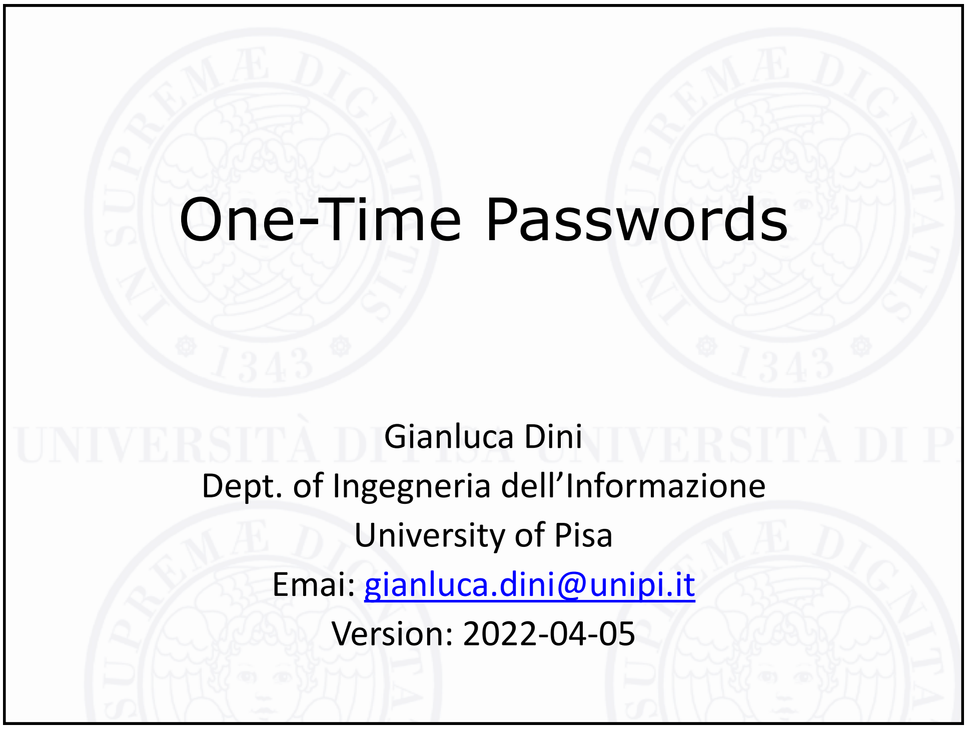
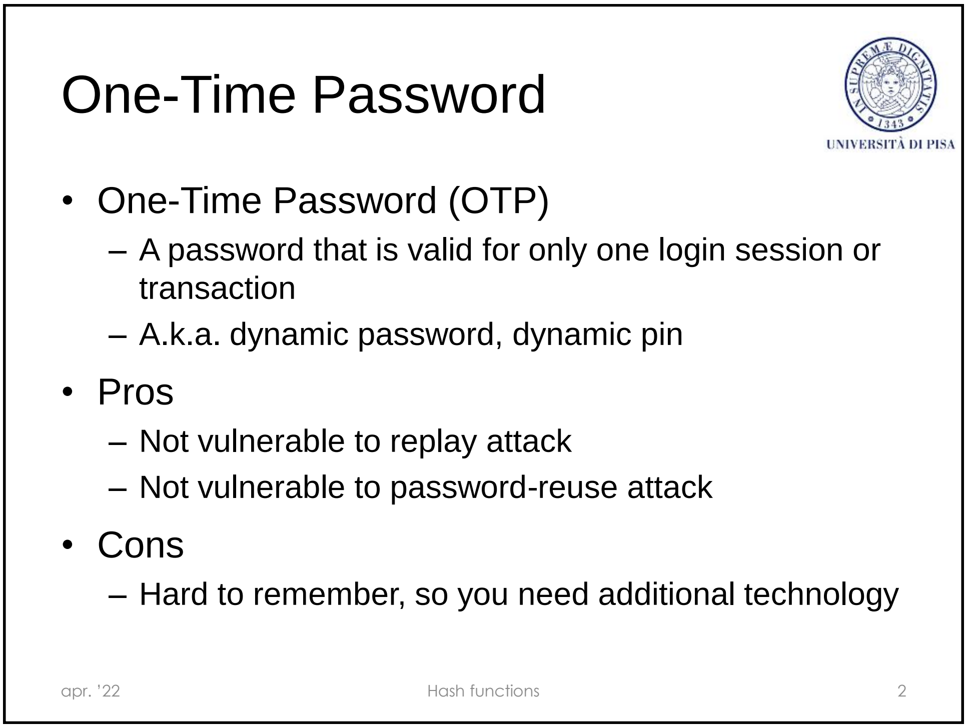


One-Time Passwords

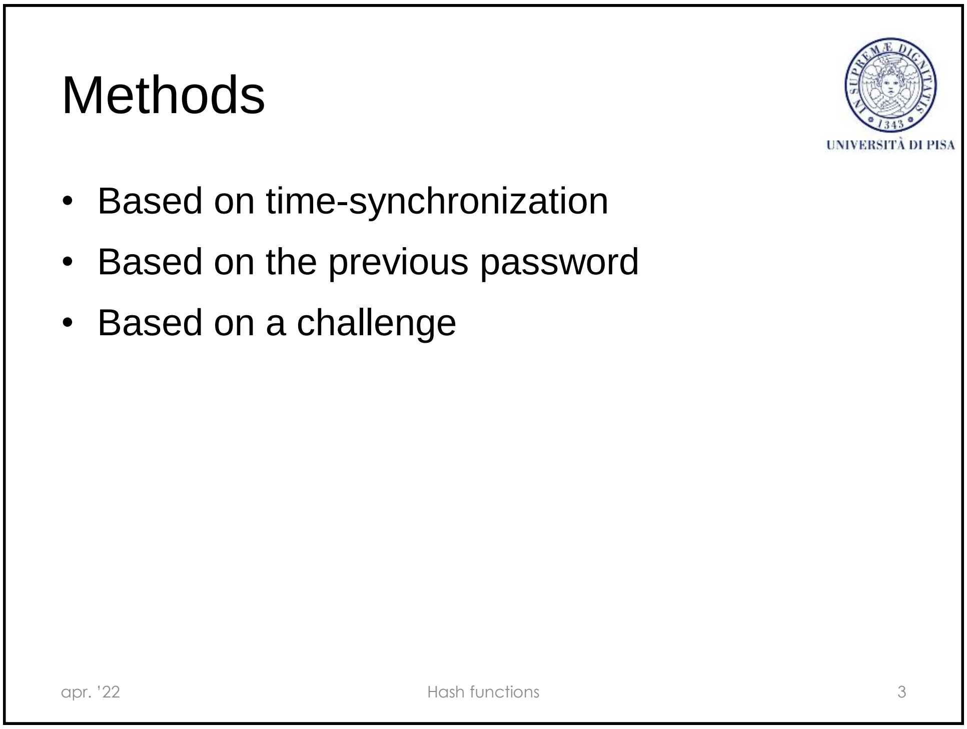
Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Emai: gianluca.dini@unipi.it

Version: 2022-04-05



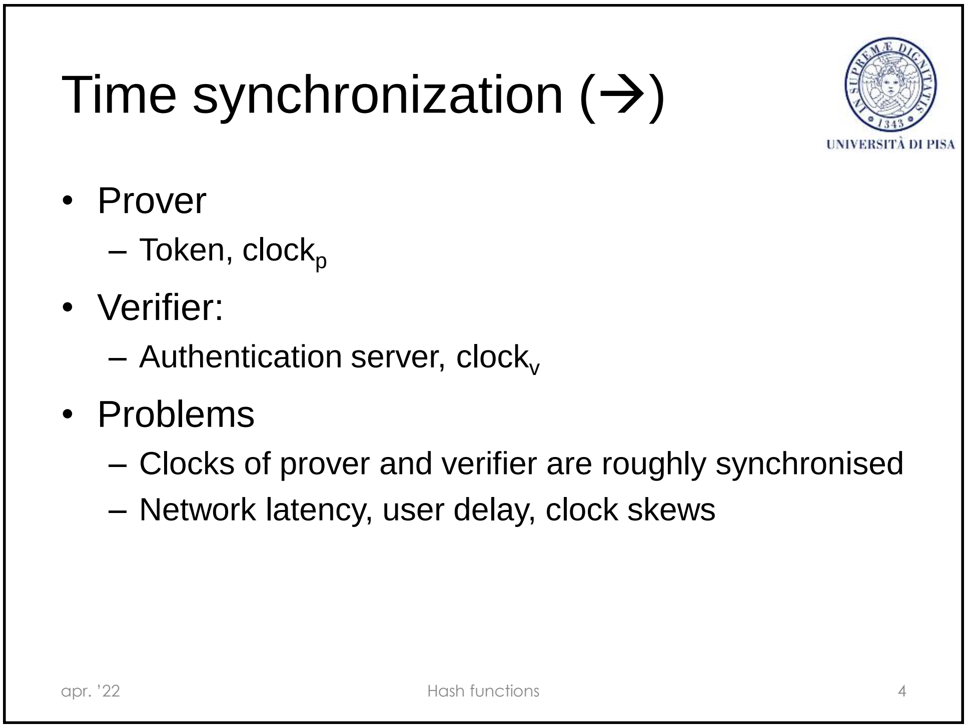
One-Time Password

- One-Time Password (OTP)
  - A password that is valid for only one login session or transaction
  - A.k.a. dynamic password, dynamic pin
- Pros
  - Not vulnerable to replay attack
  - Not vulnerable to password-reuse attack
- Cons
  - Hard to remember, so you need additional technology



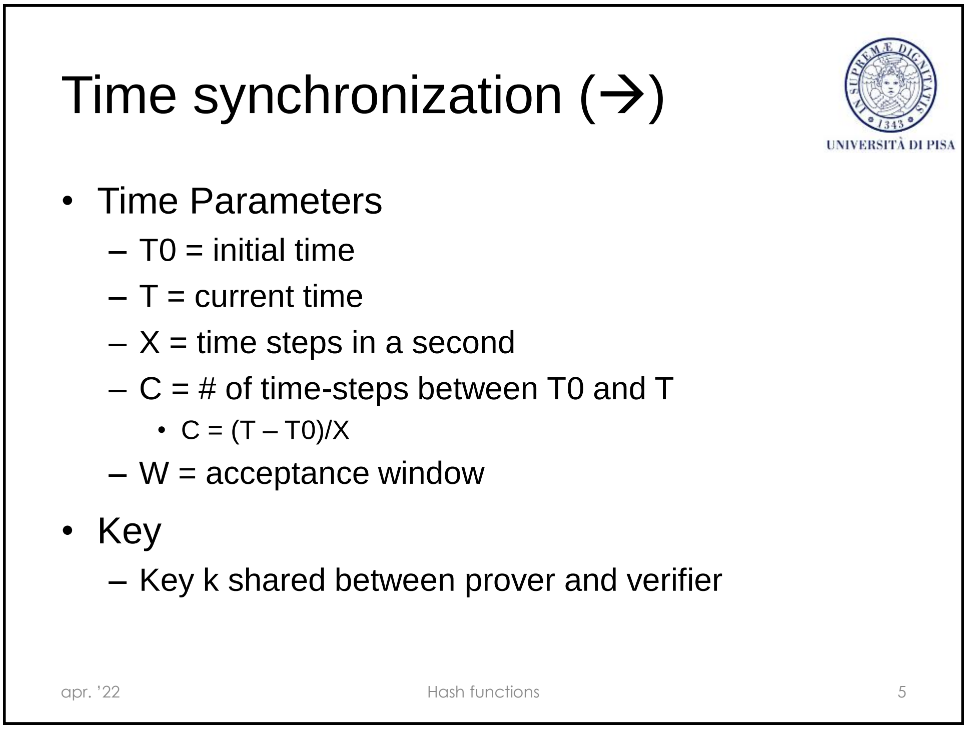
Methods

- Based on time-synchronization
- Based on the previous password
- Based on a challenge



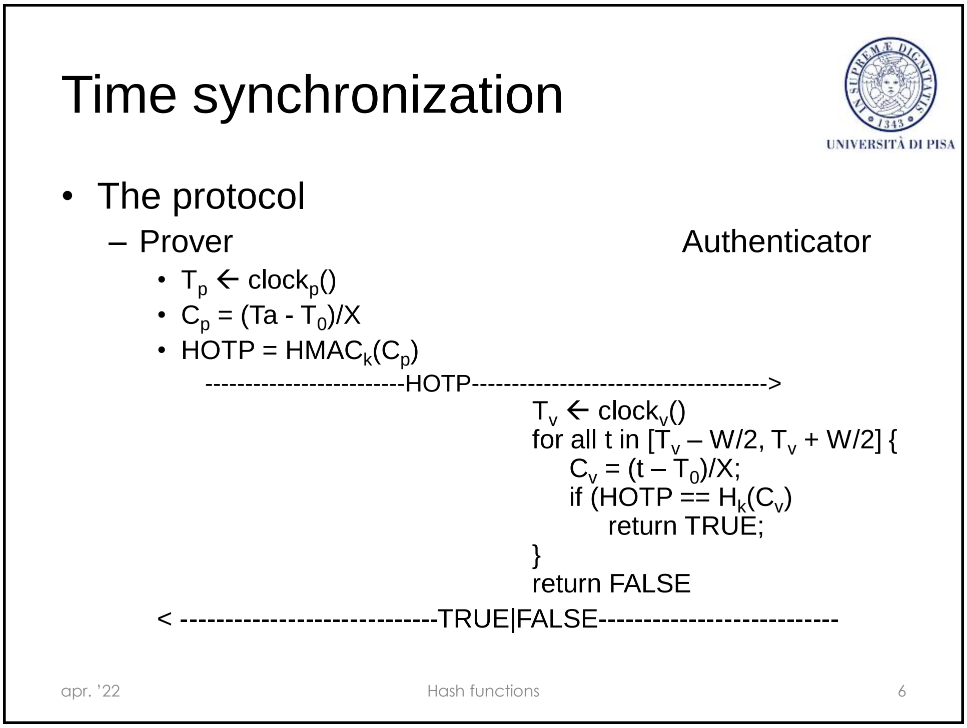
Time synchronization (→)

- Prover
  - Token, clock_p
- Verifier:
  - Authentication server, clock_v
- Problems
  - Clocks of prover and verifier are roughly synchronised
  - Network latency, user delay, clock skews



Time synchronization (→)

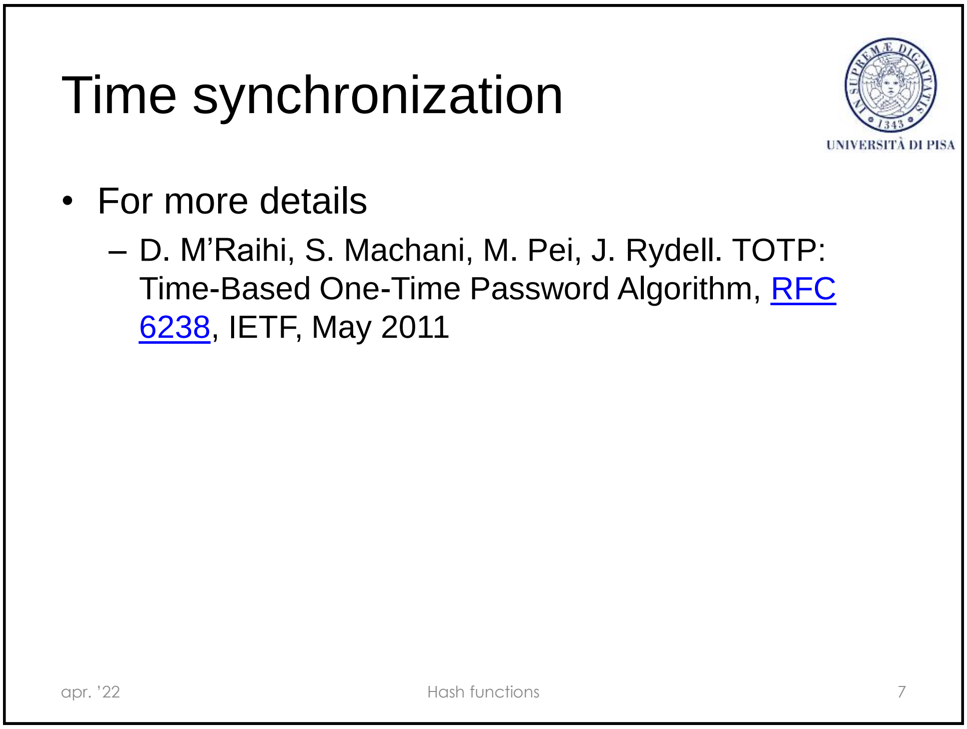
- Time Parameters
  - T0 = initial time
  - T = current time
  - X = time steps in a second
  - C = # of time-steps between T0 and T
    - C = (T – T0)/X
  - W = acceptance window
- Key
  - Key k shared between prover and verifier



Time synchronization

- The protocol
  - Prover   Authenticator

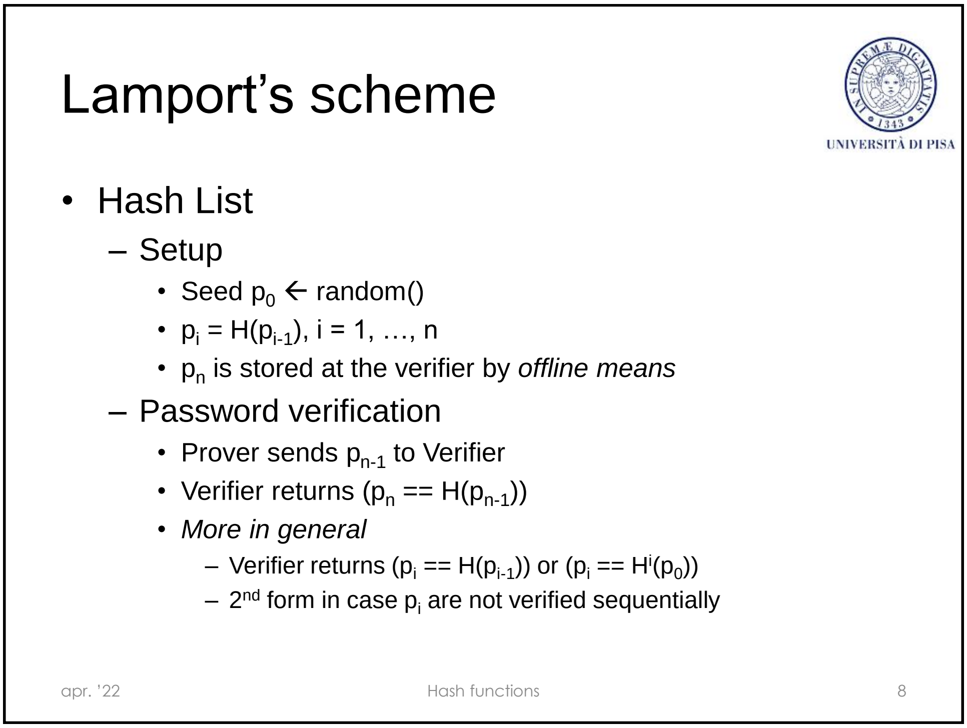
```
    • $T_p \leftarrow clock_p()$
    • $C_p = (Ta - T_0)/X$
    • $HOTP = HMAC_k(C_p)$
        ------------------------HOTP------------------------------------->
                                   $T_v \leftarrow clock_v()$
                                   for all t in $[T_v – W/2, T_v + W/2]$ {
                                       $C_v = (t – T_0)/X$;
                                       if ($HOTP == H_k(C_v)$)
                                           return TRUE;
                                   }
                                   return FALSE
    < ----------------------------TRUE|FALSE---------------------------
```



Time synchronization

- For more details
  - D. M'Raihi, S. Machani, M. Pei, J. Rydell. TOTP: Time-Based One-Time Password Algorithm, RFC 6238, IETF, May 2011

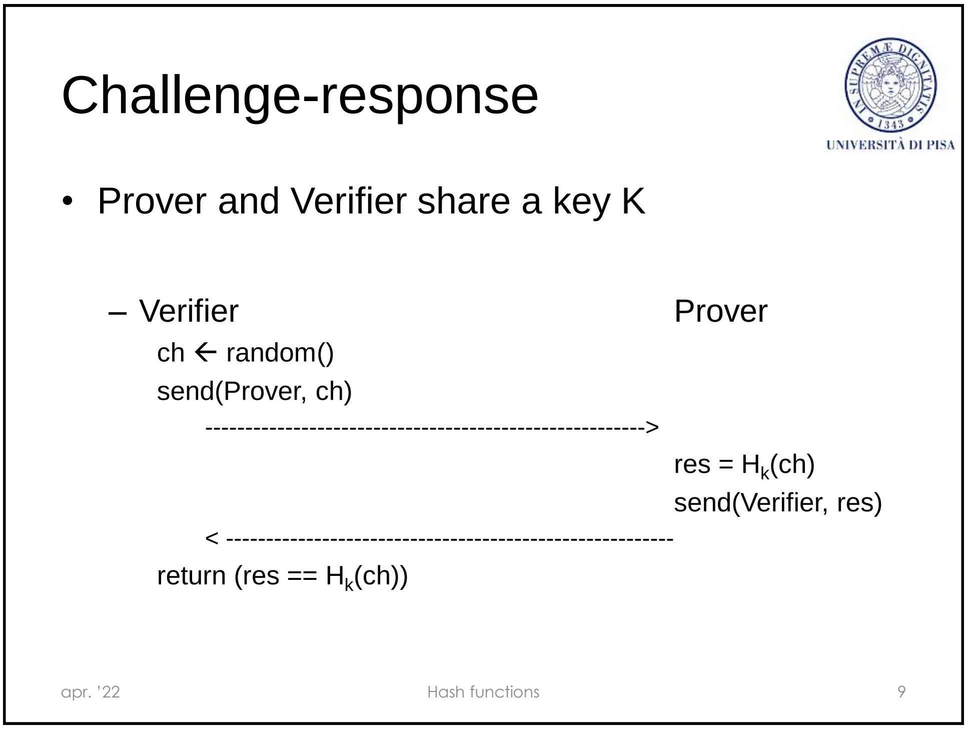
7



Lamport's scheme

- Hash List
  - Setup
    - Seed p_0 ← random()
    - $p_i = H(p_{i-1})$, $i = 1, \dots, n$
    - p_n is stored at the verifier by *offline means*
  - Password verification
    - Prover sends p_{n-1} to Verifier
    - Verifier returns ($p_n == H(p_{n-1})$)
    - *More in general*
      - Verifier returns ($p_i == H(p_{i-1})$) or ($p_i == H^i(p_0)$)
      - 2nd form in case p_i are not verified sequentially

8



Challenge-response

- Prover and Verifier share a key K

```
– Verifier                                        Prover
     ch ← random()
     send(Prover, ch)
          ------------------------------------------------------->
                                                  res = $H_k$(ch)
                                                  send(Verifier, res)
          < -------------------------------------------------------
     return (res == $H_k$(ch))
```