] If this is indeed how $\mathcal{P}_{\mathcal{FS}}$ behaved, it would be easy for $\mathcal{P}^*$ to "pull" the prover strategy $\mathcal{P}$ for $\mathcal{Q}$ out of $\mathcal{P}_{\mathcal{FS}}$ as follows: $\mathcal{P}^*$ runs $\mathcal{P}_{\mathcal{FS}}$, up until the point where $\mathcal{P}_{\mathcal{FS}}$ makes its (only) query, of the form (x, α), to the random oracle. $\mathcal{P}^*$ sends α to the verifier $\mathcal{V}$ in $\mathcal{I}$, who responds with a challenge β. $\mathcal{P}^*$ uses β as the response of the random oracle to $\mathcal{P}_{\mathcal{FS}}$'s query. $\mathcal{P}^*$ then continues running $\mathcal{P}_{\mathcal{FS}}$ until $\mathcal{P}_{\mathcal{FS}}$ terminates.

Since $\mathcal{I}$ is public coin, $\mathcal{V}$ chooses β uniformly random, which means that β is distributed appropriately to be treated as a response of the random oracle. Hence, with probability at least ε, $\mathcal{P}_{\mathcal{FS}}$ will produce an accepting transcript of the form (α, β, γ). In this case, $\mathcal{P}^*$ sends γ as its final message in $\mathcal{I}$, and the verifier accepts because (α, β, γ) is an accepting transcript. This ensures that $\mathcal{P}^*$ convinces the verifier in $\mathcal{I}$ to accept with the same probability that $\mathcal{P}_{\mathcal{FS}}$ outputs an accepting transcript, which is at least ε by assumption.

The General Case: Overview. In the general case, $\mathcal{P}_{\mathcal{FS}}$ may not behave in the manner above. In particular, $\mathcal{P}_{\mathcal{FS}}$ may ask the random oracle *many* queries, though no more than T of them, since $\mathcal{P}_{\mathcal{FS}}$ runs in time at most T. This means that it is not obvious which of the queries (x, α) $\mathcal{P}^*$ should forward to $\mathcal{V}$ as its first message α. Fortunately, we will show that it suffices for $\mathcal{P}^*$ to simply pick one of $\mathcal{P}_{\mathcal{FS}}$'s queries at random. Essentially, $\mathcal{P}^*$ will pick the "right" query with probability at least $1/T$, leading $\mathcal{P}^*$ to convince $\mathcal{V}$ to accept input x with probability at least ε/T.

What We Can Assume About $\mathcal{P}_{\mathcal{FS}}$ Without Loss of Generality. Let us assume that $\mathcal{P}_{\mathcal{FS}}$ always makes exactly T queries to the random oracle (this can be ensured modifying $\mathcal{P}_{\mathcal{FS}}$ to ask "dummy queries" as necessary to ensure that it always makes exactly T queries to the random oracle R). Let us further assume that all queries $\mathcal{P}_{\mathcal{FS}}$ makes

[10] As explained in Section 5.3.1, this isn't actually true, as $\mathcal{P}_{\mathcal{FS}}$ can also run state-restoration attacks a.k.a. grinding attacks, an issue with which the formal proof below must grapple.

are distinct (there is never a reason for $\mathcal{P}_{\mathcal{FS}}$ to query the oracle at the same location twice, since the oracle will respond with the same value both times). Finally, we assume that whenever $\mathcal{P}_{\mathcal{FS}}$ successfully outputs an accepting transcript (α, β, γ) with $\beta = R(x, \alpha)$, then at least one of $\mathcal{P}_{\mathcal{FS}}$'s T queries to R was at point (x, α). This can be ensured by modifying $\mathcal{P}_{\mathcal{FS}}$ to always query (x, α) before outputting the transcript (α, β, γ), if (x, α) has not already been queried, and making a "dummy query" otherwise.

Complete Description of $\mathcal{P}^*$. $\mathcal{P}^*$ begins by picking a random integer $i \in \{1, \ldots, T\}$. $\mathcal{P}^*$ runs $\mathcal{P}_{\mathcal{FS}}$ up until its i'th query to the random oracle, choosing the random oracle's responses to queries $1, \ldots, i-1$ uniformly at random. If the ith query is of the form (x, α) for some α, $\mathcal{P}^*$ sends α to $\mathcal{V}$ as its first message, and receives a response β from $\mathcal{V}$.[11] $\mathcal{P}^*$ uses β as the response of the random oracle to query (x, α). $\mathcal{P}^*$ then continues running $\mathcal{P}_{\mathcal{FS}}$, choosing the random oracle's responses to queries $i+1, \ldots, T$ uniformly at random. If $\mathcal{P}_{\mathcal{FS}}$ outputs an accepting transcript of the form (α, β, γ), then $\mathcal{P}^*$ sends γ to $\mathcal{V}$, which convinces $\mathcal{V}$ to accept.

Analysis of Success Probability for $\mathcal{P}^*$. As in the restricted case, since $\mathcal{I}$ is public coin, $\mathcal{V}$ chooses β uniformly random, which means that β is distributed appropriately to be treated as a response of the random oracle. This means that $\mathcal{P}_{\mathcal{FS}}$ outputs an accepting transcript of the form $(\alpha, R(x, \alpha), \gamma)$ with probability at least ε. In this event, $\mathcal{P}^*$ convinces $\mathcal{V}$ to accept whenever $\mathcal{P}_{\mathcal{FS}}$'s ith query to R was (x, α). Since we have assumed that $\mathcal{P}_{\mathcal{FS}}$ makes exactly T queries, all of which are distinct, and one of those queries is of the form (x, α), this occurs with probability exactly $1/T$. Hence, $\mathcal{P}^*$ convinces $\mathcal{V}$ to accept with probability at least ε/T. □

[11] If the ith query is not of the form (i, α), $\mathcal{P}^*$ aborts, i.e., $\mathcal{P}^*$ gives up trying to convince $\mathcal{V}$ to accept.

5.3.3 Fiat-Shamir Preserves Knowledge-Soundness in the Random Oracle Model

Theorem 5.1 roughly shows that the Fiat-Shamir transformation renders any constant-round IP or argument non-interactive in the random oracle model while preserving soundness. Later in this monograph, we will be concerned with a strengthening of soundness called *knowledge-soundness* that is relevant when the prover is claiming to *know* a witness satisfying a specified property (see Section 7.4). In the random oracle model, the Fiat-Shamir transformation does preserve knowledge-soundness, at least when applied to specific important argument systems. We cover two important examples of these results later in this survey: Section 9.2.1 shows that the Fiat-Shamir transformation preserves knowledge-soundness when applied to succinct arguments obtained from *PCPs* and *IOPs*. Section 12.2.3 establishes a similar result when the transformation is applied to a different class of arguments, called Σ-*protocols* (introduced in Section 12.2.1). A brief discussion of extensions to super-constant round variants of Σ-protocols can be found at the end of Section 14.4.2.

5.3.4 Fiat-Shamir in the Plain Model

Chaum and Impagliazzo, and Canetti *et al.* [91] identified a property called correlation-intractability (CI) such that instantiating the Fiat-Shamir transformation in the plain model results in a sound argument when the concrete hash function h is chosen at random from a hash family $\mathcal{H}$ satisfying CI. Below, we explain in more detail what CI means, before describing recent results that construct CI hash families based on standard cryptographic assumptions.

What is Correlation-Intractability? Let R denote some *property* of pairs $(y, h(y))$. A hash family $\mathcal{H}$ satisfies CI for R if it is computationally infeasible to find a pair $(y, h(y))$ satisfying property R.

Suppose $\mathcal{I}$ is an IP or argument for a language $\mathcal{L}$ such that $\mathcal{I}$ satisfies round-by-round soundness. Let R denote the relation capturing "success" of a cheating prover for the Fiat-Shamir transformation $\mathcal{Q}$ of $\mathcal{I}$. That is, R consists of all tuples $(y, h(y))$ such that $y = (x, g_1, \ldots, g_i)$ with

$x \notin \mathcal{L}$ and the following holds. Let $g_1, \ldots, g_i$ be interpreted as prover messages in the first i rounds of $\mathcal{I}$ when run on input x, with round-j verifier message equal to $h(x, g_1, \ldots, g_j)$. Then we define R to include $(y, h(y))$ if $\mathcal{I}$ is in a doomed state at the start of round i, but enters a non-doomed state if the ith verifier challenge is $h(y)$.

A cheating prover in the Fiat-Shamir-ed protocol $\mathcal{Q}$ *must* find some pair $(y, h(y))$ satisfying property R to find a convincing "proof" of membership in $\mathcal{L}$ for some $x \notin \mathcal{L}$. If $\mathcal{H}$ satisfies CI for R, then this task is intractable—no polynomial time cheating prover can find a convincing proof of a false statement with non-negligible probability.

Recent Results Constructing CI Hash Families. An exciting recent line of work [76], [88]–[90], [150], [151], [158], [161], [185], [207] has constructed CI hash families for various natural classes of IPs and arguments, with the CI property holding under standard cryptographic assumptions. In particular, [151], [207] construct CI families for various natural classes of IPs and arguments assuming the Learning With Errors (LWE) assumption, upon which many lattice-based cryptosystems are based. The constructions of CI hash families in the aforementioned works have the flavor of *fully homomorphic encryption* (FHE) schemes, which are currently highly computationally intensive, much more so than the prover and verifier in interactive proofs such as the GKR protocol. Hence, these hash families are not practical for use in obtaining non-interactive arguments. However, it is plausible that cryptographic hash families used in practice actually satisfy the relevant notions of correlation-intractability.

The aforementioned results apply, for example, to the GKR protocol. They also apply to various zero-knowledge proofs of theoretical and historical (but not practical) importance for **NP**-complete languages [58], [131] (we formally introduce the notion of zero-knowledge in Section 11). Note that these zero-knowledge proofs are not *succinct*. This means that the proof length is not shorter than the trivial (non-zero-knowledge) **NP** proof system in which the prover sends a classical static proof, i.e., an **NP** witness, for the validity of the claim at hand, and the verifier deterministically checks the proof.

Fiat-Shamir and succinct public-coin arguments. There is great interest in obtaining analogous results for the *succinct* public-coin interactive arguments described later in this survey, to obtain succinct non-interactive arguments that are secure in the plain model under standard cryptographic assumptions. Unfortunately, the results that have been obtained on this topic have so far been negative [21], [126]. For example, Bartusek *et al.* [21] show, roughly speaking, that obtaining non-interactive arguments in this manner would require exploiting some special structure in both the underlying interactive argument and in the concrete hash function used to implement the random oracle in the Fiat-Shamir transformation.

5.4 Exercises

Exercise 5.1. Section 5.2 described the Fiat-Shamir transformation and asserted that if the Fiat-Shamir transformation is applied to any IP with negligible soundness error that satisfies an additional property called round-by-round soundness, then the resulting argument system is computationally sound in the random oracle model. One may wonder if in fact the Fiat-Shamir transformation yields a computationally sound argument for any IP with negligible soundness error, not just those that are round-by-round sound. In this problem, we will see that the answer is no.

Take any IP with perfect completeness and soundness error 1/3, and sequentially repeat it n times, having the verifier accept if and only if all n invocations of the base IP lead to acceptance. This yields an IP with soundness error 3^{-n}. Explain why applying the Fiat-Shamir transformation to this IP does *not* yield a sound argument system in the random oracle model, despite the fact that the soundness error 3^{-n} is negligible.

You may assume that the prover in the IP is permitted to pad messages with nonces if it so desires, i.e., the prover may append extra symbols to any message and the verifier will simply ignore those symbols. For example, if the prover in the IP wishes to send message $m \in \{0,1\}^b$, the prover could choose to send (m, m') for an arbitrary string m', and the IP verifier will simply ignore m'.

Exercise 5.2. Recall that the GKR protocol for circuit evaluation is used to verify the claim that $\mathcal{C}(x) = y$, where $\mathcal{C}$ is an arithmetic circuit over field $\mathbb{F}$, x is a vector in $\mathbb{F}^n$, and $\mathcal{C}$, x, and y are all known to both the prover and the verifier. Consider applying the Fiat-Shamir transformation in the random oracle model to the GKR protocol, but suppose that when applying the Fiat-Shamir transformation, the input x is *not* included in the partial transcripts fed into the random oracle. Show that the resulting non-interactive argument is *not* adaptively sound. That is, for a circuit $\mathcal{C}$ and claimed output y of your choosing, explain how a cheating prover can, in time proportional to the size of $\mathcal{C}$ and with overwhelming probability, find an input $x \in \mathbb{F}^n$ such that $\mathcal{C}(x) \neq y$, along with a convincing "proof" for the claim that $\mathcal{C}(x)$ in fact equals y.