Cryptographic Protocols - Lecture Notes

Yanis Berger

Autumn 2025

Contents

1 Lecture 1 **2**

1.1 Overview of what the course will cover . . . 2

1.2 Example: Generate a random bit (coin flip) . . . 2

1.3 Ex: Secret vote among three . . . 4

2 Lecture 2 **6**

2.1 Basic technique . . . 6

2.1.1 Programs as circuits . . . 6

2.1.2 Circuit for testing if two numbers are equal . . . 6

2.2 Mathematics of public key Cryptography . . . 7

2.3 Public Key Encryption . . . 8

2.4 ElGamal public-key encryption . . . 9

1 Lecture 1

1.1 Overview of what the course will cover

- Computing with encrypted data
- AuthN without giving away any secret
- E-voting / cryptographic voting protocols
- Blockchain is transparent, cannot keep secrets
- Generate a true random & unbiased value
- Sealed bid auction w/o trusted entity

1.2 Example: Generate a random bit (coin flip)

Needs cryptography to ensure that the output is not biased among two parties A and B, since the party that sends the last message can precompute/bias the output

Use cryptography: Simultaneous commitments

- hash the message, send it
- We add randomness as hashing bit
- hash functions are collision-free: No two inputs give the same output
- output gives no info about the input

Hash func H:

$$H : \{0,1\}^* \to \{0,1\}^k$$

SHA-2: $k = 256$ as a special case of a commitment scheme

Protocol

<u>A</u>		<u>B</u>
$a \overset{\mathbb{R}}{\leftarrow} \{0,1\}$		$b \overset{\mathbb{R}}{\leftarrow} \{0,1\}$
$x \overset{\mathbb{R}}{\leftarrow} \{0,1\}^k$		$y \overset{\mathbb{R}}{\leftarrow} \{0,1\}^k$
$c \leftarrow H(a\|x)$		$d \leftarrow H(b\|y)$
	$\xrightarrow{\qquad c \qquad}$	
	$\xleftarrow{\qquad d \qquad}$	
	$\xrightarrow{\qquad a,x \qquad}$	
	$\xleftarrow{\qquad b,y \qquad}$	
verify $d \overset{?}{=} H(b\|y)$		verify $c \overset{?}{=} H(a\|x)$
<u>output $a \oplus b$</u>		<u>output</u> $a \oplus b$

$\tilde{b}$, $\tilde{y} \Rightarrow$ not possible as this would violate collision-freeness

- c and d hide a & b, because this hash function hides its input.
- A & B cannot change their inputs because H is collision free

Ex: Millionaires' Problem

A and B want to find out who is richer without disclosing their wealth

A(x)		B(x)
	$\xrightarrow{\qquad\qquad}$	
	$\xleftarrow{\qquad\qquad}$	
	$\xrightarrow{\qquad\qquad}$	
	$\xleftarrow{\quad \dots \quad}$	
output $x \overset{?}{\geq} y$		output $x \overset{?}{\geq} y$

This is easy with a trusted entity T:

$$A \longrightarrow \boxed{T} \longleftarrow B$$

$$b \leftarrow (x \overset{?}{\geq} y)$$

$$\xleftarrow{\quad b \quad} \qquad \xrightarrow{\quad b \quad}$$

We want to hide as much as we can
Important example for:

- Auctions
- Matchings
- Elections $\longrightarrow$ solution comes later

Ex: Computing with encrypted data

$A(x)$	$B(f)$
user	cloud
input x (data)	function f
Encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with keypair (pk, sk)	
$c_x \leftarrow \mathsf{Enc}(pk, x)$	
Send pk, c_x $\longrightarrow$	
	$c_y \leftarrow \text{Eval}(pk, f, c_x)$
	Eval "runs" program f on data x
	encrypt inside c_x
$\longleftarrow c_y$	
$y \leftarrow \mathsf{Dec}(sk, c_y)$	
s.t. $\quad y = \mathsf{Dec}(sk, \text{Eval}(pk, f, \mathsf{Enc}(pk, x))) = f(x)$	

Why is computing on secret data difficult?

Layers

App
VMs
Containers
OS
Hypervisor
Base OS
Hardware

high: Trusted Platform Module (TPM)
TCB Trusted Computer Base

lower levels can always see everythin that happens on higher levels/above them
TCB controls everything

1.3 Ex: Secret vote among three

Parties p_1, p_2, p_3

Each has a binary vote v_i as input. Goal is to compute the sum s of the votes and not disclose any more information than that.

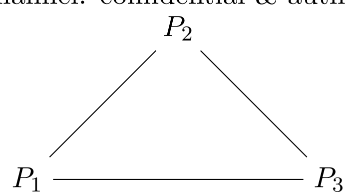
Protocol: Additive secret sharing

Primitive: $\text{split}(b) \to (x_1, x_2, x_3)$ to distribute or "share" b
Use prime p

$\underline{\text{split}(p)}$: $\{0, 1, 2, ..., p-1\}$

$x_1 \leftarrow \mathbb{Z}_p$
$x_2 \leftarrow \mathbb{Z}_p$
$x_3 \leftarrow \mathbb{Z}_p$ s.t. $x_1 + x_2 + x_3 \equiv b \pmod{p}$
return (x_1, x_2, x_3)

secure channel: confidential & authenticated



Party $p_i(v_i)$:
$(x_{i1}, x_{i2}, x_{i3}) \leftarrow \text{split}(v_i)$
<u>send</u> x_{ij} to p_j for $j \in \{1, 2, 3\}$
<u>receive</u> x_{ji} from p_j
$y_i \leftarrow (x_{1i} + x_{2i} + x_{3i}) \bmod p$
<u>send</u> y_i to p_j for $j \neq i, ..., 3$
<u>receive</u> y_j from p_j
<u>output</u> $(y_1 + y_2 + y_3) \bmod p$
where $(y_1 + y_2 + y_3) =$ s

Completeness

claim: $s \equiv v_1 + v_2 + v_3$

Proof:

$$
\begin{aligned}
s &= \sum_{i=1}^{3} y_i \\
&= \sum_{i=1}^{3} \left(\sum_{j=1}^{3} x_{ji} \right) \\
&= \sum_{j=1}^{3} \left(\sum_{i=1}^{3} x_{ji} \right) && \text{(columns sum)} \\
&= \sum_{j=1}^{3} (x_{j1} + x_{j2} + x_{j3}) && \text{(rows sum)} \\
& && v_j \text{ to split()} \\
&= \sum_{j=1}^{3} v_j \pmod p
\end{aligned}
$$

Security

$\text{split}(b) \to (x_1, x_2, x_3)$

No two values of x_i, x_j give information about b

(Generation of a one time pad)

Output s reveals nothing more than it should: nothing about v_j for $j \neq i$; only s.

Goals

Privacy: No party learns more than it should (the output)
Correctness: Every party receives the output as specified by the function they are evaluating
Input Independence: Inputs of corrupted or faulty parties do not depend on input of correct parties.
Fairness: All parties either receive an output or no party receives an output
$\longrightarrow$ faulty parties receive outputs if and only if correct parties receive output

Faults

All faulty parties are modeled as controlled (or corrupted) by one adversary $\mathcal{A}$.
semi-honest behaviour

- Corrupted parties follow protocol
- Leak all data to $\mathcal{A}$
- "passive attack", "passive adversary", "read-only attack"

Malicious behaviour

- Faulty parties behave arbitrarily, controlled by $\mathcal{A}$

2 Lecture 2

Consider Programs formulated as circuits.

- Finite state automata (Turing machines)
- Circuits
- stack automata

What can be computed on one can be computed on all
$\Longrightarrow$ For this course, consider circuits.

2.1 Basic technique

2.1.1 Programs as circuits

Every program on inputs $x_1, ..., x_n$ computes a function $f(x_1, ..., x_n)$, represented by a Turing machine or by a circuit.

Deterministic vs. Non-Deterministic Turing Machine:

TODO: add this

Polynomial-time: $O(n^k) \approx$ Polynomial time, so the run time depends on the input size in a polynomial relation.

Cook-Levin Theorem: $NP \stackrel{?}{=} P$

Every problem decided by a non-deterministic Turing Machine (= NP-problem) in Polynomial time can be formulated as the satisfiability problem (SAT) of a polynomial-size (boolean) circuit.

$\Longrightarrow$ One time use circuits, not circuits with clocks.

In every computer the CPU represents computations as stateful circuits.

Here we will represent any computation as a circuit and evaluate the circuit in “encrypted” form.

2.1.2 Circuit for testing if two numbers are equal

Given two numbers in binary.
$(x)_2 = (x_{n-1}, ..., x_0)$
$(y)_2 = (y_{n-1}, ..., y_0)$

To compute if $x \stackrel{?}{=} y$

```
i ← n
while i > 0 do
    i ← i−1
    if x_i ≠ y_i then
        return FALSE
return TRUE
```

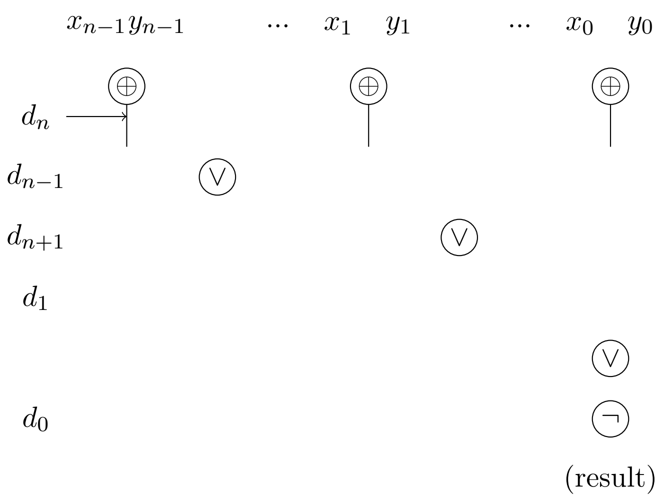
In the circuit one has to unroll all loops and eliminate imperative operations.

Circuit:

```
i ← n
d_n ← 0
while i > 0 do
```

`i ← i-1`
$d_i \leftarrow d_{i+1} \vee (x_i \oplus y_i)$
`return` $\neg d_0$

Circuit: $O(n)$ complexity



2.2 Mathematics of public key Cryptography

Modular Arithmetic: "computing modulo m"

$x \leftarrow a \bmod m$ (operation)
$x \equiv a \pmod{m}$ (Relation, can be true/false)
$\mathbb{Z}_m = \{0, ..., m-1\}$ with addition modulo m
$\mathbb{Z}_m^* = \{1, ..., m-1\}$ with multiplication modulo m, typically m is p, prime.

Cyclic groups

A cyclic group G has a generator $g \in G$ s.t. every element of G is obtained by computing g^i for $i = 0, 1, ...$

$$g^0 = 1$$

G is finite and has $|G|$ elements: $G = \{1, g, g^2, ..., g^{|G|-1}\}$
because $g^{|G|} = g^0 = 1$
write $\langle g \rangle = G$ (generator)

- $\mathbb{Z}_m$ with addition operation is cyclic with $g = 1$
- $\mathbb{Z}_p^*$ with multiplication is cyclic with $|\mathbb{Z}_p^*| = p - 1$

Ex: $\mathbb{Z}_{11}^*$
$\langle 2 \rangle_{11} = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$
$\langle 3 \rangle_{11} = \{1, 3, 9, 5, 4, ...\}$ only a subgroup

if $q \mid p-1$ is also prime then there is a cyclic subgroup with q elements, defined by multiplication modulo $p \Rightarrow$ used in cryptography

with $|p|$ about 2000 bit
and $|q|$ about 256 bit such that $p = nq + 1$
or such that $p = 2q + 1$
$(|q| = |p| - 1)$

Discrete logarithms

$G = \langle g \rangle$
the discrete log of $a \in G$ with respect to (w.r.t.) base g is the $l \in \mathbb{Z}_{|G|-1}$ s.t.

$$g^l = a$$

computing discrete logarithms is assumed to be hard (in some G) (computationally hard)
DLP: Discrete Logarithm Problem
Given y, where $\begin{cases} r \leftarrow \mathbb{Z}_q \\ y \leftarrow g^r \end{cases}$ with public info or from key
Find r.
Related: Computational Diffie-Hellman Problem
Given x and y, where
$a \xleftarrow{\mathbb{R}} \mathbb{Z}_q$ $b \xleftarrow{\mathbb{R}} \mathbb{Z}_q$
$x \leftarrow g^a$ $y \leftarrow g^b$
Find g^{ab} $[\in G]$

DDHP: Decisional DH Problem

Given either

1. (x, y, z), where
   $a \leftarrow \mathbb{Z}_q$ $y \leftarrow g^a$
   $b \leftarrow \mathbb{Z}_q$ $x \leftarrow g^b$
   $c \leftarrow \mathbb{Z}_q$ $z \leftarrow g^c$

or

2. (x, y, z), where
   $a \leftarrow \mathbb{Z}_q$ $y \leftarrow g^a$
   $b \leftarrow \mathbb{Z}_q$ $x \leftarrow g^b$
   $z \leftarrow g^{ab}$

Task is not compute z, but instead which Tripel 1 or 2 we have

2.3 Public Key Encryption

$\mathsf{KeyGen}() \to (pk, sk)$
$\mathsf{Enc}(pk, m) \to c$
$\mathsf{Dec}(sk, c) \to m$

Completeness

$\forall m : (pk, sk) \leftarrow \mathsf{KeyGen}()$
$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$

Security

- An encryption of some message m is indistinguishable from a random element of the ciphertext space.
- For two messages m_1 and m_2, no efficient adversary can distinguish $\mathsf{Enc}(pk, m_1)$ from $\mathsf{Enc}(pk, m_2)$ except with negligible probability.

$\Longrightarrow$ $\mathsf{Enc}(pk, m_1)$ is indistinguishable from $\mathsf{Enc}(pk, m_2)$
Security parameter: λ
$f(\cdot)$ is negligible if

$$\exists \lambda_0 : \forall c > 0 : f(\lambda) < \frac{1}{\lambda^c} \text{ for } \lambda \geq \lambda_0$$

2.4 ElGamal public-key encryption

- Textbook version
- Cyclic Group: $G = \langle g \rangle$, $|G| = q$

$\underline{\mathsf{KeyGen}()}$
$x \leftarrow \mathbb{Z}_q$
$y \leftarrow g^x$
$\underline{\text{return}}$ (y, x) $\qquad y : pk,\ x : sk$

$\underline{\mathsf{Enc}(y, m)}$ $\qquad m \in G$
$r \xleftarrow{R} \mathbb{Z}_q$
$R \leftarrow g^r$
$c \leftarrow y^r \cdot m$
$\underline{\text{return}}$ $(R, c) \in G \times G$

$\underline{\mathsf{Dec}(x, (R, c))}$
$\hat{m} \leftarrow c / R^x$
$\underline{\text{return}}$ $\hat{m}$

$$\hat{m} = c/R^x = (y^r \cdot m) \cdot (g^{-r})^x = g^{xr} \cdot m \cdot g^{-rx} = m$$

Security

Decide whether for some m, (y, R, c) is $(g^x, g^r, m \cdot g^{xr})$ (DH exponent, independent) or (y, R, c) is $(g^x, g^r, m \cdot g^z)$ for some $z \xleftarrow{\$} \mathbb{Z}_p$
This is equivalent to the DDHP

Digital Signature (DS) Scheme is pub key cryptography