Agenda

5. Trends in Cryptography
   - Computation over Private Data
   - Post-Quantum Cryptography

Computation over private data (CoPD)

Idea: Perform operations over private data without revealing information.

More formally: Given private data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and a function f, compute $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ without leaking information about $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

- Generate non-leakable data $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ from $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

$$(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

- Design a cryptographic primitive Π such that

$$\Pi(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

- One can recover $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ using only non-leakable data $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ by executing $\Pi(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

Computation over Private Data (CoPD)

$\textcolor{teal}{\mathbf{x}} = (\textcolor{teal}{\mathbf{x}_1, \ldots, \mathbf{x}_n})$: Initial data

$\textcolor{red}{\mathbf{x}} = (\textcolor{red}{\mathbf{x}_1, \ldots, \mathbf{x}_n})$: Non-leakable data

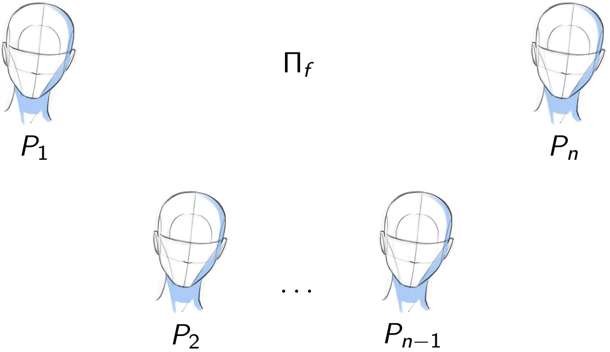
$f(\textcolor{teal}{\mathbf{x}_1, \ldots, \mathbf{x}_n})$: Expected computation

$$
\begin{tikzcd}
\textcolor{teal}{\mathbf{x}} \arrow[rr, dotted, "f"] \arrow[drr, "\Phi"'] & & f(\textcolor{teal}{\mathbf{x}}) \\
& & \Phi(\textcolor{teal}{\mathbf{x}}) = \textcolor{red}{\mathbf{x}} \arrow[u, "\Pi(\textcolor{red}{\mathbf{x}})"']
\end{tikzcd}
$$

Warning: No privacy garantee if $f(\textcolor{teal}{\mathbf{x}_1, \ldots, \mathbf{x}_n})$ leaks information on $\textcolor{teal}{\mathbf{x}_1, \ldots, \mathbf{x}_n}$

Multi-Party Computation (MPC)

Idea: Suppose there is $\mathbf{n}$ *parties* $\mathbf{P}_i$ *each getting input* $\mathbf{x}_i$*. At the end of the protocol, each party should only knows its input* $\mathbf{x}_i$ *and* $y = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$*.*

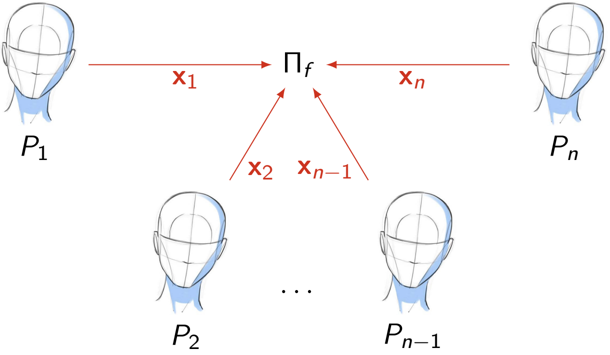


Applications: Secure comparison, set intersection, auctions...

img src: `https://drawingref.com/head/`

Multi-Party Computation (MPC)

Idea: Suppose there is $\mathbf{n}$ *parties* $\mathbf{P}_i$ *each getting input* $\mathbf{x}_i$*. At the end of the protocol, each party should only knows its input* $\mathbf{x}_i$ *and* $y = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$*.*

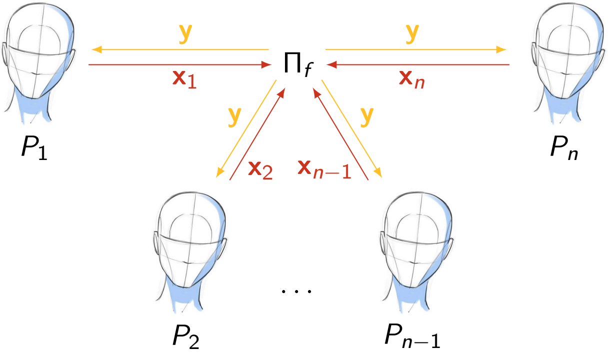


Applications: Secure comparison, set intersection, auctions...

img src: `https://drawingref.com/head/`

Multi-Party Computation (MPC)

Idea: Suppose there is $\mathbf{n}$ *parties* $\mathbf{P}_i$ *each getting input* $\mathbf{x}_i$*. At the end of the protocol, each party should only knows its input* $\mathbf{x}_i$ *and* $y = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$*.*



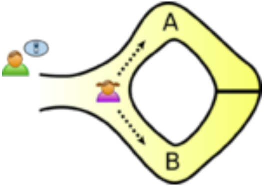
Applications: Secure comparison, set intersection, auctions...

img src: `https://drawingref.com/head/`

Zero Knowledge (ZK)

Idea: Proving the knowledge of some information **x** *without leaking any other information about it*

Example: The cave and the magic door.



The prover get throught the door with probability $\frac{1}{2}$. Repeat the process.

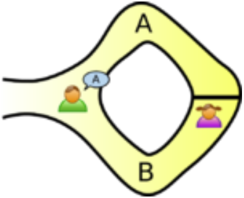
Applications: Authentication, signature, cryptocurrencies...

img src: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`

Zero Knowledge (ZK)

Idea: Proving the knowledge of some information **x** *without leaking any other information about it*

Example: The cave and the magic door.



The prover get throught the door with probability $\frac{1}{2}$. Repeat the process.

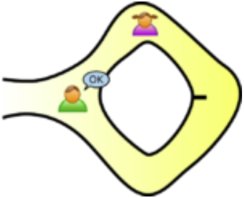
Applications: Authentication, signature, cryptocurrencies...

img src: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`

Zero Knowledge (ZK)

Idea: Proving the knowledge of some information **x** *without leaking any other information about it*

Example: The cave and the magic door.



The prover get throught the door with probability $\frac{1}{2}$. Repeat the process.

Applications: Authentication, signature, cryptocurrencies...

img src: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`

Zero Knowledge (ZK)

Idea: Proving the knowledge of some information **x** *without leaking any other information about it*

Challenge: Pick a card at random. How to prove its color (club, diamond, spade or heart) in zero-knowledge?

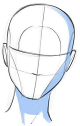
Applications: Authentication, signature, cryptocurrencies...

img src: `https://commons.wikimedia.org/wiki/File:Pick_a_card.jpg`

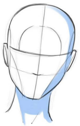
Example: Schnorr's signature

$\mathbb{G}$ a group with generator g

Goal: Prove the knowledge of x such that $h = g^x \in \mathbb{G}$



Prover(x,h)



Verifier(h)

Prover(x,h)		Verifier(h)
$r \xleftarrow{\$} \mathbb{Z}_q$		
$u \leftarrow g^r$	$\xrightarrow{\quad u \quad}$	$c \xleftarrow{\$} \mathbb{Z}_q$
$z \leftarrow r + cx$	$\xleftarrow{\quad c \quad}$	
	$\xrightarrow{\quad z \quad}$	$g^z \stackrel{?}{=} u \cdot h^c$

Completness: $g^z = g^{r+cx}$ and $u \cdot h^c = g^r \cdot (g^x)^c = g^{r+cx}$

HVZK of Schnorr's signature

Goal: Construction of a simulator able to output values indistinguishable from a legit signature and without knowledge of x

$\mathcal{D}_1 = \{(g^r, c, r + cx) : r, c \xleftarrow{\$} \mathbb{Z}_q\}$

$\mathcal{D}_2 = \{(u, c, z) : c, z \xleftarrow{\$} \mathbb{Z}_q,\ g^z = u \cdot h^c\}$

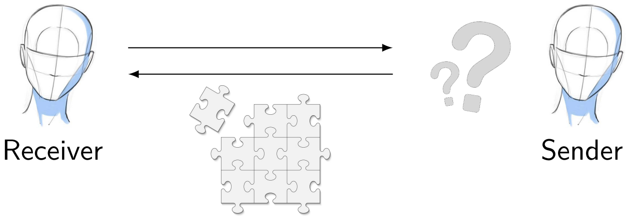
$$S(h): \quad 1.\ z \xleftarrow{\$} \mathbb{Z}_q \qquad 2.\ c \xleftarrow{\$} \mathbb{Z}_q$$
$$3.\ u \leftarrow \frac{g^z}{h^c} \qquad 4.\ output(u, c, z)$$

z random imply u random and output $\mathcal{D}_2$ identically distributed as $\mathcal{D}_1$

Why HVZK and not ZK? Because a malicious verifier could choose c adaptively (not randomly)

Oblivious Transfert (OT)

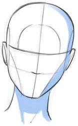
Idea: Protocol in which a sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred.



First form of oblivious transfert was introduced in 1981 by Michael O. Rabin

img src: `https://www.vectorstock.com/royalty-free-vectors/puzzle-schema-vectors`

Oblivious Transfert (OT)

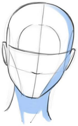


Receiver

$\{m_0, m_1\}$

$\mathbb{G}$ a cyclic group

g a generator of $\mathbb{G}$



Sender

Chooses $\sigma \in \{0, 1\}$

$h_0 \xleftarrow{\$} \mathbb{G}$

$a_1 \in \mathbb{Z}_q$ and $h_1 = g^{a_1}$

$$\xleftarrow{\{h_0, h_1\}}$$

$r \xleftarrow{\$} \mathbb{Z}_q$

$u = g^r$

$v_0 = {h_0}^r \cdot m_0$

$v_1 = {h_1}^r \cdot m_1$

$$\xrightarrow{\{u, v_0, v_1\}}$$

$u^{a_1} = (g^r)^{a_1} = {h_1}^r$

$\frac{v_1}{u^{a_1}} = \frac{{h_1}^r \cdot m_1}{{h_1}^r} = m_1$

Fully Homomorphic Encryption (FHE)

Idea: Fully Homomorphic Encryption allows operations directly over the encrypted values

$$\Pi(x_1) + \Pi(x_2) = \Pi_+(x_1 + x_2)$$
$$\Pi(x_1) \times \Pi(x_2) = \Pi_\times(x_1 \times x_2)$$

$$\begin{tikzcd}
x_1, x_2 \arrow[r, dotted] \arrow[d, "Enc"'] & x_1 + x_2 \\
x_1, x_2 \arrow[r, "\Pi_+"'] & x_1 + x_2 \arrow[u, dotted, "Dec"']
\end{tikzcd}
\qquad
\begin{tikzcd}
x_1, x_2 \arrow[r, dotted] \arrow[d, "Enc"'] & x_1 \times x_2 \\
x_1, x_2 \arrow[r, "\Pi_\times"'] & x_1 \times x_2 \arrow[u, dotted, "Dec"']
\end{tikzcd}$$

Partially Homomorphic Encryption

A partially homomorphic encryption scheme allows only addition or multiplication of ciphertexts

Example: The ElGamal encryption scheme

- Public key encryption scheme
- Based on Diffie-Hellman key exchange
- Described by Taher ElGamal in 1985
- Used in GPG and PGP

Partially Homomorphic Encryption

ElGamal encryption scheme:

Ambient space: $\mathbb{G}$ a cyclic group of order q g a generator of $\mathbb{G}$ A message M	**Key Generation**: $x \xleftarrow{\$} [\![1, q-1]\!]$ $h = g^x$ $pk : h$ $sk : x$
Encryption(M, pk) Map M to $m \in \mathbb{G}$ $y \xleftarrow{\$} [\![1, q-1]\!]$ $s = h^y$ (shared secret) $(c_1 = g^y, c_2 = m \cdot s)$	**Decryption(c_1, c_2, sk)** ${c_1}^x = g^{xy} = h^y = s$ Calculates s^{-1} $c_2 \cdot s^{-1} = m \cdot s \cdot s^{-1} = m$ Recover M from m

Partially Homomorphic Encryption

ElGamal encryption compatibility with multiplication:

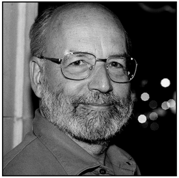
1^{st} ciphertext: $(c_{11} = g^{y_1}, c_{12} = m_1 \cdot s_1)$ with $s_1 = h^{y_1}$
2^{nd} ciphertext: $(c_{21} = g^{y_2}, c_{22} = m_2 \cdot s_2)$ with $s_2 = h^{y_2}$

$$c_{11} \cdot c_{21} = g^{y_1 + y_2}$$

$$\begin{aligned} c_{12} \cdot c_{22} &= m_1 \cdot s_1 \cdot m_2 \cdot s_2 \\ &= m_1 \cdot m_2 \cdot h^{y_1} \cdot h^{y_2} \\ &= m_1 m_2 \cdot h^{y_1 + y_2} \end{aligned}$$

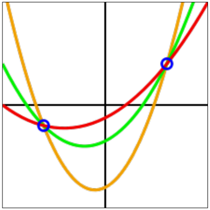
$(c_{11} c_{21}, c_{12} c_{22})$ is a ciphertext of $m_1 m_2$

Secret Sharing



Shamir's secret sharing idea: Two points are sufficient to define a line, three to define a parabola, four to define a cubic curve and so forth...

- **Secret**: Intersection between a polynomial P of degree $k-1$ and the ordinate axis
- **Shares**: n different points of the polynomials



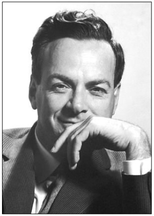
- Knowledge of k points let you reconstruct P by interpolation
- Given $k-1$ different points, every point in the ordinate axis could still be the secret

Introduction to quantum computing

- **Quantum mechanics**: Branch of physics that studies natural phenomena occuring at atomic scale
- **Quantum computing**: Study of computation systems that make use of quantum mechanical phenomena
- Introduced by Richard Feynman in 1982

"Anyone who claims to understand quantum theory is either lying or crazy"

Richard Feynman



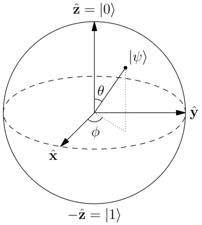
Post-quantum vs quantum cryptography

- **Cryptography**: Cryptographic algorithms over classical computers and aimed to be secured against attacks on classical computers
- **Post-quantum cryptography**: Cryptographic algorithms over classical computers and aimed to be secured against attacks on both classical and quantum computers
- **Quantum cryptography**: Cryptographic algorithms over quantum computers and aimed to be secured against attacks on both classical and quantum computers

Classical vs quantum computers

	Classical computer	**Quantum computer**
Composition	Transistor	Any physical particle that can have two properties at the same time
Computation model	Bits: 0 or 1	QuBits: Linear combination of 0 and 1
During computation	1 state of N bits	2^N states of N bits
End of computation	1 state of N bits	1 states of N bits (probabilistic)

- A quantum computer is **not** a super fast classical computer
- A quantum computer is **not** a computer that computes all solutions in parallel



Quantum computers vs cryptography

Algorithm	**Grover (1996)**	**Shor (1994)**
Impact	Symmetric cryptography Hash function	Asymmetric cryptography
Problem	Key brute force search Collision brute force search	Integer factorization Discrete logarithm
Classical computing	Exponential complexity	Sub-exponential complexity
Quantum computing	Sub-exponential complexity	Polynomial complexity (n^3)
Consequences	Problem easier than expected	Problem no longer difficult

Impact of quantum computers on cryptography

Algorithm	Impact of quantum computers
AES	Larger key sizes needed 256 bits for a 128-bits security level [$\times$2]
SHA-2, SHA-3	Larger output needed 384 bits for a 128-bits security level [$\times$3]
RSA	No longer secure
DSA, ECDSA	No longer secure
DH, ECDHE	No longer secure

Quantum computers development

- Quantum computers with 4000 to 6000 qubits are a threat to cryptography
- Estimations for large scale quantum computers availability range from 15 to 20 years
- Industry migration will take time and will be complex
- Some applications already need quantum resistant cryptography