Foundations of Cryptography: Lecture 2

Lecture by Moni Naor, scribe notes by Yarden Sheffer

November 8, 2021

A brief review of the previous lecture

The fundamental idea of cryptography is that computational hardness can be good news: we can use the computational intractability of certain problems to construct secure systems. We also gave an example for a secure system that does not rely on the computational power of the adversary: the secret sharing protocol. In the t-out-of-n secret sharing protocol there is a dealer that has a secret s and distributes shares $a_1, ..., a_n$ among n users, such that an adversary, no matter how strong, cannot obtain information about the secret unless they have t or more shares. Formally, for all secrets s_1, s_2, users $i_1, ..., i_{t-1}$ the shares $a_{i_1}, ..., a_{i_{t-1}}$ are distributed independently of whether $s = s_1$ or $s = s_2$.

Identification & Authentication

Consider the problem of approving a single message. In the "single guard" setup from the previous lecture: We have Alice and Bob sharing some secret setup, and an identification step in which Alice sends information to Bob and at which Eve can interfere (see Fig. 1). We have two requirements from the system:

- *Completeness*: If Eve does not interfere and Alice approves (sends the approve message) then Bob should "accept".
- *Soundness:* If Alice does not "approve" then, no matter what Eve does, Bob does not accept (except with probability ϵ).

Quote Boaz Barak:

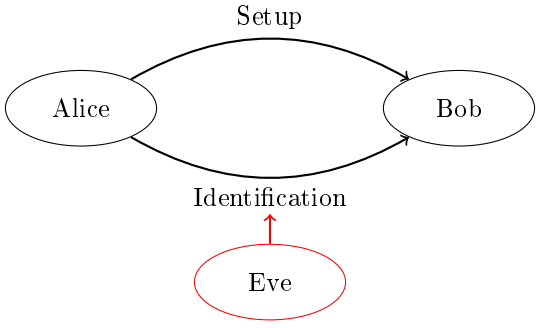


Figure 1: The single-guard problem

> "When you see a mathematical definition that attempts to model some real-life phenomenon such as security, you should pause and ask yourself: 1. Do I understand mathematically what the definition is stating? 2. Is it a reasonable way to capture the real life phenomenon we are discussing?".

If one tries to see whether the above requirements fit what we would like from, say, a login system, then the weakness of the above definition is that it makes the system susceptible to a "denial of service" (DOS) attack. According to the definition it is perfectly fine for Bob to shut down if Eve interferes.

A possible implementation of the protocol is that Alice and Bob agree on a predefined password $x \in_R \{0,1\}^\ell$ uniformly distributed. Then for a single guess of Eve the probability that she makes Bob accept is at most $2^{-\ell}$. Assuming that Bob shuts down after a single failed attempts, then Eve has exponentially low probability of making Bob accept. This allows, however, for Eve to mount a DOS attack by trying to guess passwords until Bob is forced to shut-down (without such a mechanism on Bob's side, Eve can enter given enough time).

It is worth to also consider, for example, the case in which Alice is a human choosing a password that is easy to remember. The password is now distributed with some distribution over $\Gamma \subseteq \{0,1\}^\ell$. A useful parameter to define the quality of the password is the *Shannon entropy* of the distribution p_X of possible passwords, defined as

$$H_1(X) = -\sum_{x \in \Gamma} p_X(x) \log_2 p_X(x) \,.$$

Note that the Shannon entropy is not sufficient to say that our probability distribution is good as a password: For example, for the probability distribution

$$p(x) = \begin{cases} \frac{1}{2} + 2^{-\ell-1} & x = 0^\ell \\ 2^{-\ell-1} & x \neq 0^\ell \end{cases}$$

the Shannon entropy would be high: $H_1 \approx \frac{\ell+2}{2}$, but this would not be a good password (an adversary which knows the probability distribution can crack the passwords with $O(1)$ probability). A better characterization is the *min entropy*

$$H_\infty(X) = \max_{x \in \Gamma} (-\log_2 p_X x)$$

which represents the most probable guess for a password. Since the best strategy Eve can have is to guess the password with maximal probability, her probability of cracking the password is $2^{-H_\infty(X)}$.

The two-guards problem

Suppose that we have two guards, and Eve has the information available to the guards at setup (say, Eve corrupted one of the guards, and Alice wants to pass through the other), see Fig. 2. Can we design a protocol that will prevent Eve from breaking?

The solution we mentioned in the previous lecture is to use a *one-way function* $f : \{0,1\}^n \to \{0,1\}^n$. At setup Alice remembers $x \in \{0,1\}^n$ and Bob knows the function f. During setup Alice computes $y = f(x)$ and gives y to Bob as a *public key*. The identification is done by Alice sending x and Bob checking that $y = f(x)$. Note that this protocol relies on the fact that Eve is not all-powerful, otherwise she could just invert f and obtain $x \in f^{-1}(y)$ given y. We recall the definition of a one-way function:

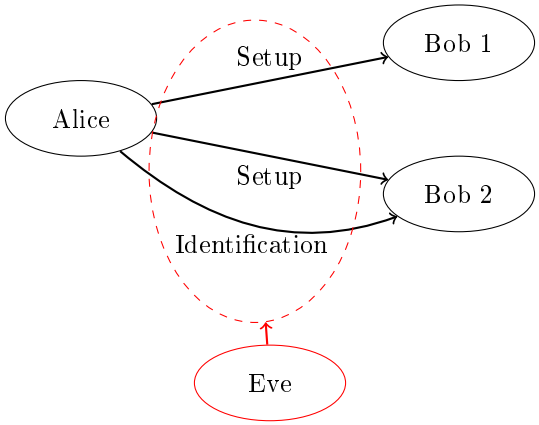


Figure 2: The two-guards problem

Definition (One-way function): $f : \{0,1\}^n \to \{0,1\}^n$ is a *one-way function* if f can be evaluated in polynomial time and for any probabilistic-polynomial Turing machine M and any polynomial $p(n)$ for large enough n

$$P_{x\in\{0,1\}^n, y=f(x)}\left[M\left(y=f(x)\right)\in f^{-1}(x)\right] \leq \frac{1}{p(n)}.$$

The probability is taken over both the distribution of passwords and the randomization of the algorithm.

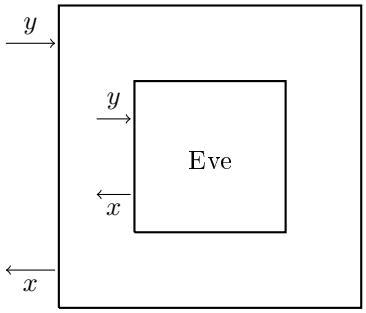
It is not clear that one-way functions exists (we rely on the $P \neq NP$ conjecture). We have, however, the following:

Theorem: The two-guards identification problem against a polynomial-time adversary has a solution if and only if one-way functions exist.

One direction is relatively easy: if one way functions exist, then the above protocol is a good one. To show this we use a reduction technique that will be prevalent throughout the course. We take an adversary that can break the protocol and use it to break the one-way function. Formally, suppose Eve can break protocol, that is she can take a public key y and output a guess x'. Eve wins if $f(x') = y$. Eve breaking the protocol means that her guess is correct w.p. $\geq 1/\text{poly}(n)$. Therefore, given the algorithm used by Eve we can construct a machine that inverts the one-way-function f (see Fig. 3). That is, given $y \in \{0,1\}^n$ we simply feed Eve with the same y and output the result of Eve's guess for $x' \in \{0,1\}^n$ which, with probability $1/\text{poly}(n)$ satisfies $f(x') = y$. This contradicts the assumption that f is a one-way function. Note that this reduction is simple and does not involve any deterioration in the time and probability of success.

For the other direction, we first need to control the possible kinds of protocols Alice and Bob can use. In general, Alice and Bob can use an interactive scheme during identification, and this is indeed what they will use in the multi-session case. However, we claim that, for this "single-session" protocol, for any interactive scheme we can construct a non-interactive scheme with similar completeness and soundness. The way we show this is by noticing that if Alice and Bob have an interactive scheme then Alice can send her state (as a Turing machine) and Bob can use it to simulate the interaction. Since both or them are polynomial time this gives the same completeness and soundness.

Now that we reduced the problem to a non-interactive scheme we can use the setup function as a one-way



One-way function inverter

Figure 3: Reduction from an efficient adversary of the two-guards problem protocol to an inversion of the one-way function f.

function. Suppose that for a given set of random bits $|r| = \text{poly}(n)$ used in the setup

$$\text{setup}(r) = (y, x).$$

We define the candidate one-way function as $f_{\text{setup}}(r) = \{\text{first output } y \text{ of the setup function}\}$.

Definition: We say that a protocol has *perfect completeness* if for any choice of the random coins during the setup and during the identification phase, if Alice follows the protocol and Eve does not interfere then Bob accepts.

Claim: With perfect completeness f is a one-way function.

Proof: We see that if f is not a one-way function, then Eve can obtain $r \in f^{-1}(y)$ with probability $1/\text{poly}(n)$. Eve can then generate x by applying setup (r). This entire process takes polynomial time assuming the setup takes polynomial time.

We note that f is not necessarily a one-way function in the case where we have less-than-perfect completeness. For example, assume that the setup is given two random strings $x_1, x_2 \in_R \{0,1\}^n$, such that the setup outputs x_2 if $x_1 = 0^n$, and otherwise outputs $f(x_2)$ for some one-way function. In this case given y, the function we generated by our setup scheme is not a one-way function since we can always invert to $r = (0^n, y)$. However the protocol is still valid: the probability of deceit is the probability of inverting the one-way function f plus 2^{-n}.

The way to remedy this problem is to add a single bit to the one-way function candidate output, which is one only if the setup is accepted, that is

$$\tilde{f}_{\text{setup}}(r) = (\{\text{first output } y \text{ of the setup function}\}, 1_{r \text{ is accepted}}).$$

In this case $\tilde{f}_{\text{setup}}(r) = (y, 1)$ can indeed be inverted only with negligible probability. For any randomized

polynomial-time Turing machine M we then have

$$P_{r\in\{0,1\}^n}\left(M\left(y=\tilde{f}_{\text{setup}}(r)\right)\in\tilde{f}^{-1}_{\text{setup}}(y)\right)\leq P_{r\in\{0,1\}^n}\left(M\left(y=\tilde{f}_{\text{setup}}(r)\right)\in\tilde{f}^{-1}_{\text{setup}}(y)\text{ and }r\text{ is accepted}\right)$$
$$+P\left(r\text{ not accpeted}\right)$$

The first term on the RHS is negligible by the assumption that no polynomial-time adversary can break the protocol, while the second is negligible by the assumption that Bob accepts except with negligible probability.

One additional complication we can add is the case in which Bob is randomized. This can also be remedied by running poly (n) instances of Bob and taking a cutoff of $2/3$ probability. In this case we can define the one-way function as $f(r)=\left(y,1_{r\text{ is accepded w.p.}\geq 2/3}\right)$. f can now be calculated in poly (n) steps by running setup (r) to obtain y and running Bob k times to obtain an estimate of the probability of acceptance. Since the probability that $1_{r\text{ is accepded w.p.}\geq 2/3}$ is estimated incorrectly decreases exponentially with k the required runs of Bob to obtain f with good probability is $O(1)$.

Finding one-way functions

Claim: If $P=NP$ then one-way functions do not exist.

The proof is based on a "search to decision" reduction. We consider the NP decision problem of given $y\in\{0,1\}^n$ and $x_1,...,x_i\in\{0,1\}$ are $x_{i+1},...,x_n\in\{0,1\}$ s.t. $f(x=(x_1,...,x_n))=y$. This problem is clearly in NP as if the answer is yes one can give x as a witness. If $P=NP$ we can decide for this answer in polynomial time. Therefore, for any given $y\in\{0,1\}^n$ one can work bit-by-bit, reconstructing x_1 by asking the decision problem with $i=1$, then work on for $i=2,3...$ Since each decision is made in polynomial time, the full retrieval of x is also done in polynomial time.

We see that to find one-way functions we need to consider problems that are conjecturally in $NP\backslash P$. Examples are:

- Subset sum: We have $0\leq a_1,...,a_n\leq 2^m-1$, and there is a target T. Is there a subset $s_1,...,s_n$ s.t.

$$\sum_{i,s_i=1}a_i=T\mod 2^m.$$

  The corresponding one-way function is

$$f(a_1,...,a_n,s_1,...,s_n)=(a_1,...,a_n,T).$$

- Discrete log, factoring, Goldreich's one-way function: next week.

Theorem (without proof): If one-way functions exist, for any NP-complete language L there are polynomial-time samplable distributions D_0,D_1, where given $x_R\in D_b$ it is hard to guess b and $x\in_R D_0\Rightarrow x\notin L, x\in_R D_1\Rightarrow x\in L$.

Levin's Universal one-way function

It is possible to find a function that is "complete" for one-way functions? The universal one-way function is a one-way function such that if there exists a one-way function then this function is a one-way function. We will see a construction, but one which is useless in practice.

We begin with the observation that if there exists a one-way function f calculable in $p(n)$ time for any polynomial p, there exists a one-way function f' running in n^2-time. This is done by defining

$$f'(x_1, x_2) = (f(x_1), x_2)$$

where $x_1 \in \{0,1\}^m, x_2 = \{0,1\}^{p(m)}$ and $n = p(m) + m$. f' can be calculated in quadratic time and is a one-way function.

To construct the universal one-way function we use a *combiner*. That is, for two one-way functions $f_1, f_2 : \{0,1\}^n \to \{0,1\}^n$ the combiner outputs f such that

$$f(x = (x_1, x_2)) = (f_1(x_1), f_2(x_2)) := f_1(x_1) || f_2(x_2).$$

Claim: If either f_1 or f_2 are one-way functions then so is f.

Proof: If we can invert f in polynomial time we can invert, e.g., f_1 in polynomial time by feeding the f-inverting machine with $(f_1(x_1), 0^n)$ and take the first n bits in $x \in f^{-1}(f_1(x_1), 0^n)$. Thus if f is not one-way then so are f_1, f_2.

The universal one-way function is then the combined version of the list of one-way functions listed according to the length of the Turing machine generating them. Let $M_1, ..., M_{\log n}$ be the list of the first $\log n$ Turing machines (in lexicographic order). Define

$$f(x = (x_1, ..., x_{\log n})) = M_1(x_1) || \cdots || M_{\log n}(x_{\log n})$$

for $x_i \in \{0,1\}^n$. $M_i(x_i)$ is defined to return the result of M_i on x_i after $2n^2$ steps. Then, assuming that one-way functions exist, one of $M_i(x)$ is one-way, and so f is one-way.

References

[1] Leonid A Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.