Cryptography lecture notes

Marko Horvat, taken from Arora-Barak

January 16, 2018

Introduction

- Cryptography enables security
  - Secure browsing, voting, cryptocurrencies, smart contracts, etc.
- Computational complexity enables cryptography
  - Symmetric and asymmetric keys, one-way functions, encryption schemes, digital signatures, zero-knowledge proofs, etc.
  - These cryptographic primitives should not be breakable by *any* realistic adversary
  - Realistic=efficient, (probabilistic) polynomial-time
  - Security proofs via reductions: breaking crypto would lead to breaking a convincingly unbreakable computational hardness assumption
- Kerckhoff's principle: adversary knows our methods
  - Under some restrictions, we can allow even more: he has access to encryption/decryption oracles
- We try to make *realistic adversary* precise because achieving security against *all* adversaries is highly impractical
  - The biggest hindrance is: keys need to be as long as messages

Perfect secrecy and its limitations

- Alice wants to send a secret message x (the *plaintext*) to Bob, but an adversary Eve is eavesedropping on the communication channel
- One way for Alice to do so is to encrypt the message using some information that Bob has, but Eve does not; we call it the *key* k
- Alice can then send the encrypted message (the *ciphertext*) $E_k(x)$, so that Bob can decrypt it upon receipt to recover the plaintext
- Note that Alice and Bob need to have agreed on two things beforehand:
  - the key k; an important part of cryptography and information security is devoted to *authenticated key exchange (AKE)*
  - the *encryption scheme*: a pair of algorithms (E, D) such that for all messages $x \in \{0,1\}^m$ and keys $k \in \{0,1\}^n$, $D_k(E_k(x)) = x$

– e.g. one-time pad/Vernham cipher: $x \oplus k$ where k is a random bit string

- Ideally, we want the scheme to reveal *nothing* about the payload to the adversary
  – e.g. getting just the first character should be impossible

Definition 1. *An encryption scheme (E, D) is* perfectly secret *if for all x, x', the random variables $E_{U_n}(x)$ and $E_{U_n}(x')$ are identically distributed (U_n is the uniform distribution over $\{0,1\}^n$).*

– Example: one-time pad (never reuse keys!)

Example 1 (Exercise 9.1)**.** *The one-time pad is perfectly secret.*

Proof sketch. Assume $m = n = 1$, and let $X, K : \Omega \to \{0,1\}$ be independent random variables. We know

HW: m, n arbitrary

$$P(K=0) = P(K=1) = \frac{1}{2}$$
$$P(X=0) = p; \quad P(X=1) = 1-p$$

$$\begin{aligned} P(X \oplus K = 0) &= P(X=0|K=0) \cdot P(K=0) \\ &\quad + P(X=1|K=1) \cdot P(K=1) \\ &= p \cdot \frac{1}{2} + (1-p) \cdot \frac{1}{2} = \frac{1}{2} \end{aligned}$$

□

- However, the one-time pad is not practical enough to secure communications

Example 2 (Exercise 9.2)**.** *Every perfectly secret encryption scheme uses keys at least as long as messages.*

Proof. Let (E, D) be a perfectly secret encryption scheme. Assume to the contrary that $n < m$. Pick any x, k_1 and consider $E_{k_1}(x)$. Then

$$D_{k_1}(E_{k_1}(x))$$
$$D_{k_2}(E_{k_1}(x))$$
$$\vdots$$
$$D_{k_{2^n}}(E_{k_1}(x))$$

are all the messages that can be encrypted to $E_{k_1}(x)$ and there is at most 2^n of them. Due to $n < m$, this is strictly less than the number 2^m of all messages. Hence there exists

$$x' \in \{0,1\}^m \setminus \{D_{k_1}(E_{k_1}(x)), \ldots, D_{k_{2^n}}(E_{k_1}(x))\}.$$

This implies

$$P(E_{U_n}(x') = E_{k_1}(x)) = 0 \text{ and}$$
$$P(E_{U_n}(x) = E_{k_1}(x)) > 0.$$

HW: How much is the probability exactly?

□

Computational security

Question: Can an encryption scheme be used with short keys to achieve perfect secrecy *with respect to a polynomial-time adversary*?

Theorem 1. *Assume $P = NP$. Let (E, D) be an encryption scheme with $n < m$. Then there is a polynomial-time algorithm A such that for every input length m, there is a pair $x_0, x_1 \in \{0,1\}^m$ such that*

$$\Pr_{\substack{b \in_R \{0,1\} \\ k \in_R \{0,1\}^n}} (A(E_k(x_b)) = b) \geq \frac{3}{4}.$$

Proof. Let $x_0 = 0^m$, $S = \{E_k(x_0) : k \in \{0,1\}^n\}$ and $A(x) = \mathbf{1}_{\{0,1\}^m \setminus S}(x)$. Then

$$\begin{aligned}\Pr_{\substack{b \in_R \{0,1\} \\ k \in_R \{0,1\}^n}} (A(E_k(x_b)) = b) &= \frac{1}{2}\Pr(A(E_{U_n}(x_0)) = 0) + \frac{1}{2}\Pr(A(E_{U_n}(x_1)) = 1) \\ &= \frac{1}{2} + \frac{1}{2}\Pr(A(E_{U_n}(x_1)) = 1).\end{aligned}$$

It suffices to prove there exists x_1 such that $\Pr(A(E_{U_n}(x_1)) = 1) \geq \frac{1}{2}$, i.e. $\Pr(E_{U_n}(x_1) \in S) \leq \frac{1}{2}$. Suppose otherwise that for all x_1, $\Pr(E_{U_n}(x_1) \in S) > \frac{1}{2}$. Then for $X \sim U_m$,

$$\Pr(E_{U_n}(X) \in S) = \sum_x \Pr(E_{U_n}(x) \in S | X = x)\Pr(X = x) > \frac{1}{2}\sum_x \Pr(X = x) = \frac{1}{2}$$

$$\Pr(E_{U_n}(X) \in S) = \sum_{\substack{k,x \\ E_k(x) \in S}} \Pr(U_n = k)\Pr(X = x) = |\{(k,x) : E_k(x) \in S\}| 2^{-n} 2^{-m} \overset{(*)}{\leq} \frac{1}{2}$$

We know $(*)$ because for each k, the injection $x \mapsto E_k(x)$ maps at most $|S| \leq 2^n$ x's to S. □

Answer: Perhaps, but only if **P** ≠ **NP** (whether this assumption is enough is an open problem). We will strengthen this assumption (by assuming that *one-way permutations* exist) to construct a *computationally secure* encryption scheme.

One way functions: Definition and some examples

Definition 2 (Negligible function)**.** *A function $\epsilon : \mathbb{N} \to [0,1]$ is called* negligible *if*

$$\forall c \in \mathbb{N}. \exists n_0 \in \mathbb{N}. \forall n \geq n_0. \epsilon(n) < n^{-c}.$$

We also write this as $\epsilon(n) = n^{-\omega(1)}$.

Definition 3 (One-way function)**.** *A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is a* one-way function *if for every probabilistic polynomial-time algorithm A, there is a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that for every n,*

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = f(x)}} (A(y) = x' \text{ s.t. } f(x') = y) < \epsilon(n)$$

- Existence of one-way functions implies **P** ≠ **NP**
  - Intuitively, a one-way function is easy to compute, but hard to invert
  - It is unknown whether the converse holds
  - Conjectured one-way functions: multiplication, the RSA function, AES, etc.

Encryption from one-way functions

- We set out to design a reasonably secure encryption scheme with reasonably short keys against an efficient adversary
- There are many definitions of security of varying strength; we choose a fairly simple one

Definition 4 (Computationally secure encryption scheme)**.** *We say that an encryption scheme* (E, D) *is* computationally secure *if for every probabilistic polynomial-time algorithm* A*, there is a negligible function* $\epsilon : \mathbb{N} \to [0, 1]$ *such that*

$$\Pr_{\substack{k \in_R \{0,1\}^n \\ x \in_R \{0,1\}^m}} (A(E_k(x)) = (i, b) \text{ s.t. } x_i = b) \leq \frac{1}{2} + \epsilon(n)$$

- The one-time pad is a computationally secure encryption scheme (Exercise 9.3), but it warrants long keys, i.e. shared random strings
- For every $c \in \mathbb{N}$, we can *stretch* random strings of length n to *pseudorandom* strings of length n^c by using *pseudorandom generators*
  - Pseudorandom strings cannot be *distinguished* from random by a poly-time adversary

Definition 5 (Polynomial-time computable function of stretch l)**.** *Let* $G : \{0,1\}^* \to \{0,1\}^*$ *and* $l : \mathbb{N} \to \mathbb{N}$ *be polynomial-time computable functions such that for every* $n \in \mathbb{N}$*,* $l(n) > n$*. We say that* G *is a* polynomial-time computable function of stretch l *if for all* $x \in \{0,1\}^*$*,* $|G(x)| = l(|x|)$*.*

Definition 6 (Secure pseudorandom generator of stretch l)**.** *Let* G *be a polynomial-time computable function of stretch* l*. We say that* G *is a* secure pseudorandom generator of stretch l *if for all probabilistic polynomial-time algorithms* A*, there exists a negligible function* $\epsilon : \mathbb{N} \to [0, 1]$ *such that for all* $n \in \mathbb{N}$*,*

$$|\Pr(A(G(U_n)) = 1) - \Pr(A(U_{l(n)}) = 1)| < \epsilon(n).$$

- Can we take a short random string of length n, apply to it a suitable secure pseudorandom generator, and use the pseudorandom string of length n^c as a key with the one-time pad to securely encrypt a plaintext of length n^c?
  - Yes! If a poly-time adversary A could predict a bit of the plaintext with chance much greater than $\frac{1}{2}$ in polynomial time, it could distinguish between a random and pseudorandom key having been used in the one-time pad with this probability
- In order to get the existence of such PRGs, we will assume that one-way permutations exist
  - It is actually enough to assume the existence of one-way functions (Håstad et al., 1999)

Lemma 1. *Suppose that there exists a bijective one-way function* $f : \{0,1\}^* \to \{0,1\}^*$ *such that for all* $x \in \{0,1\}^*$*,* $|f(x)| = |x|$*. Then for every* $c \in \mathbb{N}$*, there exists a secure pseudorandom generator of stretch* n^c*.*

- To prove this lemma, we need two famous results:

  Yao (1982): unpredictability implies pseudorandomness

Goldreich-Levin (1989): construction of PRG of stretch 1 from one-way permutation

Definition 7 (Unpredictable function of stretch l)**.** *Let G be a polynomial-time computable function of stretch l. We say that G is an* unpredictable function of stretch l *if for every probabilistic poly-time algorithm B, there is a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that for all $n \in \mathbb{N}$,*

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = G(x) \\ i \in_R \{1,\ldots,l(n)\}}} (B(1^n, y_1, \ldots, y_{i-1}) = y_i) \leq \frac{1}{2} + \epsilon(n).$$

- It is easy to see that a predictor B for a function G of stretch l breaks pseudorandomness:
  - Define $A(y)$ as 1 if $l(n) = |y|$ and $B(1^n, y_1, \ldots, y_{l(n)-1}) = y_{l(n)}$, and 0 otherwise
  - Then there exists an ϵ such that for all $n \in \mathbb{N}$,

$$\Pr(A(U_{l(n)}) = 1) \leq \frac{1}{2} + \epsilon(n)$$

  - For 2ϵ (as well as any other negligible function), there is an n_0 such that

$$\Pr(A(G(U_n)) = 1) \geq \frac{1}{2} + 2\epsilon(n_0)$$

$$|\Pr(A(G(U_{n_0})) = 1) - \Pr(A(U_{l(n_0)}) = 1)| \geq \epsilon(n_0)$$

Theorem 2 (Yao, 1982)**.** *Let G be an unpredictable function of stretch l. Then G is a secure pseudorandom generator. Moreover, for every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm B such that for every $n \in \mathbb{N}$ and $\epsilon > 0$, if $\Pr(A(G(U_n)) = 1) - \Pr(A(U_{l(n)}) = 1) \geq \epsilon$, then*

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = G(x) \\ i \in_R \{1,\ldots,l(n)\}}} (B(1^n, y_1, \ldots, y_{i-1}) = y_i) \geq \frac{1}{2} + \frac{\epsilon}{l(n)}.$$

Proof. It suffices to prove the second part of the theorem. Assuming the second part and that G is not a PRG, there is some algorithm A and $c \in \mathbb{N}$ such that for infinitely many $n \in \mathbb{N}$, we have

$$|\Pr(A(G(U_n)) = 1) - \Pr(A(U_{l(n)}) = 1)| \geq n^{-c}.$$

We can ensure that the above holds without the absolute value (we might need to replace A with $1 - A$); then by our assumption there is a predictor B that succeeds with probability at least $\frac{1}{2} + \frac{n^{-c}}{l(n)}$.

We now prove the second part. Let A be more likely to output 1 for inputs drawn from $G(U_n)$ than $U_{l(n)}$. We now define B; on input $(1^n, y_1, \ldots, y_{i-1})$, B draws bits $z_i, \ldots, z_{l(n)}$ independently and uniformly at random, and outputs z_i if $A(y_1, \ldots, y_{i-1}, z_i, \ldots, z_{l(n)}) = 1$; otherwise, B outputs $1 - z_i$.

The next step is called *the hybrid argument*. For every i, we define the distribution D_i; choose $x \in_R \{0,1\}^n$ and let $y = G(x)$, draw bits $z_{i+1}, \ldots, z_{l(n)}$ independently and uniformly at random, and output $y_1, \ldots, y_i, z_{i+1}, \ldots, z_{l(n)}$. We define $p_i = \Pr(A(D_i) = 1)$ and compute

$$\epsilon \leq p_{l(n)} - p_0 = (p_{l(n)} - p_{l(n)-1}) + (p_{l(n)-1} - p_{l(n)-2}) + \ldots + (p_1 - p_0) = l(n) \cdot E_{i \in \{1,\ldots,l(n)\}}(p_i - p_{i-1}).$$

We also have for all $i \in \{1, \ldots, l(n)\}$,

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = G(x)}}(B(1^n, y_1, \ldots, y_{i-1}) = y_i) = \frac{1}{2}\Pr(A(D_i) = 1 | z_i = y_i) + \frac{1}{2}(1 - \Pr(A(D_i) = 1 | z_i = 1 - y_i))$$
$$= \frac{1}{2} - \Pr(A(D_i) = 1) + \Pr(A(D_i) = 1 | z_i = y_i) = \frac{1}{2} + p_i - p_{i-1}.$$

Now we can combine the results to get

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = G(x) \\ i \in_R \{1,\ldots,l(n)\}}}(B(1^n, y_1, \ldots, y_{i-1}) = y_i) = E_{i \in \{1,\ldots,l(n)\}}(\Pr_{\substack{x \in_R \{0,1\}^n \\ y = G(x)}}(B(1^n, y_1, \ldots, y_{i-1}) = y_i))$$
$$= \frac{1}{2} + E_{i \in \{1,\ldots,l(n)\}}(p_i - p_{i-1}) \geq \frac{1}{2} + \frac{\epsilon}{l(n)}.$$

□

HW: Complete the construction of a computationally secure encryption scheme that uses short keys against a poly-time adversary.

- Missing ingredient: Goldreich-Levin
  - If f is a one-way permutation of length n, then $G(x, r) = (f(x), r, x \odot r)$ is a PRG of stretch 1.
- Yao's theorem: extension to arbitrary stretch
- Then otp can be used
  - Keys should never be reused if XOR-ed directly: $(x \oplus k) \oplus (x' \oplus k) = x \oplus x'$
  - Either use each key only once, or combine otp with Goldreich-Goldwasser-Micali (GGM)!
  - GGM construction: suppose G is a n-to-$2n$ PRG; rather than sending $x \oplus k$, choose $r \in_R \{0,1\}^n$ and send $(r, x \oplus f_k(r))$, where $f_k(r) = G_{k_n}(\ldots(G_{k_2}(G_{k_1}(r)))\ldots)$ and $G_0(r), G_1(r)$ are the left and right half of $G(r)$, respectively
  - Another application: MAC (Message Authentication Code)—send $(x, r, f_k(x, r))$ to ensure integrity of x

Zero-knowledge proofs

- In order to convince somebody that a statement is true, we might want to avoid saying *why* it is true
  - We know a way to save millions for a future employer—say an airline where we have a more efficient flight schedule—we want to prove this during the job interview, but without revealing any details about the schedule
  - Idea: run an interactive probabilistic proof where the verifier learns only what it could have computed by itself, without interaction

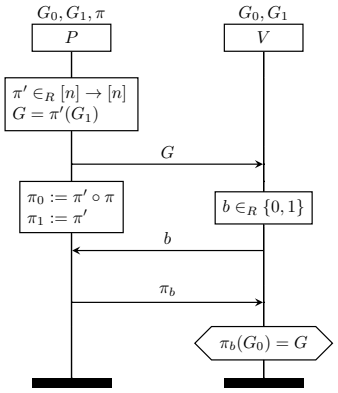
– Formal definition of *perfect zero knowledge* for $L \in \mathbf{IP} \cap \mathbf{NP}$ and prover P for L: For every probabilistic polynomial-time interactive strategy V^*, there exists an expected probabilistic polynomial-time (stand-alone) algorithm S^* such that for all $x \in L$ and certificates u,

$$out_{V*}\langle P(x,u), V^*(x)\rangle \equiv S^*(x).$$

- This condition can be relaxed (Exercise 9.17)
- Simulation is a central idea in enabling security through crypto (Exercise 9.9, semantic security; Section 9.5.4, secure multi-party computation)

Zero-knowledge proof for graph isomorphism

Public input: graphs G_0, G_1 with n vertices
Prover's private input: permutation $\pi : [n] \to [n]$ such that $G_1 = \pi(G_0)$ (π permutes rows and columns of adjacency matrix)



- Completeness: if P and V follow the protocol, V accepts with probability 1
- Soundness: if G_0 and G_1 are not isomorphic, V rejects with probability $\frac{1}{2}$ (there is a b for which $\pi_b(G_0)$ is not equal to G, and that b is chosen with probability $\frac{1}{2}$)

HW: Perfect zero knowledge

Tossing coins over the phone

- Both parties contribute, last to reveal must secretly commit
- Assuming we have a one-way permutation f_n, we can apply the Goldreich-Levin theorem