CMSC 858K — Advanced Topics in Cryptography February 10, 2004

Lecture 5

Lecturer: Jonathan Katz *Scribe(s):* *Rengarajan Aravamudhan* *Morgan Kleene* *Nan Wang* *Aaron Zollman*

1 Semantic Security May Not be Enough

The following trick can be played on the El-Gamal encryption scheme which, under the Decisional Diffie-Hellman Hypothesis (DDH) is semantically secure. We show how an attacker can subvert a sealed-bid auction conducted using El-Gamal encryption, where the auction is won by the bidder who submits the higher bid. Assume the auctioneer has public key $PK = (g, y = g^x)$, and let the bid of the first bidder be m.

Bidder 1	$C \leftarrow (g^r, y^r \cdot m)$ (where r is random)	$\xrightarrow{C=(C_1,C_2)}$	Auctioneer decrypts m
Bidder 2	$C' = (C_1, C_2 \cdot \alpha)$ (where $\alpha = 2$)	$\xrightarrow{C'}$	Auctioneer decrypts $m' = m \cdot \alpha$

Although Bidder 2 has no idea what was bid (he doesn't even know his own bid!), he is still able to outbid bidder 1 by a factor of α.

The following system for verifying credit cards is also malleable. A user has a credit card number $C_1, C_2, C_3, ..., C_{48}$ (where each C_i represents one bit) which is encrypted, bit-wise, with the merchant's public key pk and sent to the merchant as follows:

$$E_{pk}(C_1), E_{pk}(C_2), E_{pk}(C_3), ..., E_{pk}(C_{48})$$

The merchant then immediately responds ACCEPT or REJECT, indicating whether the credit card is valid. Now, an adversary need not decrypt the message to recover the credit card: consider what happens if the first element of the above ciphertext is replaced by $E_{pk}(0)$ (which an attacker can compute since the public key is available!) — if the message is accepted by the merchant, the first bit of the credit card must be zero; if rejected, it is one. Continuing in this way, the adversary learns the entire credit card number after 48 such attempts.

These two examples motivate the concept of *malleability* [4]. Informally, an encryption scheme is *malleable* if, given an encryption C of some message M, it is possible to construct a different ciphertext C' decrypting to some "related" message M'. Non-malleability precludes the attacks shown above (in particular). Attacks that are thematically similar to the ones given above have been implemented [2], although they are much more complicated than the above examples.

This motivates the development of stronger notions of security preventing the above attacks. It turns out that non-malleability is (for the cases of interest here) equivalent [1] to a security property which is simpler to define called *security against chosen-ciphertext attacks*. We may further consider security against *non-adaptive chosen-ciphertext attacks* (**CCA1**) or security against *adaptive chosen-ciphertext attack* (**CCA2**); we define both of these now.

Definition 1 [IND-CCA2] An encryption scheme is secure against adaptive chosen-ciphertext attacks (CCA2) if the following is negligible for all PPT adversaries $\mathcal{A}$:

$$\left|\Pr\left[\begin{array}{c}(pk, sk) \leftarrow \mathsf{KeyGen}(1^k); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{D}_{sk}(\cdot)}(pk); \\ b \leftarrow \{0,1\}; c \leftarrow \mathcal{E}_{pk}(m_b); b' \leftarrow \mathcal{A}^{\mathcal{D}_{sk}(\cdot)}(pk, c)\end{array} \quad : \quad b = b'\right] - \frac{1}{2}\right|$$

where $\mathcal{A}$ cannot query $\mathcal{D}_{sk}(c)$. ♢

We note that for non-adaptive chosen-ciphertext attacks (CCA1) the adversary is only allowed to query $\mathcal{D}_{sk}(\cdot)$ in the first stage (i.e., before being given the ciphertext c).

2 Zero-Knowledge Proofs

Toward our eventual goal of designing encryption schemes secure against chosen-ciphertext attacks, we define a class of exchanges for which it holds that one party is able to convince another that he holds some information without revealing the information itself. We first review what kinds of computation we consider feasible and then discuss the actual exchanges that have been devised.

2.1 NP-Completeness

The kinds of computations that can be carried out efficiently are typically considered to be those that can be done in polynomial time. We consider computational problems as the recognition of a set of strings, referred to as a *language*. We say a Turing machine M accepts a language L if: $x \in L \Leftrightarrow M(x)$ outputs "accept". We let a "1" signify acceptance.

There are two sets of computational problems which are of special importance. The first is the set of languages that can be decided in polynomial time, denoted P. Formally, a language L is in P if there exists a Turing machine M which takes at most $p(|x|)$ steps for some polynomial p (where $|x|$ denotes the length of its input string x), and accepts if and only if $x \in L$. The class NP is the set of languages for which there exist proofs of membership that can be checked in polynomial time. Formally, a language L is in the class NP if there exists a polynomial-time Turing machine M[1] such that:

$$x \in L \Leftrightarrow \text{there exists a string } w_x \text{ s.t. } M(x, w_x) = 1.$$

A w_x of this sort is called a *witness for x*. One can think of this as an efficiently-verifiable "proof" that $x \in L$.

Intuitively, if we can use a solution to problem A to solve problem B it seems as if problem A is in some sense "(at least) as hard as" problem B. This is formalized by the notion of a *polynomial-time reduction* between two languages. We say that language L_1 is

[1]By convention, the running time of a Turing machine taking multiple inputs is measured as a function of the length of its *first* input.

poly-time reducible to language L_2 if there exists a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that: (1) f is computable in polynomial time, and (2) $x \in L_1$ if and only if $f(x) \in L_2$. We will sometimes abbreviate this by writing $L_1 \leq_p L_2$. Note that if $L_1 \leq_p L_2$ and $L_2 \in P$ (i.e., L_2 can be decided in polynomial time) then L_1 can be decided in polynomial time using the following algorithm: Given a string x, compute $x' = f(x)$ and then decide whether $x' \in L_2$. Similarly, if $L_1 \leq_p L_2$ and $L_2 \in NP$ then $L_1 \in NP$ as well.

There are languages which, in a certain sense, are "the hardest languages" in NP in the sense that all problems in NP are poly-time reducible to them. These problems are called *NP complete*. Note that if an NP-complete problem could be shown to be in P, then *all* of NP would be in P, by the discussion above. A classic example of an NP-complete language is satisfiability (i.e., given a boolean formula does there exist an assignment of truth values to variables such that the formula evaluates to true). There are a variety of other well-known NP-complete problems; the ones we will encounter in this class are:

- Hamiltonian Cycle: This is the lnaguage $\{\ G : G$ is a graph which contains a Hamilton cycle $\}$. (A *Hamiltonian cycle* is a cycle in which each vertex appears exactly once.)
- 3-colorability: This is the language $\{\ G = (V, E) : G$ is a graph and there exists a function $\phi : V \rightarrow \{Red,\ Green,\ Blue\}$ such that if $\{u, v\} \in E$, $\phi(u) \neq \phi(v)\ \}$

Looking (way) ahead, we will eventually show a proof system for all of NP by showing a proof system for a particular NP-complete language.

2.2 Non-Interactive Proof Systems

We first informally discuss the notion of an *interactive proof system.* Here we have a prover P and a polynomial-time verifier V who both have some common input x, and the prover wants to convince V that $x \in L$ for some agreed-upon language L. The prover will attempt to do this by interacting with the verifier in multiple communication rounds. Informally, we would like that if $x \in L$ then the prover can always make the verifier accept, while if $x \notin L$ then no matter what the prover does the verifier should reject with high probability.

We first give two trivial examples: we claim that all languages in P have an interactive proof system with 0 rounds. Here, P does not communicate with V at all, but V merely decides on its own whether $x \in L$ (it can do this since $L \in P$). Next, we claim that any $L \in NP$ has a 1-round interactive proof system in which P simply sends to V the witness for x (assuming $x \in L$), and V verifies this. Note that the definition of NP implies that V will always accept if $x \in L$ (assuming P wants to make V accept) and that P can *never* fool V into accepting if $x \notin L$.

Things get more interesting when we allow more rounds of interaction (and languages outside of NP can be shown to have interactive proof systems), but this is for a class in complexity theory. We will be more interested in strengthening the model of interactive proofs so as to require also that V does not learn anything from interacting with the prover other than the fact that $x \in L$. So far we have described a scheme that does not specify what V may learn from the interaction. The below definition is motivated by the desire to let V ascertain with high probability whether a particular string is in its language of interest without actually learning anything about *why* it is in the language. To put it another way, V should not learn anything from P that he could not have figured out himself. We formalize this now for the case of non-interactive proofs (where there is additionally a common random string available to both parties), and later in the course we will formalize it for interactive proofs. See [3, 5] for more details.

Definition 2 A pair of PPT algorithms[2] (P, V) is a *non-interactive zero-knowledge (NIZK) proof system* for a language $L \in NP$ if:

Completeness For any $x \in L$ (with $|x| = k$) and witness w for x, we have:

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

In words: a random string r is given to both parties. P is given r, x, and the witness that $x \in L$, and produces a proof π which he sends to V. The verifier, given r, x, and π, decides whether to accept or reject. The above just says that if $x \in L$ and everyone is honest, then V always accepts.

Soundness If $x \notin L$ then $\forall P^*$ (even all-powerful P^*), the following is negligible (in $|x| = k$):

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

Zero-knowledge There exists a PPT simulator S such that for all $x \in L$ (with $|x| = k$, the security parameter) and any witness w for x, the following distributions are computationally indistinguishable:

1. $\{r \leftarrow \{0,1\}^{\mathsf{poly}(k)}; \pi \leftarrow P(r,x,w) : (r,x,\pi)\}$
2. $\{(r,\pi) \leftarrow S(x) : (r,x,\pi)\}$.

$\diamondsuit$

The last condition restricts the information V may obtain from P. Intuitively, it says that if V has "learned" anything from interacting with P he could also have learned it by himself, using the polynomial-time simulator S.

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. Crypto '98.

[2] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS. Crypto '98.

[3] M. Blum, P. Feldman, and S. Micali. Non-interactive Zero-Knowledge and its Applications. STOC '88.

[4] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. *SIAM J. Computing* 30(2): 391–437, 2000.

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

[2] Here, we require that P run in probabilistic polynomial time as well, since we are going to eventually want to use P to construct efficient cryptographic protocols!