CMSC 858K — Advanced Topics in Cryptography March 2, 2004

Lecture 11

Lecturer: Jonathan Katz *Scribe(s): Rengarajan Aravamudhan, Nan Wang*

1 Review of the Cramer-Shoup Encryption Scheme

At the beginning of this lecture, we reviewed the proofs of security for both the Cramer-Shoup "lite" and the full Cramer-Shoup schemes. However, rather than repeating the proofs here, we instead refer the interested reader to the previous lecture notes.

2 NIZK Proof Systems

Previously in the course, we have seen the Naor-Yung and Dolev-Dwork-Naor encryption schemes, both of which rely on adaptively-secure non-interactive zero-knowledge (NIZK) proof systems. In the next few lectures, we will see how to construct such proof systems. Our discussion is drawn from the work of Feige-Lapidot-Shamir [7, 3, 2, 4] and is also based on [5, Section 4.10]. We define both "non-adaptive" NIZK and adaptively-secure NIZK. Although we need adaptively-secure NIZK for our applications, non-adaptive NIZK will be useful toward developing intuition for the problem.

Definition 1 ((Non-Adaptive) NIZK) A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is an *NIZK proof system* for a language $L \in \mathcal{NP}$ if, for some polynomials p_1, p_2:

- **(Completeness)** For all $x \in L \cap \{0,1\}^{\leq p_1(k)}$ and witness w for x,

$$\Pr[r \leftarrow \{0,1\}^{p_2(k)}; \pi \leftarrow \mathcal{P}(1^k, r, x, w) : \mathcal{V}(1^k, r, x, \pi) = 1] = 1.$$

  We say π is *valid proof for* x (assuming k, r are clear from context) if $\mathcal{V}(1^k, r, x, \pi) = 1$.

- **(Soundness)** For all (even unbounded) $\mathcal{P}^*$ and all $x \notin L$, the following is negligible:

$$\Pr[r \leftarrow \{0,1\}^{p_2(k)}; \pi \leftarrow \mathcal{P}^*(r, x) : \mathcal{V}(1^k, r, x, \pi) = 1].$$

- **(Zero-Knowledge)** There exists a PPT simulator Sim such that the following two ensembles are computationally indistinguishable for all PPT A:

$$\{(x, w) \leftarrow A(1^k); r \leftarrow \{0,1\}^{p_2(k)}; \pi \leftarrow \mathcal{P}(1^k, r, x, w) : (r, x, \pi)\}$$

$$\{(x, w) \leftarrow A(1^k); (r, \pi) \leftarrow \mathsf{Sim}(1^k, x) : (r, x, \pi)\}.$$

  (We require that $A(1^k)$ output (x, w) with $x \in L \cap \{0,1\}^{\leq p_1(k)}$ and w a witness for x.)

$\diamondsuit$

The following definition strengthens the previous one in two ways: first, soundness holds even when the (cheating) $\mathcal{P}^*$ chooses $x \notin L$ *after* seeing the random string r (i.e.,

adaptively). Second, the zero-knowledge property holds even when the simulator learns x *after* fixing its simulated random string r (in particular, the zero-knowledge property holds even if an efficient adversary chooses x after seeing the simulated random string).

Definition 2 (Adaptively-Secure NIZK) A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is an *adaptively-secure NIZK proof system* for a language $L \in \mathcal{NP}$ if, for some polynomials p_1, p_2:

- **(Completeness)** As before. Again, we say π is *a valid proof of* x (assuming k, r are clear from context) if $\mathcal{V}(1^k, r, x, \pi) = 1$.
- **(Adaptive Soundness)** For all (even unbounded) $\mathcal{P}^*$, the following is negligible:

$$\Pr[r \leftarrow \{0,1\}^{p_2(k)}; (x, \pi) \leftarrow \mathcal{P}^*(r) : \mathcal{V}(1^k, r, x, \pi) = 1 \wedge x \notin L].$$

- **(Adaptive Zero-Knowledge)** There exists a PPT simulator $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for all PPT adversaries A the following are computationally indistinguishable:

$$\{r \leftarrow \{0,1\}^{p_2(k)}; (x, w) \leftarrow A(1^k, r); \pi \leftarrow \mathcal{P}(1^k, r, x, w) : (r, x, \pi)\}$$

$$\{(r, \mathsf{state}) \leftarrow \mathsf{Sim}_1(1^k); (x, w) \leftarrow A(1^k, r); \pi \leftarrow \mathsf{Sim}_2(1^k, x, \mathsf{state}) : (r, x, \pi)\},$$

  where we require that $A(1^k, \cdot)$ output a pair (x, w) with $x \in L \cap \{0,1\}^{\leq p_1(k)}$ and w a witness for x.

$\diamondsuit$

We note that it is easy to transform any NIZK proof system into one achieving adaptive *soundness* as follows: Let $(\mathcal{P}, \mathcal{V})$ be an NIZK proof system satisfying "non-adaptive" soundness. Assume for simplicity that $p_1(k) = k$, and let $\mathsf{Bad}_k = \bar{L} \cap \{0,1\}^{\leq k}$. Soundness implies that for any fixed $x \in \mathsf{Bad}_k$, the probability over random choice of r that there exists a valid proof π for x is negligible, and in particular less than $1/2$. (We assume that $\mathcal{V}$ simply rejects if the statement x is longer then $p_1(k)$). Consider the modified protocol $(\mathcal{P}', \mathcal{V}')$ using a random string of length $2k \cdot p_2(k)$ (where $p_2(\cdot)$ is the length of the random string used by $(\mathcal{P}, \mathcal{V})$). The prover $\mathcal{P}'(1^k, r', x, w)$ parses the given random string r' as $2k$ strings $r_1, \ldots, r_{2k}$ and runs $\mathcal{P}(1^k, r_i, x, w)$ using each of these strings to generate proofs $\pi_1, \ldots, \pi_{2k}$. The verifier $\mathcal{V}'$ accepts only if all of these proofs are valid (with respect to $\mathcal{V}$).

It is not hard to see that completeness and (non-adaptive) zero-knowledge are unaffected by this transformation. We now bound the probability, over random choice of common random string r', that there exists an $x \in \mathsf{Bad}_k$ and a valid proof $\pi = \pi_1, \ldots, \pi_n$ for x. For any fixed $x \in \mathsf{Bad}_k$, a simple probability calculation shows that the probability, over r', that there exists a valid proof for x is at most 2^{-2k}. In other words, for any given $x \in \mathsf{Bad}_k$ at most a fraction 2^{-2k} strings r' are "bad" for this x. Then summing over all 2^{k+1} strings in Bad_k shows that at most a fraction 2^{-k+1} strings r' are "bad" for *some* $x \in \mathsf{Bad}_k$. In other words, the probability of such a "bad" r' is at most 2^{-k+1}, which is negligible.

We remark that it is unknown how to convert an arbitrary NIZK proof system into one achieving adaptive zero-knowledge. However, we will see a specific construction that works. Let us briefly outline the next few lectures:

- We first introduce the "hidden-bits" model and show that any NIZK proof system in this model can be transformed to an NIZK proof system in the real model (i.e., where a common reference string is available) assuming the existence of trapdoor permutations.
- We then show how to construct an NIZK proof system in the "hidden-bits" model. Coupled with the previous result, this yields a construction of an NIZK proof system in the real model.
- Finally, we note that the construction above actually achieves *adaptively-secure* NIZK without any further modification.

3 The Hidden-Bits Model

Informally, an NIZK proof system in the hidden-bits model proceeds as follows: the prover is initially given some sequence of bits which are hidden from the verifier. In the course of proving that $x \in L$, the prover can choose to reveal some arbitrary set of these bits to the verifier. The verifier never learns the bits of the string that are *not* revealed to it by the prover, and the prover cannot cheat and change the values in the string it is given. Formally, we imagine that the prover is given a string r of length n and sends to the verifier (along with other information) a set of indices $I \subseteq [n]$ (where $[n] = \{1, \ldots, n\}$). The verifier is then given the bits $\{r_i\}_{i \in I}$, which we denote by r_I. We stress that the hidden-bits model is not meant to be a realistic, but is instead only a conceptual model useful as a step toward our ultimate goal. The full definition follows.

Definition 3 (NIZK in the Hidden-Bits Model) A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is an *NIZK proof system in the hidden-bits model* if, for some polynomials p_1, p_2:

- **(Completeness)** For all $x \in L \cap \{0,1\}^{\leq p_1(k)}$ and witnesses w for x:

$$\Pr[r \leftarrow \{0,1\}^{p_2(k)}; (\pi, I) \leftarrow \mathcal{P}(1^k, r, x, w) : \mathcal{V}(1^k, r_I, x, \pi, I) = 1] = 1.$$

- **(Soundness)** For all (unbounded) $\mathcal{P}^*$, the following is negligible:

$$\Pr[r \leftarrow \{0,1\}^{p_2(k)}; (x, \pi, I) \leftarrow \mathcal{P}^*(r) : \mathcal{V}(1^k, r_I, x, \pi, I) = 1 \wedge x \notin L].$$

- **(Zero-Knowledge)** There exists a PPT simulator Sim such that the following are computationally indistinguishable for all PPT A:

$$\{(x, w) \leftarrow A(1^k); r \leftarrow \{0,1\}^{p_2(k)}; (\pi, I) \leftarrow \mathcal{P}(1^k, r, x, w) : (r_I, x, \pi, I)\};$$

$$\{(x, w) \leftarrow A(1^k); (r_I, \pi, I) \leftarrow \mathsf{Sim}(1^k, x) : (r_I, x, \pi, I)\}.$$

  (We require that $A(1^k)$ output (x, w) with $x \in L \cap \{0,1\}^{\leq p_1(k)}$ and w a witness for x.)

♢

We now show how to transform any NIZK proof system in the hidden-bits model to an NIZK proof system in the model where there is a common random string available to the players. The transformation is secure assuming the existence of any trapdoor permutation family. (See [6, Appendix C] for further details on necessary assumptions.)

Theorem 1 *Assuming the existence of trapdoor permutations and any NIZK proof system $(\mathcal{P}', \mathcal{V}')$ in the hidden-bits model, we may construct an NIZK proof system $(\mathcal{P}, \mathcal{V})$ in the common random string model.*

Proof We first show the construction, and then prove that the construction works as claimed. We use the following notation for our trapdoor permutation family[1]: algorithm $\mathsf{Gen}(1^k)$ is a randomized algorithm which outputs a pair of (efficiently computable) functions (f, f^{-1}) where f^{-1} is called the "trapdoor" for f. Furthermore, f is always a permutation over $\{0,1\}^k$, and $f^{-1}(f(x)) = x$ for all $x \in \{0,1\}^k$. We also assume that the set of "legal" f's (i.e., those that can possibly be output by Gen) is efficiently decidable.[2] Finally, we let h denote a hard-core bit for this trapdoor permutation family. Formally, this means that for all PPT algorithms A the following is neglible:

$$\left| \Pr\left[(f, f^{-1}) \leftarrow \mathsf{Gen}(1^k); x \leftarrow \{0,1\}^k; y = f(x) : A(1^k, f, y) = h(x) \right] - \frac{1}{2} \right|.$$

We also assume that $h(x)$, for randomly-chosen x, gives an perfectly unbiased bit (this is not essential, but it makes the proof slightly easier).

We make the following additional assumptions without loss of generality. First, we assume that the random string used by $\mathsf{Gen}(1^k)$ has length k, and hence the maximum number of different f's that can be output is 2^k. We also assume that the soundness error of $(\mathcal{P}', \mathcal{V}')$ (i.e., the probability that a cheating $\mathcal{P}^*$ succeeds in giving a valid proof for some $x \notin L$) is at most 2^{-2k}. Using the technique shown earlier in these notes, if $(\mathcal{P}', \mathcal{V}')$ does not satisfy this condition we may construct a new NIZK proof system (still in the hidden-bits model) that *does* satisfy this condition by running $2k$ copies of $(\mathcal{P}', \mathcal{V}')$ in parallel.

Let $n = p_2(k)$ refer to the length of the string r' given to $\mathcal{P}'$ in the hidden-bits model for security parameter k; we simply write n when k is clear from context. In the common random string model, we let the string r shared by $\mathcal{P}$ and $\mathcal{V}$ have length $k \cdot n$. Given a common string $r = r_1 | \cdots | r_n$, where each $r_i \in \{0,1\}^k$, the prover and verifier proceed as follows:

1. $\mathcal{P}(1^k, r, x, w)$ runs $\mathsf{Gen}(1^k)$ to obtain (f, f^{-1}). It then computes an n-bit string r' by setting $r'_i = h(f^{-1}(r_i))$ (here, r'_i simply denotes the i^{th} bit of r').
2. $\mathcal{P}$ then runs $\mathcal{P}'(1^k, r', x, w)$ to obtain π, I. Finally, $\mathcal{P}$ outputs f, π, I, and $\{f^{-1}(r_i)\}_{i \in I}$.
3. $\mathcal{V}(1^k, r, x, (f, \pi, I, \{z_i\}_{i \in I}))$ proceeds as follows: it first checks that f is valid (here is where we use the fact that the set of f's generated by Gen is efficiently decidable). For each $i \in I$, it checks that $f(z_i) = r_i$ (if any of these fail, then $\mathcal{V}$ outputs 0). Next, it sets $r'_i = h(z_i)$ for each $i \in I$. Finally, it outputs $\mathcal{V}'(1^k, r'_I, x, \pi, I)$.

The intuition is as follows: assume for a moment that the prover honestly generates (f, f^{-1}) at random, independent of r. Then the string r' constructed by the prover is uniformly distributed (to see this, note that once f^{-1} is fixed independent of r then $r'_i = h(f^{-1}(r_i))$

[1] Again, see [6, Appendix C] for more careful treatment of what assumptions are necessary.

[2] This assumption is necessary for the construction given below. However, it is possible to remove this assumption using a more complicated construction [1].

is an unbiased bit for each i). Having the prover send $z_i = f^{-1}(r_i)$ to the verifier has the effect of "revealing" the i^{th} bit of r' to the verifier; also once f^{-1} is fixed the prover cannot "cheat" by changing the value of r'_i (since the verifier will check that $f(z_i) = r_i$ before computing $r'_i = h(x_i)$, and the inverse of r_i under f is unique). Finally, at least informally, the bits of r' that are not revealed by the prover to the verifier remain "hidden" by the security of f^{-1} and its associated hard-core bit h.

It is immediate that $(\mathcal{P}, \mathcal{V})$ satisfies completeness. Next, we prove the soundness of $(\mathcal{P}, \mathcal{V})$. Say that $\mathcal{P}^*$ *cheats* if it outputs a valid proof for some $x \notin L$. First consider any fixed (f, f^{-1}). By what we have said above, the string r' generated using this pair will be uniformly distributed and hence the soundness of $(\mathcal{P}', \mathcal{V}')$ implies that for any cheating $\mathcal{P}^*$ we have

$$\Pr_r[\mathcal{P}^* \text{ can cheat using } f] \leq 2^{-2k}.$$

However, there is nothing to prevent $\mathcal{P}^*$ from generating and using some (f, f^{-1}) in a way which *depends* on r! (For example, it is easy to construct a cheating prover that always picks (f, f^{-1}) in such a way that $r'_1 = 0$, say.) We now use the fact that the number of possible (valid) f's is at most 2^k, and also that $\mathcal{P}^*$ cannot send an invalid f to $\mathcal{V}$ without being caught. Summing the above inequality over all possible f's shows that:

$$\Pr_r[\mathcal{P}^* \text{ can cheat using } \textit{any } f] \leq 2^k \cdot 2^{-2k} = 2^{-k},$$

which is negligible.

To complete the proof, we show a zero-knowledge simulator for $(\mathcal{P}, \mathcal{V})$. Let Sim' be the simulator for $(\mathcal{P}', \mathcal{V}')$. We construct simulator Sim for $(\mathcal{P}, \mathcal{V})$ as follows:

$\underline{\mathsf{Sim}(1^k, x)}$
$(r'_I, \pi, I) \leftarrow \mathsf{Sim}'(1^k, x)$;
$(f, f^{-1}) \leftarrow \mathsf{Gen}(1^k)$;
for $i \in I$:
  $z_i \leftarrow \{0,1\}^k$ s.t. $h(z_i) = r'_i$;
  $r_i = f(z_i)$;
for $i \notin I$:
  $r_i \leftarrow \{0,1\}^k$;
output $(r, f, \pi, I, \{z_i\}_{i \in I})$

Intuitively, there are two differences between real proofs (given by $\mathcal{P}$) and simulated proofs (given by Sim): first, the simulated proofs use the simulator Sim' for the original proof system rather than the actual prover $\mathcal{P}'$ for the original proof system. Second, the values $\{r_i\}_{i \notin I}$ now define completely random bits $\{r'_i\}_{i \notin I}$ in the underlying string r'; this is not necessarily so for real proofs. However, a hybrid argument will show that the above differences are inconsequential: the first due to the zero-knowledge of $(\mathcal{P}', \mathcal{V}')$ and the second due to the security of the trapdoor permutation family.

Formally, let A be a PPT algorithm. Our goal is to show that

$$\left\{(x, w) \leftarrow A(1^k); r \leftarrow \{0,1\}^{k \cdot n}; (f, \pi, I, \{z_i\}_{i \in I}) \leftarrow \mathcal{P}(1^k, r, x, w) : (r, x, f, \pi, I, \{z_i\}_{i \in I})\right\} \quad (1)$$

and

$$\left\{(x, w) \leftarrow A(1^k); (r, f, \pi, I, \{z_i\}_{i \in I}) \leftarrow \mathsf{Sim}(1^k, x) : (r, x, f, \pi, I, \{z_i\}_{i \in I})\right\} \quad (2)$$

are computationally indistinguishable (cf. Definition 1). We define an intermediate experiment via an algorithm Hybrid as follows:

$\underline{\mathsf{Hybrid}(1^k, x, w)}$
$r' \leftarrow \{0,1\}^n$;
$(\pi, I) \leftarrow \mathcal{P}'(1^k, r', x, w)$;
$(f, f^{-1}) \leftarrow \mathsf{Gen}(1^k)$;
for $i \in I$:
  $z_i \leftarrow \{0,1\}^k$ s.t. $h(z_i) = r'_i$;
  $r_i = f(z_i)$;
for $i \notin I$:
  $r_i \leftarrow \{0,1\}^k$;
output $(r, f, \pi, I, \{z_i\}_{i \in I})$

Claim 2 *Assuming that $(\mathcal{P}', \mathcal{V}')$ is an NIZK proof system in the hidden-bits model with simulation Sim', ensemble (2) is computationally indistinguishable from the following:*

$$\left\{(x,w) \leftarrow A(1^k); (r, f, \pi, I, \{z_i\}_{i \in I}) \leftarrow \mathsf{Hybrid}(1^k, x, w) : (r, x, f, \pi, I, \{z_i\}_{i \in I})\right\}. \tag{3}$$

Assume to the contrary that the claim is false. Then there is a PPT distinguisher D which can distinguish between the two ensembles with probability that is not negligible. We construct a D' which violates the claimed security of Sim' as a zero-knowledge simulator for the proof system $(\mathcal{P}', \mathcal{V}')$ in the hidden-bits model.

Let A output (x, w) as above. D' is then given a tuple (r'_I, x, π, I) (coming either from the real prover $\mathcal{P}'$ in the hidden-bits model, or from Sim') and runs as follows:

$\underline{D'(1^k, r'_I, x, \pi, I)}$
$(f, f^{-1}) \leftarrow \mathsf{Gen}(1^k)$;
for $i \in I$:
  $z_i \leftarrow \{0,1\}^k$ s.t. $h(z_i) = r'_i$;
  $r_i = f(z_i)$;
for $i \notin I$:
  $r_i \leftarrow \{0,1\}^k$;
output $D(r, x, f, \pi, I, \{z_i\}_{i \in I})$

One can check that if (r'_I, x, π, I) is distributed according to real proofs generated by $\mathcal{P}'$, then the input to D is distributed according to (3). On the other hand, if (r'_I, x, π, I) is distributed as the output of Sim', then the input to D is distributed according to (2). So the distinguishing advantage of D' (in distinguishing real proofs generated by $\mathcal{P}'$ from simulated proofs generated by Sim') is equal to the distinguishing advantage of D. But this contradicts the zero-knowledge property of $(\mathcal{P}', \mathcal{V}')$ with respect to Sim'. □

Claim 3 *Assuming that Gen defines a secure trapdoor permutation family, ensemble (1) is computationally indistinguishable from ensemble (3).*

Again, we prove this by contradiction. Assume to the contrary that there is a PPT distinguisher D which can distinguish between the two ensembles with a probability that is not

negligible. We then construct a D' that violates the security of the trapdoor permutation family. Before doing so, we note the following easy-to-prove fact. The security of Gen (and a standard hybrid argument) implies that no PPT algorithm D', given a randomly-generated f, outputting a sequence of bits $r'_1, \ldots, r'_\ell$, and receiving in return a sequence of k-bit values $r_1, \ldots, r_\ell$, can distinguish between the case when each r_i is randomly chosen in $\{0,1\}^k$ and the case when each r_i is randomly chosen in $\{0,1\}^k$ subject to $h(f^{-1}(r_i)) = r'_i$.

Define D' as follows:

$\underline{D'(1^k, f)}$
$(x, w) \leftarrow A(1^k)$;
$r' \leftarrow \{0,1\}^n$;
$(\pi, I) \leftarrow \mathcal{P}'(1^k, r', x, w)$;
for $i \in I$:
  $z_i \leftarrow \{0,1\}^k$ s.t. $h(z_i) = r'_i$;
  $r_i = f(z_i)$;
output $r'_{\bar{I}}$ and get back $r_{\bar{I}}$;
// note: for all i, $|r'_i| = 1$ and $|r_i| = k$
output $D(r, x, f, \pi, I, \{z_i\}_{i \in I})$

It is easy to see that in case the values $r_{\bar{I}}$ are randomly chosen in $\{0,1\}^k$, then the input to D is distributed according to (3). Though harder to see, it is also the case that when the values $r_{\bar{I}}$ are randomly chosen in $\{0,1\}^k$ subject to $h(f^{-1}(r_i)) = r'_i$ then the input to D is distributed according to (1). To see this, note that in (1) r and f are chosen (independently) at random and these are used to generate r' by setting $r'_i = h(f^{-1}(r_i))$; on the other hand, in the above experiment (when the case in question occurs) r' and f are chosen (independently) at random and then r is chosen randomly subject to $h(f^{-1}(r_i)) = r'_i$. Thus, the distributions on (r, f, r') are the same in both cases. Furthermore, these values determine the remaining values in each experiment in the same way.

The above shows that the distinguishing advantage of D' (in determining which case occurs) is equal to the distinguishing advantage of D. But, as we have noted above, this contradicts the security of the trapdoor permutation family. □

The above two claims complete the proof of the theorem via a standard hybrid argument. ■

References

[1] M. Bellare and M. Yung. Certifying Permutations: Non-Interactive Zero-Knowledge Based on any Trapdoor Permutation. *J. Crypto.* 9: 149–166, 1996.

[2] U. Feige. *Alternative Models for Zero-Knowledge Interactive Proofs.* PhD Thesis, Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, 1990. Available from http://www.wisdom.weizmann.ac.il/~feige.

[3] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String. *FOCS*, pp. 308–317, 1990.

[4] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM J. Computing*, Vol. 29, No. 1, pp. 1–28, 1999.

[5] O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools*, Cambridge University Press, 2001.

[6] O. Goldreich. *Foundations of Cryptography, vol. 2*, to appear. Preliminary versions available from Goldreich's web page.

[7] D. Lapidot and A. Shamir. Publicly Verifiable Non-Interactive Zero-Knowledge Proofs. *Advances in Cryptology — CRYPTO '90*, pp. 353–365, 1990.