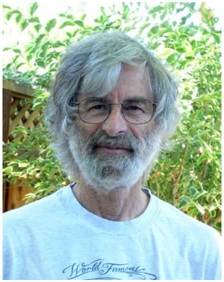
Brief history of Lamport's works



- **Inventing LaTeX:** a group of macros, which can make the life of TeXusers a lot easier!!
- **One way authentication** in Whitfield Diffie's "*New directions in cryptography*" (1976)

brief description of **One-way Function**

- A one way function $\boldsymbol{f}$ is a function that is **easy to compute** but whose **inverse is difficult to compute**:

[1] Lamport, "*Constructing Digital Signatures from a One Way Function*"

[2] i.e. to say: $\forall$ data object $d' \neq d : \phi(d') \neq \phi(d)$

brief description of **One-way Function**

- A one way function ***f*** is a function that is **easy to compute** but whose **inverse is difficult to compute**:
- or to say f is one-way function iff (adapted from[1]):

[1]Lamport, "*Constructing Digital Signatures from a One Way Function*"

[2]i.e. to say: $\forall$ data object $d' \neq d : \phi(d') \neq \phi(d)$

brief description of **One-way Function**

- A one way function $\boldsymbol{f}$ is a function that is **easy to compute** but whose **inverse is difficult to compute**:
- or to say f is one-way function iff (adapted from[1]):
  1. for all value v, finding data object d s.t. $\phi(d) = v$ is **computationally infeasible.**

[1] Lamport, "*Constructing Digital Signatures from a One Way Function*"

[2] i.e. to say: $\forall$ data object $d' \neq d : \phi(d') \neq \phi(d)$

brief description of **One-way Function**

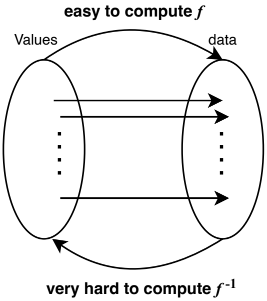
- A one way function $\boldsymbol{f}$ is a function that is **easy to compute** but whose **inverse is difficult to compute**:
- or to say f is one-way function iff (adapted from[1]):
  1. for all value v, finding data object d s.t. $\phi(d) = v$ is **computationally infeasible.**
  2. f is not **one-to-one** or proving it otherwise is **computationally infeasible.**[2]

[1]Lamport, "*Constructing Digital Signatures from a One Way Function*"

[2]i.e. to say: $\forall$ data object $d' \neq d : \phi(d') \neq \phi(d)$

brief description of **One-way Function**

- A one way function ***f*** is a function that is **easy to compute** but whose **inverse is difficult to compute**:
- or to say f is one-way function iff (adapted from[1]):
  1. for all value v, finding data object d s.t. $\phi(d) = v$ is **computationally infeasible.**
  2. f is not **one-to-one** or proving it otherwise is **computationally infeasible.**[2]



[1] Lamport, "*Constructing Digital Signatures from a One Way Function*"

[2] i.e. to say: $\forall$ data object $d' \neq d : \phi(d') \neq \phi(d)$

brief description of **One-way Authentication**

Definition (Digital signature)

A digital signature created by **sender P** for **document m** is a data item $\sigma_p(m)$ that is when received together with **m**, one can determine (e.g. in a court of law) that **P** generated document **m**.

[1] Lamport, L. (1979) "*Constructing Digital Signatures from a One Way Function*"

brief description of **One-way Authentication**

Definition (Digital signature)

A digital signature created by **sender P** for **document m** is a data item $\sigma_p(m)$ that is when received together with **m**, one can determine (e.g. in a court of law) that **P** generated document **m**.
Hence A tool for determining **validity** of something sent.[3]

[1]Lamport, L. (1979) "*Constructing Digital Signatures from a One Way Function*"

brief description of **One-way Authentication**

Definition (One way authentication)

It must be **easy for anyone** to recognize the signature as **authentic** but **impossible** for anyone other than the signer to produce it![4]

[1]Diffie, W. (1976)" *New Directions in Cryptography*"

A practical Example of a **One-way function** in **one-way authentication**
Login Problem

- User **A** enters Password PW and computer store it as $f(PW)$

A practical Example of a **One-way function** in **one-way authentication**
Login Problem

- User **A** enters Password PW and computer store it as $f(PW)$
- where $f(PW)$ is a one-way function of 10 million instructions

A practical Example of a **One-way function** in **one-way authentication**
Login Problem

- User **A** enters Password PW and computer store it as $f(PW)$
- where $f(PW)$ is a one-way function of 10 million instructions
- and its inverse has 10^{30} more instructions (or computations), which practically makes it **noninvertible**

A practical Example of a **One-way function** in **one-way authentication**
Login Problem

- User **A** enters Password PW and computer store it as $f(PW)$
- where $f(PW)$ is a one-way function of <u>10 million instructions</u>
- and its inverse has 10^{30} more instructions (or computations), which practically makes it **noninvertible**
- for example, finding square root of x_0 given in $f(x) = x^2$ is much harder than computing x^2 at x_0.

brief description of **One-way Authentication** *Cont'd*

- However, determining exactly what the one-way function should be is originally solved by **Lamport**
  which further lead to the publication of the paper: " *Constructing Digital Signatures from a One Way Function*"

brief description of **One-way Authentication** *Cont'd*

- But how this solution relates to the ecosystem of **public keys** is out of the scope of the presentation and discussed in the paper: *"New Directions in Cryptography"* by Whitfield Diffie (1976)

Brief history of Lamport's works

- **Inventing LaTeX:** a group of macros, which can make the life of TeXusers a lot easier!!
- **One way authentication** in Whitfield Diffie's "*New directions in cryptography*" (1976)
- **bakery algorithm** an algorithm to ensure mutual exclusion in concurrent processes

[1]Silberschatz, "Database System Concepts", Ch. 19, P. 965

Brief history of Lamport's works

- **Inventing LaTeX:** a group of macros, which can make the life of TeXusers a lot easier!!
- **One way authentication** in Whitfield Diffie's "*New directions in cryptography*" (1976)
- **bakery algorithm** an algorithm to ensure mutual exclusion in concurrent processes
- that is to ensure a data structure is modified by at most one process at a time
  and no process is reading a data structure while it is being written by other processes.

[1] Silberschatz, "Database System Concepts", Ch. 19, P. 965

Brief history of Lamport's works

- **Inventing LaTeX:** a group of macros, which can make the life of TeXusers a lot easier!!
- **One way authentication** in Whitfield Diffie's "*New directions in cryptography*" (1976)
- **bakery algorithm** an algorithm to ensure mutual exclusion in concurrent processes
- that is to ensure a data structure is modified by at most one process at a time
  and no process is reading a data structure while it is being written by other processes.
- **Paxos algorithm**: an algorithm used in distributed systems for reaching consensus, used in distributed storage systems

[1] Silberschatz, "Database System Concepts", Ch. 19, P. 965

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:
    - in a distributed system, sometimes it is **impossible** to say an event has happened before something else.

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:
  - in a distributed system, sometimes it is **impossible** to say an event has happened before something else.
  - hence, relation "*happened before* is a partial ordering relation.

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:
  - in a distributed system, sometimes it is **impossible** to say an event has happened before something else.
  - hence, relation "*happened before* is a partial ordering relation.
  - **Partial ordering? sounds familiar...**

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:
  - in a distributed system, sometimes it is **impossible** to say an event has happened before something else.
  - hence, relation "*happened before* is a partial ordering relation.
  - **Partial ordering? sounds familiar...**

Definition (Partial ordering)

Partial ordering relation is an ordering relation in which not all members of the set need to be comparable!

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:
    - in a distributed system, sometimes it is **impossible** to say an event has happened before something else.
    - hence, relation "*happened before* is a partial ordering relation.

    - He proposed in that paper a partial ordering of "happened before" and gave a **distributed algorithm** for **extending it to a total ordering of events**

Brief history of Lamport's works *cont'd*

- Time, clocks and ordering of events in a distributed system:
  - in a distributed system, sometimes it is **impossible** to say an event has happened before something else.
  - hence, relation "*happened before* is a partial ordering relation.
  - He proposed in that paper a partial ordering of "happened before" and gave a **distributed algorithm** for **extending it to a total ordering of events**

But where is the "Byzantine generals" problem in the list?