Cryptographic Protocols - Lecture Notes

Yanis Berger

Autumn 2025

Contents

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 mashing 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

A		B
$a \xleftarrow{\mathbb{R}} \{0,1\}$		$b \xleftarrow{\mathbb{R}} \{0,1\}$
$x \xleftarrow{\mathbb{R}} \{0,1\}^k$		$y \xleftarrow{\mathbb{R}} \{0,1\}^k$
$c \leftarrow H(a\|x)$		$d \leftarrow H(b\|y)$
	$\xrightarrow{\quad c \quad}$	
	$\xleftarrow{\quad d \quad}$	
	$\xrightarrow{\quad a,x \quad}$	
	$\xleftarrow{\quad b,y \quad}$	
verify $d \stackrel{?}{=} H(b\|y)$		verify $c \stackrel{?}{=} H(a\|x)$
output $a \oplus b$		output $a \oplus b$

$\tilde{b}$, $\tilde{y} \Rightarrow$ 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

$\longrightarrow$ Find out how exactly this gives a random bit

1.3 Example: Millionaires' Problem

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

$A(x)$ $B(y)$

output $x \geq y$ output $x \geq y$

This is easy with a trusted entity T.

$$A \longrightarrow |T| \longleftarrow B$$
$$b \leftarrow (x \geq y)$$

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 $\longrightarrow$

$c_y \leftarrow \mathrm{Eval}(pk, f, c_x)$

Eval "runs" program f on data x

encrypt inside c_x

$\longleftarrow c_y$

$y \leftarrow \mathsf{Dec}(sk, c_y)$

s.t. $y = \mathsf{Dec}(sk, \mathrm{Eval}(pk, f, \mathsf{Enc}(pk, x))) = f(x)$

Why is computing on secret data difficult?

Layers

App
VMs
Containers
OS
Hypervisor
Base OS
Hardware

high TCB
(Trusted Computer Base)

see ↑ lower levels can see everything above them
can always see what happens on higher levels
⟶ cc: lower levels can see everything above them
TCB controls everything

1.4 Ex: Secret vote among three

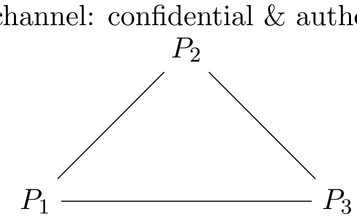
Parties p_1, p_2, p_3

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: split(b) $\rightarrow (x_1, x_2, x_3)$ to distribute or "share" b
Use prime p
split(b): $\{0, 1, 2, ..., p-1\}$
$x_1 \xleftarrow{\$} \mathbb{Z}_p$
$x_2 \xleftarrow{\$} \mathbb{Z}_p$
$x_3 \xleftarrow{\$} \mathbb{Z}_p$ s.t. $x_1 + x_2 + x_3 \equiv b \pmod p$
return (x_1, x_2, x_3)

secure channel: confidential & authenticated



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

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

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 $\mathcal{A}$