Week 4

Nadav Kohen

July 16, 2025

Last week we studied all of the prerequisites to prove that the Schnorr digital signature is secure in the Random Oracle Model (ROM). We will now study a proof of the security of the Schnorr digital signature, where we will first introduce the Schnorr identification protocol, which we will show is secure against impersonations if the Discrete Log Problem (DL) is hard. Then we will use the Fiat-Shamir transform to show that Schnorr signatures are secure. Next, we will continue our study of more advanced topics by giving another proof that Schnorr signatures are secure in the Algebraic Group Model (AGM). Finally, we will begin our survey of a class of Zero Knowledge Proofs that generalize the Schnorr signature known as Sigma protocols.

In the following reading, I personally think their introduction of (I, st) in the definition of an identification protocol is a little bit confusing. I encourage you to think of the initial message, I, as just some random "public key", whose "private key", st, will be used in the prover's final response to ensure that they are not leaking their usual private key. Additionally, I think they could mention sooner than they do that step 1 of Algorithm $\mathcal{A}$ in the proof of Theorem 13.10 is literally just guessing which of the queries that $\mathcal{A}'$ makes to the random oracle contains the message that $\mathcal{A}'$ is trying to create a forgery for, and acting accordingly after making this guess. This makes it very unlikely ($\frac{1}{q}$ chance the guess was correct) that Algorithm $\mathcal{A}$ succeeds, even if $\mathcal{A}'$ *always* succeeds. However, because there is a polynomial

bound on the number of oracle queries $\mathcal{A}'$ makes, even if $\mathcal{A}$ is worse than $\mathcal{A}'$ we still get that $\mathcal{A}$ has a non-negligible success probability if $\mathcal{A}'$ does ($\frac{1}{q}$ is non-negligible).

Reading 1. *Read about the Fiat-Shamir Transform in:*
Jonathan Katz and Yehuda Lindell - Introduction to Modern Cryptography - Section 13.5.1.

Finally, we are ready to read a proof of the security of the Schnorr signature in the ROM! The final part of the proof where they discuss probabilities is a little bit tricky, but try to get at least a high-level understanding and we can discuss this in detail together or on the Discourse.

Reading 2. *Read about the Schnorr signature's security in:*
Jonathan Katz and Yehuda Lindell - Introduction to Modern Cryptography - Section 13.5.2.

Now let's work towards another proof! In 2018, Fuchsbauer, Kiltz, and Loss formalized the Algebraic Group Model (AGM). In the AGM, adversaries have the restriction that every time an "algebraic adversary" outputs a group element it must also output a list of exponents that show how to compute the group element as a combination of the group elements that have been given to the adversary as inputs (or public parameters). For example, recall the Computational Diffie-Hellman (CDH) assumption that it is hard to compute g^{xy} given $X = g^x$ and $Y = g^y$ as input. An algebraic adversary (i.e., adversary in the AGM) that plays the CDH attack game is given X and Y (as well as the generator g) and it must output not only a group element Z, but also a list (a, b, c) such that $Z = g^a X^b Y^c$.

Requiring adversaries to know how to construct new group elements out of previous elements rules out so-called "non-algebraic adversaries" that somehow attack by performing non-group operations on group elements. Thus, a proof of security in the AGM shows that it is not possible to succeed in an attack using an algebraic program. One benefit of the AGM is that proofs in the AGM usually avoid rewinding and forking arguments such as we saw in the proof of the Schnorr signature's security. However, since we only consider

algebraic adversaries, proofs in the AGM are weaker than proofs that do not use the AGM, since they only show that there are no attacks from adversaries that handle the group and its elements algebraically and there may yet exist non-algebraic adversaries with successful attacks. Another benefit of the AGM is that many standard assumptions are equivalent to the DL assumption in the AGM. For example,

Exercise 1. *Use proof by reduction to show that in the AGM, if the DL assumption holds for a group, then so does the CDH assumption. Recall that an adversary that succeeds in the CDH attack game in the AGM also provides a list of exponents!*

Let us now construct an EUF-CMA security proof for Schnorr signatures in the AGM (and also still using the ROM to model our hash function as a random oracle).

Exercise 2. *Use proof by reduction in the AGM to show that if the DL assumption holds, then the Schnorr signature is EUF-CMA secure.*

Hint: Recall that our goal is to use a program $\mathcal{A}'$ that plays the EUF-CMA attack game, from Reading 2 of last week, in the ROM to construct a program $\mathcal{A}$, which takes a group element, pk, as input and attempts to compute a value sk such that $g^{sk} = pk$. Recall that $\mathcal{A}'$ takes a public key as input, then it makes some number, q, of signature queries for messages $m_1, \ldots, m_q$ (whose responses are of the form (R_i, s_i)), then it makes some number (say q again, letting q be the larger of the two numbers) of hash queries for hash preimages $(\tilde{R}_1, m_1), \ldots, (\tilde{R}_q, m_q)$, before it finally outputs a forgery attempt (m, R, s). Note that since the $\tilde{R}_j$ are group elements, the adversary must also output a list $(a_0, a_1, \ldots, a_{q+1})$ such that $\tilde{R}_j = g^{a_0} pk^{a_1} \prod_{i=1}^{q} R_i^{a_{i+1}}$ because $\mathcal{A}'$ has received pk as input, g is a public parameter, and it has also received the R_i as input in return to its signature queries. Thus, in order to use $\mathcal{A}'$ we must respond to its signature queires by using the simulation trick for Schnorr Identification transcripts from last week, then we can respond to hash queries with random values (unless one of the $\tilde{R}_i$ is a value we've seen before, then we have to be more careful). Finally, given

the value R in the forgery, we find the list $(a_0, \ldots, a_{q+1})$ that $\mathcal{A}'$ used to represent R when it had made its hash oracle query and our goal is to compute a value sk such that $pk = g^{sk}$ (you must use the fact that the forgery satisfies verification, and that you know values k_i such that $R_i = g^{k_i}$).

Notice that in the AGM, we did not need to use any kind of rewinding argument (that succeeding to make a forgery once with non-negligible probability implies that we can succeed twice with non-negligible probability). In particular, we have shown that being able to produce a single forgery as an algebraic adversary is enough to efficiently compute the secret key for which the forgery was produced.

We will now shift our focus back to the ROM without the AGM to study a generalization of the Schnorr identification protocol.

Reading 3. *Using the Schnorr identification protocol as a mental model, read about the definition of a Sigma protocol in:*
Dan Boneh and Victor Shoup - A Graduate Course in Applied Cryptography - Section 19.4.

Note that one reason we are willing to assume that our verifier is honest, is that we will eventually be applying the Fiat-Shamir transform to replace them with a hash function, which will be honest. Let's quickly exercise these new definitions,

Exercise 3. *Many applications of Sigma protocols require a large challenge space. This exercise shows that we can always take a Sigma protocol with a small challenge space and turn it into one with a large challenge space, essentially by parallel repetition.*
Let (P, V) be a Sigma protocol for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$, with challenge space $\mathcal{C}$. Let k be a positive integer. Define a new Sigma protocol (P^k, V^k) as follows. Here, the prover P^k takes as input $(x, y) \in \mathcal{R}$, the verifier V^k takes as input $y \in \mathcal{Y}$, and the challenge space is $\mathcal{C}^k$.

- *P^k initializes k instances of P on input (x, y), obtaining commitments $t_1, \ldots, t_k$, and sends these to V^k.*

- *V^k chooses $(c_1, \ldots, c_k) \in \mathcal{C}^k$ at random, and sends this to P^k.*
- *For $i = 1, \ldots, k$ the prover P^k feeds c_i into the ith instance of P, obtaining a response z_i. It sends $(z_1, \ldots, z_k)$ to V^k.*
- *For $i = 1, \ldots, k$ the verifier V^k verifies that (t_i, c_i, z_i) is an accepting conversation for y.*

(a) Show that (P^k, V^k) is a Sigma protocol for $\mathcal{R}$.

(b) Show that if (P, V) provides knowledge soundness, then so does (P^k, V^k).

(c) Show that if (P, V) is special HVZK, then so is (P^k, V^k).

Next week, we will see some more examples of concrete Sigma protocols, and then we will generalize the Fiat-Shamir transform to Sigma protocols by replacing our verifiers with hash functions modeled as random oracles.