Cryptography and Security Protocols **IST-UL, 2024/2025**

A first look at cryptography beyond encryption

Lecturer: João Ribeiro

1 Introduction

So far we have been focusing on secret-key encryption, and this problem has also been the direct motivation behind the more basic objects we have introduced (PRGs and hardcore predicates). The goal of these brief notes is to show you that there is much more to cryptography besides encryption, and that the tools we have developed so far are useful for solving a multitude of problems. We will see how to solve many other important cryptographic problems using the basic toolset we developed later. We will also see cryptographic problems that we know probably cannot be solved using only these basic tools.

These notes are heavily based on Noah Stephens-Davidowitz's notes for his cryptography course at Cornell. You should definitely go and read those.

2 Coin tossing over the telephone

Alice and Bob are talking over the phone and wish to toss a coin. Alice prefers "heads" (i.e., outcome $b = 0$) and Bob prefers "tails" (i.e., outcome $b = 1$). Of course, they could just have Alice toss a coin on her end and report the result. However, Bob does not trust Alice! How can Alice and Bob toss a fair coin without trusting each other?

This is a classical problem in cryptography, first studied by Blum in the 80s [Blu83]. Besides being a beautifully simple problem, it is also very relevant in practice – randomness is central to cryptography. For example, blockchains (and other real-world systems) rely on randomness beacons (this could be a nice topic for a course project). These beacons output public random bits at regular intervals – they must be unbiasable by coalitions of dishonest participants in the randomness generation protocol, and their output must be the same for every participant and verifiable. Designing fast, scalable, and secure randomness beacons is a challenging problem, both in theory and in practice. We will not formally define any of these concepts and the desired security guarantees at this stage.

Here is a first try at a coin tossing protocol:

1. Alice samples $b_A \leftarrow \{0,1\}$ and sends b_A to Bob.
2. Bob samples $b_B \leftarrow \{0,1\}$ and sends b_B to Alice.
3. They compute the coin toss outcome $b = b_A \oplus b_B = b_A + b_B \pmod 2$.

This protocol is not secure! For example, Bob could be dishonest and, instead of sampling b_B uniformly at random and independently of b_A, could instead set $b_B = b_A \oplus 1$. This would guarantee that $b = b_A \oplus b_B = b_A \oplus b_A \oplus 1 = 1$ always!

We will solve this problem using hardcore predicates! Suppose that $f\colon \{0,1\}^* \to \{0,1\}^*$ is an efficiently computable permutation and P is a hardcore predicate for f. Here is a new protocol, parameterized by a security parameter n:

1. Alice samples $b_A \leftarrow \{0,1\}$ and $x \leftarrow \{0,1\}^n$. Then, Alice sends
$$(y, z) = (f(x), P(x) \oplus b_A)$$
to Bob.
2. Bob samples $b_B \leftarrow \{0,1\}$ and sends b_B to Alice.
3. Alice sends (b_A, x) to Bob.
4. Bob checks whether $f(x) = y$. If so, Bob computes $b_A = P(x) \oplus z$ and they both obtain the coin outcome $b = b_A \oplus b_B$. Otherwise, Bob aborts the protocol.

This protocol produces a coin so long as $f(x) = y$ in Step 4. The only way this check can fail is if Alice is dishonest.[1] Let's try to informally understand why this protocol works. A first claim is that after Step 1 Bob learns nothing about Alice's bit b_A. Therefore, Bob must sample b_B without knowledge of b_A, i.e., b_B is independent of b_A. Second, in Steps 3 and 4 Alice cannot make Bob compute a different $b'_A \neq b_A$, which could be useful given that Alice now knows b_B and could try to bias the coin her way. In fact, if Alice sends $x' \neq x$ to Bob, then $f(x') \neq f(x)$ because f is a permutation, and so Bob will catch Alice cheating and will abort the protocol.

3 Commitment schemes

In some sense, the protocol above forces Alice to *commit* to her coin b_A, while not revealing any information to Bob before Alice *opens* her commitment by revealing x. Although we will not formally define security notions nor formally prove the security of the protocol above, we will now formally define this notion of a *bit commitment scheme*, and formally prove that the ideas above indeed yield a bit commitment scheme.

Definition 1 (Commitment scheme) *A function* $\mathsf{Com}\colon \{0,1\} \times \{0,1\}^* \to \{0,1\}^*$ *is a* bit commitment scheme *if the following properties hold:*

- *There exists a deterministic polynomial-time algorithm that given a bit* $b \in \{0,1\}$ *and randomness* $r \in \{0,1\}^*$ *outputs* $\mathsf{Com}(b, r)$.

[1] In fact, we cannot really hope for better output guarantees in this setting, unless we, e.g., relax the notion of "fairness" of the protocol (this has been explored in a recent fascinating line of research about rational cryptography).

- ***Perfect binding:*** *For any distinct $b, b' \in \{0,1\}$ and any $r, r' \in \{0,1\}^*$ we have $\mathsf{Com}(b,r) \neq \mathsf{Com}(b',r')$.*
- ***Computational hiding:*** *For any PPT adversary $\mathcal{A}$ there is a negligible function $\varepsilon(n)$ such that*
$$\Pr_{b \leftarrow \{0,1\}, r \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, \mathsf{Com}(b,r)) = b] \leq \frac{1}{2} + \varepsilon(n).$$

A bit commitment scheme should be used as follows. To commit to a bit b, Alice samples $r \leftarrow \{0,1\}^n$ and computes the commitment $w = \mathsf{Com}(b,r)$. Now, suppose that Bob holds w and Alice wants to prove that she indeed committed to b. To "reveal/open" the commitment, Alice sends (b,r) to Bob, and Bob checks that $w = \mathsf{Com}(b,r)$.

As already hinted in the previous section, we can construct perfectly binding and computationally hiding bit commitment schemes from permutations with hardcore predicates. We prove this now.

Theorem 1 *Suppose that there exists an efficiently computable permutation f and a hardcore predicate P for f. Then, there exists a perfectly binding and computationally hiding bit commitment scheme.*

Proof: We consider $\mathsf{Com}(b,r) = (f(r), P(r) \oplus b)$. Perfect binding is easy to show. Consider arbitrary $b' \neq b$ and $r, r' \in \{0,1\}^n$. Then, either $r \neq r'$, in which case $f(r) \neq f(r')$ and so $\mathsf{Com}(b,r) \neq \mathsf{Com}(b',r')$, or $r = r'$, meaning that $P(r) \oplus b \neq P(r) \oplus b' = P(r') \oplus b'$, and so $\mathsf{Com}(b,r) \neq \mathsf{Com}(b',r')$.

It remains to establish computational hiding. Suppose that Com is not computationally hiding. This means that there exists a PPT adversary $\mathcal{A}$ and a polynomial $p(n)$ such that
$$\Pr_{b \leftarrow \{0,1\}, r \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, \mathsf{Com}(b,r)) = b] \geq \frac{1}{2} + \frac{1}{p(n)}.$$
We will use $\mathcal{A}$ to design a PPT adversary $\mathcal{A}'$ that predicts the hardcore predicate P. Given $f(r)$ for $r \leftarrow \{0,1\}^n$, the adversary $\mathcal{A}'$ behaves as follows:

1. Samples $a \leftarrow \{0,1\}$.
2. Computes $b' \leftarrow \mathcal{A}(1^n, (f(r), a))$.
3. Outputs $b^* = a \oplus b'$.

We now lower bound the probability that $\mathcal{A}'$ guesses $P(r)$. We have
$$\begin{aligned}
\Pr_{r \leftarrow \{0,1\}^n}[\mathcal{A}'(1^n, f(r)) = P(r)] &= \Pr_{a \leftarrow \{0,1\}, r \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, (f(r), a)) = P(r) \oplus a] \\
&= \Pr_{b \leftarrow \{0,1\}, r \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, (f(r), b \oplus P(r))) = b] \\
&\geq \frac{1}{2} + \frac{1}{p(n)},
\end{aligned}$$
contradicting the fact that P is a hardcore predicate for f. The second equality uses the fact that a and r are sampled independently. This means that $P(r) \oplus a$ is also a uniformly random bit independent of r. We can then define $b = P(r) \oplus a$ and we can write $a = b \oplus P(r)$. ∎

References

[Blu83] Manuel Blum. Coin flipping by telephone: A protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, January 1983. Preliminary version in CRYPTO 1981.