University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 41

1 An Improved Signature Scheme in the RO Model

Last time, we presented an efficient signature scheme which could be proven secure in the random oracle model. Signing a message m required hashing m to a random element of $\mathcal{D}_{\mathsf{k}}$ (using the random oracle) and then inverting the trapdoor permutation on that random element; thus, the signature on m is simply $f_{\mathsf{k}}^{-1}(H(m))$ (where H is the random oracle). The signature scheme is known as the "Full-Domain Hash" (or FDH) scheme.

We saw in the previous lecture that if the trapdoor permutation was (t, ϵ)-secure, then the FDH signature scheme constructed based on that permutation is $(t, q_h\epsilon)$-secure, where q_h represents the number of oracle queries made by an adversary. While this is progress since we at least have a measure of provable security, the result is not all that great. Since q_h corresponds to the number of times the adversary evaluates the hash function H, since evaluating H is typically very efficient, and since evaluations of H can be done by the adversary off-line (and without the signer's knowledge), q_h might well be very large. A dedicated adversary might well be able to have $q_h \approx 2^{60}$. In this case, using even a very secure trapdoor permutation with $\epsilon \approx 2^{-60}$ would result in a not-very-secure signature scheme (since $2^{60}\epsilon \approx 1$!). Of course, we can simply use a trapdoor permutation with lower ϵ, but this may lead to a less efficient scheme.[1]

Here, we show that not all is lost. For the particular case when the trapdoor permutation used is the RSA permutation, a better proof of security is possible. We first state the theorem, then briefly discuss the implications, and finally give a proof.

Theorem 1 *Assume that RSA is a (t, ϵ)-secure trapdoor permutation. Then the FDH signature scheme instantiated with RSA is $(t, eq_s\epsilon)$-secure, where q_s is the number of signatures the adversary requests from the signer (and $e \approx 2.7$ is the base of the natural logarithms).*

Thus, an adversary attacking FDH based on RSA has probability of forgery $q_s\epsilon$ rather than $q_h\epsilon$ as would be expected from the proof of security for the case of general trapdoor permutations. In practice, $q_s \ll q_h$; to see why, notice that computing signatures takes longer and more importantly must be done by the signer. It is much more difficult for an adversary to get a signer to sign 1000 messages of the adversary's choice than for the adversary to evaluate a hash function 1000 times. So, Theorem 1 indicates that for practical purposes using RSA with $\epsilon \approx 2^{-60}$ is perfectly fine.

Proof We give a high level overview of the proof before presenting the details. As usual, we will take an adversary A attacking the signature scheme and use this to construct an

[1] As an example, inverting RSA for 1024-bit moduli might correspond to $\epsilon \approx 2^{-60}$. But obtaining $\epsilon \approx 2^{-90}$ might require using RSA with 2048-bit moduli, which would be less efficient.

adversary A' which inverts RSA. For the proof of the previous lecture (for the case of a general trapdoor permutation), we can describe the strategy of A' as follows: let q_h denote the number of hash queries made by A. Pick a random index $i \in \{1, \ldots, q_h\}$ and set the output of H in such a way that (1) A' can answer signature queries corresponding to every query to H *except* the i^{th} query and (2) if A forges a signature corresponding to the i^{th} query to H, then A' computes the desired inverse. Since i is chosen at random (and since A cannot ask for signatures on messages corresponding to *all* queries to H), the probability that A outputs a forgery at the desired point in at least $1/q_h$.

We could improve the probability that A outputs a forgery for a message that helps A' if we allow A' to choose multiple indices in $\{1, \ldots, q_h\}$ at which to "embed" the value that it wants to invert. But in general this is not possible: for example, if A' sets y as the output of H on more than one input then H no longer acts as a random oracle (in particular, A should see collisions in H will negligible probability). But for the case of RSA we *can* embed our instance in more than one place and thereby increase our chances of success. We give the details now.

Again, we are given algorithm A which forges signatures for FDH instantiated with RSA with some probability δ. We use A to construct an algorithm A' which tries to invert a given RSA instance (i.e., given N, e, y, tries to compute x such that $x^e = y \bmod N$).

$A'(N, e, y)$
Set $PK = (N, e)$; run $A(PK)$
When A asks for $H(m_i)$, answer as follows:
with probability α:
pick $r_i \leftarrow \mathbb{Z}_N^*$ and return $r_i^e \bmod N$
(call m_i of this sort *type 1*)
with probability $1 - \alpha$:
pick $r_i \leftarrow \mathbb{Z}_N^*$ and return $y \cdot r_i^e \bmod N$
(call m_i of this sort *type 2*)
When A asks for a signature on message m_i:
if m_i is type 1, return r_i
if m_i is type 2, abort
when A outputs forgery (m_i, σ):
if m_i is type 1, abort; otherwise, output σ / r_i

We may note the following: (1) as long as A' does not abort, the simulation it provides for A is perfect. In particular, the outputs of H are uniformly and independently distributed (for type 1 m_i, this is clear; for type 2 m_i it follows from the fact that $r_i^e \bmod N$ is random so multiplying by y still gives a random value). Furthermore, (2) if A' does not abort and if A outputs a valid forgery, then A' outputs the correct inverse of y. This is so since if A outputs a forgery it means that $\sigma^e = H(m_i) = y \cdot r_i^e \bmod N$ so that $(\sigma / r_i)^e = y \bmod N$.

All that remains is to determine the probability that A' does not abort. Each signature query of A can be answered by A' with probability exactly α (since A' can answer the query only if it corresponds to a type 1 message). When A outputs its forgery, this "helps" A' (and A' does not abort) with probability exactly $1 - \alpha$. Putting this together shows that the total probability that A' does not abort is $\alpha^{q_s}(1 - \alpha)$.

We now maximize this probability. Taking the derivative and setting equal to zero gives:

$q_s - (q_s+1)\alpha = 0$, or $\alpha = q_s/(q_s+1)$. Plugging this in shows that in this case the probability of not aborting is:

$$\left(\frac{q_s}{q_s+1}\right)^{q_s} \cdot \frac{1}{q_s+1} = \frac{1}{q_s}\left(1 - \frac{1}{q_s+1}\right)^{q_s+1} \approx \frac{e^{-1}}{q_s},$$

where this holds for reasonably large q_s (and e here is the base of the natural logarithm).

Putting everything together, we see that the probability that A' inverts the given RSA instance is (at least) $e^{-1}\delta/q_s$ (i.e., the probability that A' forges multiplied by the probability that A' does not abort). Since this can be at most ϵ we obtain $\delta \leq eq_s\epsilon$, completing the proof. ■