The Fiat-Shamir Transformation

CS6230: Topics in Information Security
Lecture 11 (Oct 26, 2021)
Lecture and notes: Prashant Nalini Vasudevan

In the previous lecture, we saw a way to construct interactive arguments using PCPs[1]. Today, we will see how such arguments, in certain cases, can be made *non-interactive.* A non-interactive proof or argument has several advantages over an interactive one – for one, it can be written down at one point in time and then verified at a much later date without needing the prover to be present. Also, interaction can often be an expensive resource – multiple rounds of interaction over the internet, for instance, can add non-trivial latency – and fewer rounds of interaction in a proof is always welcome.

Of course, if such non-interactive proofs are required to be perfectly complete and sound, then they default to classical mathematical proofs, and we lose a lot of the power we had when working with interactive proofs. The non-interactive arguments we will be considering will have weaker soundness requirements, and also make use of randomness or some cryptography in its place.

1 The Fiat-Shamir Transformation

Much earlier in the class, we saw a simple procedure for reducing the number of rounds of interaction in a public-coin interactive proof. This transformation, however, lost any efficiency guarantees the prover may have had, and is not meaningful in practice. A relatively simple transformation, first proposed by Amos Fiat and Adi Shamir [FS86], performs a similar reduction in the number of rounds while keeping the prover's efficiency more-or-less the same. It compromises on the soundness guarantee – even if you start with a proof, you would only end up with an argument – but it also works if you start with an argument.

Given its simplicity and the fact that that arguments are usually the appropriate kind of interactive proof in practice, the Fiat-Shamir transformation is widely used, often implicitly, in many systems. We will see one such example in the next lecture. Today, we will see the transformation itself, and prove that it works in an idealised model.

For simplicity, suppose we start with a 3-message public-coin argument system (P, V) for a language L with negligible soundness error. That is, given input $x \in \{0,1\}^n$,

1. P sends the first message $\alpha \leftarrow P(x)$, contained in $\{0,1\}^a$
2. V sends a response $\beta \leftarrow \{0,1\}^b$
3. P sends the final message $\gamma \leftarrow P(x, \alpha, \beta)$, contained in $\{0,1\}^c$
4. V decides to accept or reject according to an algorithm $V(x, \alpha, \beta, \gamma)$

The Fiat-Shamir tranformation will tranform this argument into an argument (P_{FS}, V_{FS}) where the prover sends a single message π, and the verifier decides to accept or reject based only on this. Of course, if we asked for soundness against computationally unbounded provers in such a proof system, this would simply be an MA proof. We are interested in arguments that are more efficient than generic MA proofs, and this will be enabled by our weakening of the soundness requirement to only hold against efficient provers, and the involvement of cryptography.

The idea behind the transformation is to replace the verifier's uniformly random message β with something the prover can compute on its own, without needing the verifier to send it. Of course, the prover cannot be allowed to pick β arbitrarily, as it may then choose a β that always allows it to make the verifier accept. Instead, we will have the prover compute it as a function of the transcript of the protocol so far. Consider a function $H : \{0,1\}^n \times \{0,1\}^a \to \{0,1\}^b$, whose properties we will describe later. The transformed protocol (P_{FS}, V_{FS}) works as follows on input x:

1. P_{FS} computes $\alpha \leftarrow P(x)$
2. P_{FS} computes $\beta \leftarrow H(x, \alpha)$
3. P_{FS} computes $\gamma \leftarrow P(x, \alpha, \beta)$

[1] In reality, half of this lecture was the conclusion of Kilian's construction of arguments from PCPs, but I have left that in the previous lecture's notes for continuity.

4. P_{FS} sends (α, γ) to V_{FS}
5. V_{FS} computes $\beta \leftarrow H(x, \alpha)$, and accepts iff $V(x, \alpha, \beta, \gamma)$ accepts

If the original protocol is perfectly complete, then it is easy to see that this is as well, irrespective of the choice of H. For soundness, however, the choice of H matters. One desirable property for H here is that, if $x \notin L$, then it should be computationally hard for a cheating prover P^*_{FS} to find an $\alpha \in \{0,1\}^a$ such that when β is computed as $H(x, \alpha)$, there exists some $\gamma \in \{0,1\}^c$ such that $V(x, \alpha, \beta, \gamma)$ accepts. Such α's may exist, but we want them to be hard to find. More generally, this property of a function of it being hard to find an input-output pair satisfying a given relation is called *correlation intractability.* So if we instantiate the transformation above with an H that is correlation intractable in the manner described above, we would end up with a non-interactive argument (P_{FS}, V_{FS}) that is sound.

In practice, a cryptographic hash function like SHA-256 is used in place of H. It is generally believed that this instantiation is secure due to the difficulty of finding any non-trivial structure in SHA-256, but as it is a single fixed function, such an instantiation is not amenable to security proofs under the kinds of definitions we have set up so far. Instead, we will show that the transformed protocol is sound if instantiated with an idealised function H that is not practically implementable, but captures some strong security properties.

1.1 The Random Oracle Model

The Random Oracle Model [BR93] is an abstract setting used to study the security of protocols that make use of strong cryptographic functions (in various senses of this term). In this model, this function is replaced by a random oracle – an oracle that computes a randomly selected function with pre-specified input and output lengths. The idea is that if the function indeed has sufficiently strong cryptographic properties – usually pseudorandomness or collision-resistance – then it presents no discernible "structure" to a computationally efficient observer. This being the case, to this computationally efficient observer, the function looks effectively like a randomly chosen function, which indeed lacks any structure whatsoever. Thus, if we can argue that a protocol is secure in this model, this should give us some confidence that, when a good enough function H is used in place of the oracle, it will remain secure.

In our case, the random oracle is of the form $O : \{0,1\}^n \times \{0,1\}^a \to \{0,1\}^b$. Both parties P_{FS} and V_{FS} have access to this oracle, and the protocol (P^O_{FS}, V^O_{FS}) after the Fiat-Shamir transformation works as follows on input x:

1. P_{FS} computes $\alpha \leftarrow P(x)$
2. P_{FS} computes $\beta \leftarrow O(x, \alpha)$
3. P_{FS} computes $\gamma \leftarrow P(x, \alpha, \beta)$
4. P_{FS} sends (α, β, γ) to V_{FS}
5. V_{FS} checks that $\beta = O(x, \alpha)$, and accepts iff $V(x, \alpha, \beta, \gamma)$ accepts

As the oracle is now not just a single function, but rather a randomly selection function from the family of all functions, this protocol is now amenable to our usual approach of proving security, allowing us to state the following theorem.

Theorem 1.1. *If (P, V) is a public-coin argument for some language L with negligible soundness error, then (P^O_{FS}, V^O_{FS}) is an argument for L with negligible soundness error in the random oracle model.*

The intuition behind why this theorem should hold is as follows. Fix some input $x \notin L$. For any potential prover message $\alpha \in \{0,1\}^a$, we say that a $\beta \in \{0,1\}^b$ is *bad* for α if there exists a $\gamma \in \{0,1\}^c$ such that $V(x, \alpha, \beta, \gamma)$ accepts. As noted earlier, one sufficient condition for the protocol (P^O_{FS}, V^O_{FS}) to be sound is that it be hard for any efficient prover to find an α such that $O(x, \alpha)$ is bad for α. The soundness of the underlying proof system (P, V) implies that for any $\alpha \in \{0,1\}^a$ the fraction of $\beta \in \{0,1\}^b$ that are bad for it is negligible. Then, as $O(x, \alpha)$ is uniformly random, the probability that it is bad for α is also negligible. As a cheating prover has no other knowledge of the random oracle O apart from what it can learn by querying it on various inputs, in order to find an α such that $O(x, \alpha)$ is bad for it, it should have to make a super-polynomial number of queries. In other words, no polynomial-time prover should be able to find an α such that it can produce a proof that makes V^O_{FS} accept.

Proving this formally requires a little more work, as the prover's queries could be arbitrarily correlated with other queries and responses. None of that turns out to matter, however, since O is a random oracle.

Proof Sketch of Theorem 1.1. Suppose there is a (w.l.o.g.) deterministic polynomial-time cheating prover P_{FS}^{*O} that makes V_{FS}^{O} accept with non-negligible probability (over the randomness in O) on some input $x \notin L$. Then, we will show how to construct a cheating prover P^* that makes V accept with non-negligible probability in (P, V) on the same input.

To start with, suppose P_{FS}^{*O} acts in the following "semi-honest" manner: it comes up with an $\alpha \in \{0,1\}^a$ somehow, makes exactly one query to the oracle to get $\beta \leftarrow O(x, \alpha)$, comes up with $\gamma \in \{0,1\}^c$ somehow, and outputs (α, β, γ) that makes V_{FS}^{O} accept with non-negligible probability. Then, the cheating prover P^* is simply works as follows:

1. Get α from P_{FS}^*, and send it to V
2. Receive β from V, and give it as response ot P_{FS}^*'s one oracle query
3. Receive γ from P_{FS}^* and send it to V

It is easy to see that the probability that V then accepts is the same as the probability that V_{FS} accepts when interacting with P_{FS}^*. In general, however, the behaviour of P_{FS}^* need not follow this ideal pattern. It could make many correlated queries to the oracle, and not even write down an α until the very end of its execution.

It turns out, however, that a natural modification to this strategy works even in the general case – if P_{FS}^* makes T queries, P^* picks an $i \leftarrow [T]$ at random, and acts as above assuming the i^{th} query is what was used to compute the β that is eventually output by P_{FS}^*. That is, P^* works roughly as follows:

1. Pick $i \leftarrow [T]$, and wait till P^* makes its i^{th} oracle query, answering all the other queries uniformly at random
2. Suppose the i^{th} query is (x, α). Send α to V, receive its message β, and give β to P_{FS}^* as the response to this query
3. Complete the execution of P_{FS}^*, answering all remaining oracle queries uniformly at random, until it produces output $(\alpha', \beta', \gamma')$
4. If $\alpha' \neq \alpha$, or $\beta' \neq \beta$, send $\perp$ to V
5. Else, send γ' to V

Without loss of generality, P_{FS}^* does make the query (x, α') at some point. If P^* manages to guess this query correctly, then the probability that V accepts is the same as that of V_{FS}^* accepting. As P_{FS}^* makes at most T queries, P^* guesses correctly with probability at least $1/T$, which is only polynomially small. Thus, if V_{FS}^* accepts with non-negligible probability when interacting with P_{FS}^*, then V also accepts with non-negligible probability when interacting with P^*. So if (P, V) has negligible soundness error, then so does (P_{FS}, V_{FS}) in the random oracle model. □

1.2 Protocols with more rounds

The Fiat-Shamir transformation extends naturally to public-coin protocols with larger number of rounds. The verifier's message in each round is simply derived as the hash of all of the messages sent by the prover so far. The above proof can be extended to work for any constant number of rounds, though with slightly more work requiring some clever Markov bounds. Further, it turns out that the Fiat-Shamir transformation also preserves stronger notions of soundness (namely, knowledge soundness, which we will see in the next lecture). For details about the techniques in these more general proofs, see the original proof by Pointcheval and Stern [PS96].

References

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer* *and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.