Schnorr Identification and Signatures

David Mandell Freeman

October 20, 2011

1 Identification

An *identification scheme* is an interactive protocol between two parties, a *prover* $\mathcal{P}$ and a *verifier* $\mathcal{V}$. If the protocol is successful, then at the end of the protocol the verifier is convinced he is interacting with the prover, or more precisely, with someone who knows the secret key that corresponds to the prover's public key.

A simple example is the standard protocol of password authentication. The prover's secret key is her password pw, and the public key is $H(pw)$ where H is a "one-way" hash function. The protocol consists of the prover sending $H(pw)$, and the verifier checks that this matches the stored value. While this is secure against "direct" attacks, it is not secure against "eavesdropping" attacks, where the adversary can observe some interactions between $\mathcal{P}$ and $\mathcal{V}$ and then try to impersonate $\mathcal{P}$ — as soon as the adversary sees $H(pw)$ from one interaction, he can then impersonate $\mathcal{P}$ in all further interactions.

In the Schnorr identification protocol using elliptic curves, the prover's secret is the discrete logarithm a of a pair $(P, Q = [a]P)$ where P and Q are points on an elliptic curve $E(\mathbb{F}_q)$. (The system works in any finite abelian group where discrete logarithm is hard, but we use elliptic curves for concreteness and since that's what this course is about!) To protect against eavesdropping attacks, the protocol has three rounds:

Schnorr Identification Protocol. The public key pk consists of an elliptic curve $E(\mathbb{F}_q)$ and two points $P, Q \in E(\mathbb{F}_q)$ of order r. The secret key sk is an integer in $[1, r]$ such that $Q = [a]P$.

1. $\mathcal{P}$ chooses a random $k \overset{\text{R}}{\leftarrow} [1, r]$ and sends $R = [k]P$ to $\mathcal{V}$.
2. $\mathcal{V}$ chooses a random "challenge" $e \overset{\text{R}}{\leftarrow} [1, r]$ and sends e to $\mathcal{P}$.
3. $\mathcal{P}$ computes $s = k + ae \pmod r$ and sends s to $\mathcal{V}$.

$\mathcal{V}$ accepts if $[s]P = R + [e]Q$.

The function of the initial random value k is to "blind" the secret a so that it can be reused in subsequent executions of the protocol.

Theorem 1. *If the discrete logarithm assumption holds in $E(\mathbb{F}_p)$, then the Schnorr identification protocol is secure against direct impersonation attacks.*

Specifically, suppose there is an efficient algorithm $\mathcal{A}$ that can interact with $\mathcal{V}$ such that $\mathcal{V}$ accepts with probability at least ϵ. Then there is an efficient algorithm $\mathcal{B}$ that can solve the discrete logarithm problem in $E(\mathbb{F}_q)$ with probability at least $\epsilon^2 - \epsilon/r$.

Proof sketch. The idea (due to Pointcheval and Stern) is to use a "rewinding technique": we fix the random value k used by the impersonation algorithm $\mathcal{A}$ and have $\mathcal{A}$ interact twice with the verifier $\mathcal{V}$, where each time $\mathcal{V}$ chooses a different random e. Suppose $\mathcal{V}$ chooses e_1 and e_2 for the two different executions, and $\mathcal{A}$ computes s_1 and s_2 in the two executions. If $\mathcal{V}$ accepts both times, then $s_i = k + ae_i$ for $i = 1, 2$, and then we have $a = (s_1 - s_2)/(e_1 - e_2) \pmod r$ as long as $e_1 \neq e_2$.

The "rewinding lemma" tells us that the probability that we obtain a discrete log solution $\epsilon(\epsilon - 1/r)$. The ϵ factor is the accept probability of $\mathcal{V}$ on the first execution, while the $\epsilon - 1/r$ factor is (at least) the probability that $\mathcal{V}$ accepts on the second execution and $e_2 \neq e_1$. □

In an *eavesdropping attack* the adversary gets to observe transcripts of interactions between $\mathcal{P}$ and $\mathcal{V}$ before he mounts his impersonation attempt. The Schnorr identification protocol is also secure against such attacks. The idea behind the proof is that the adversary cannot learn anything from observing transcripts of interactions that he could not compute himself; formally, we show that there exists a simulator $\mathcal{S}$ that given only pk can output transcripts that are distributed identically to real transcripts. The simulator $\mathcal{S}$ works as follows:

1. Choose random $e, s \xleftarrow{\text{R}} [1, r]$.
2. Compute $R = [s]P - e[Q]$.
3. Output the transcript (R, e, s).

The key idea is that the order in which transcript elements are generated doesn't matter when observing the transcript as a whole. (This is different from an "active attack" where the adversary gets to interact with the prover, and thus must generate messages in the correct order.)

2 Signatures

We can create a signature scheme out of the Schnorr identification protocol using the "Fiat-Shamir transform." The idea is that the prover becomes the signer, and instead of e being a random challenge from the verifier, e encodes the message to be signed. To use the security properties of the underlying identification scheme, the encoded message must look random to everyone involved. This is done using a hash function H that takes arbitrary bit strings and outputs integers in $[1, r]$; if the hash function is well-designed, then the output looks random.

Schnorr Signature Scheme. The scheme $\mathcal{S}$ consists of three algorithms, Gen, Sign, and Verify.

- $\mathsf{Gen}()$ does the following:
  1. Choose an elliptic curve E over a finite field $\mathbb{F}_q$.
  2. Choose a random point $P \xleftarrow{\text{R}} E(\mathbb{F}_q)$
  3. Choose a random integer $a \xleftarrow{\text{R}} [1, r]$ where r is the order of P.
  4. Choose a hash function $H \colon \{0, 1\}^* \to [1, r]$.
  5. Output $\mathsf{pk} = (P, Q = [a]P)$ and $\mathsf{sk} = (a, \mathsf{pk})$.
- $\mathsf{Sign}(\mathsf{sk}, M)$ does the following:

1. Choose random $k \overset{\mathrm{R}}{\leftarrow} [1, r]$ and set $R = [k]P$.
2. Set $e = H(M \| R)$.
3. Set $s = k + ae \pmod{r}$.
4. Output the signature $\sigma = (R, s)$.

- $\mathsf{Verify}(\mathsf{pk}, \sigma = (R, s))$ does the following:

  1. Compute $e = H(M \| R)$.
  2. If $R + [e]Q = [s]P$, output "accept"; else output "reject".

Note that in real life, Gen will ensure that r is a large prime, equal to or very close to $\#E(\mathbb{F}_q)$.

To analyze the security of Schnorr signatures, we model the hash function as a "random oracle." This means that the adversary attacking the signature scheme is not given a description of H but rather is allowed to query an "oracle" that returns the value of H on the specified input. When queried on input x, the oracle for H is allowed to return any value for $H(x)$ as long as (a) our answers are consistent (i.e., we always return the same output on the same input), and (b) answers to previously unseen queries are uniformly random and independent of anything the adversary has previously seen.

We now show that an adversary that can break the Schnorr signature scheme (in the random oracle model) can break the identification scheme, and thus solve discrete logarithms.

Theorem 2. *If the Schnorr identification protocol is secure, then the Schnorr signature scheme is secure in the random oracle model.*

Specifically, suppose there is an efficient adversary $\mathcal{A}$ that produces a valid signature forgery with probability ϵ while making at most q_H hash function queries and q_S signature queries. Then there is an algorithm $\mathcal{B}$ that successfully impersonates the prover in the Schnorr identification protocol with probability at least $\epsilon/(q_H + 1) - (q_H + q_S + 1)/r$.

Proof sketch. Suppose we are given an public key $P, Q = [a]P$ for the Schnorr identification scheme and an adversary $\mathcal{A}$ that breaks the signature scheme. We construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to attack the ID scheme. Adversary $\mathcal{B}$ will act as a challenger in the signature game, responding to $\mathcal{A}$'s signature and hash function queries. $\mathcal{B}$ will then use the information it receives from $\mathcal{A}$ to interact with the verifier $\mathcal{V}$ in the ID protocol.

Before describing $\mathcal{B}$ we make a simplifying assumption. By modifying $\mathcal{A}$ to make one more hash function query if necessary, we assume that if $\mathcal{A}$'s forgery is (m^*, R^*, s^*) then $m^* \| R^*$ was queried to the hash function at some point. (This is where the $q_H + 1$ comes from in the probability statement.)

We now describe $\mathcal{B}$. First suppose that $\mathcal{A}$ makes only the hash query $m^* \| R^*$ and no signature queries. When $\mathcal{A}$ makes this query, $\mathcal{B}$ will use R^* as the first message in the ID protocol. $\mathcal{B}$ then obtains a challenge e^* from the verifier $\mathcal{V}$ and returns this value for $H(m^*, R^*)$. The s^* component of the adversary's forgery is then sent as the final message in the ID protocol. If the forgery is valid, then we have $[s^*]P = R^* + [e^*]Q$, and therefore the verifier accepts and we have broken the ID scheme.

Now we allow $\mathcal{A}$ to make signature queries on messages m_i. We simulate a signature in the same way we simulated transcripts for the ID scheme above: we choose random e_i and s_i and set

$R_i = [s_i]P - [e_i]Q$. We then set the value of the hash function on input $m_i \| R_i$ to be $H(m_i \| R_i) = e_i$. Since s_i and e_i are random, this value looks random to the adversary, as required.

Now we allow $\mathcal{A}$ to make more hash queries $\tilde{m}_j \| \tilde{R}_j$. For now assume that $m^* \| R^*$ is the first hash query. For other hash queries, we either use the preprogrammed value (for example, if $\tilde{m}_j \| \tilde{R}_j = m_i \| R_i$ for the ith signature query), or we choose a new random value $\tilde{e}_j$. From the point of view of the adversary, these responses are consistent with H being a random function.

Finally, since the hash query used by $\mathcal{A}$ in his forgery may not be the first one, we use a "guessing" argument: namely, we pick in advance a random $\omega \in [1, q_H + 1]$ and hope that $\mathcal{A}$ will forge on the ωth hash query. Since our chances of being correct are $1/(q_H + 1)$, our success probability is reduced by this factor.

In detail, adversary $\mathcal{B}$ is given a public key (P, Q) for the Schnorr identification protocol and works as follows:

- Initialization:
  1. Choose a random $\omega \overset{\mathrm{R}}{\leftarrow} [1, q_H + 1]$.
  2. Send pk to $\mathcal{A}$ as the signature public key.
- On receiving the ith signature query m_i, do the following:
  1. Choose random $e_i, s_i \overset{\mathrm{R}}{\leftarrow} [1, r]$.
  2. Set $R_i = [s_i]P - [e_i]Q$.
  3. Return $\sigma_i = (R_i, s_i)$.
  4. Store e_i as the value of $H(m_i \| R_i)$.
- On receiving the jth hash query $\tilde{m}_j \| \tilde{R}_j$, do the following:
  1. If the value of $H(\tilde{m}_j \| \tilde{R}_j)$ has already been set (during a signature query or a previous hash query), return that value.
  2. If $j \neq \omega$, choose a random $\tilde{e}_j \overset{\mathrm{R}}{\leftarrow} [1, r]$, and set $H(\tilde{m}_j \| \tilde{R}_j) = \tilde{e}_j$.
  3. If $j = \omega$:
     (a) Send $\tilde{R}_j$ to $\mathcal{V}$ as the first step in the ID protocol.
     (b) Obtain a challenge e^* from $\mathcal{V}$.
     (c) set $H(\tilde{m}_j \| \tilde{R}_j) = e^*$.
- On receiving a forgery attempt (m^*, R^*, s^*) from $\mathcal{A}$, send s^* to $\mathcal{V}$ as the final step in the ID protocol.

If the jth hash query is $m^* \| R^*$ (which happens with probability $1/(q_H + 1)$ and the adversary's forgery is valid (which happens with probability ϵ), then we have

$$[s^*]P = R^* + [H(m^* \| R^*)]Q = R^* + [e^*]Q$$

and therefore $\mathcal{V}$ accepts.

The term $(q_H + q_S + 1)/r$ in the probability statement comes from the chance that there will be a collision among the points R_i chosen in response to signing queries; we omit the details. □

Corollary 3. *If the discrete logarithm assumption holds in $E(\mathbb{F}_q)$, then the Schnorr signature scheme is secure.*