One-Time Passwords

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Emai: gianluca.dini@unipi.it
Version: 08/04/25

1

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

2

Methods

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

08/04/2025 One-time passwords 3

3

One-Time Password

OTP BASED ON TIME SYNCHRONIZATION

08/04/2025 One-time passwords 4

4

Time synchronization (→)

UNIVERSITÀ DI PISA

Used by some banks when they give you HW token

- Prover (User)
  - Hardware token, $clock_p$
- Verifier:
  - Authentication server, $clock_v$
- Prover and Verifier share a secret key k
- Problems
  - Clocks of prover and verifier are roughly synchronised
  - Network latency, user delay, clock skews

↳ HT is called prover, with its own clock

Server (bank) with a clock

Clock Cp in particular is not very accurate

08/04/2025 One-time passwords 5

5

Time synchronization (→)

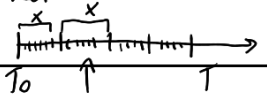
UNIVERSITÀ DI PISA

- 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

} not in seconds but in time steps

• not ??

X increases granularity for checks:



08/04/2025 One-time passwords 6

6

Trip: say T is current time, To is initial time and we compute C given # time steps in a second

Io posso decidere e prendere ogni singolo blocco. Così seleziono un chunk e aumento granularità.

Time synchronization

Imagine 2FA. This is the second factor

UNIVERSITÀ DI PISA

- The protocol
  - **Prover** **Authenticator**
    - $T_p \leftarrow clock_p()$
    - $C_p = (T_p - T_0)/X$
    - $HOTP = HMAC_k(C_p)$

There is a delay here

Computes the hash

```
----------HOTP---------->
                 Tv ← clockv()
                 for all t in [Tv – W/2, Tv + W/2] {
                     Cv = (t – T0)/X;
                     if (HOTP == Hk(Cv)
                         return TRUE;
                 }
                 return FALSE
< ----------TRUE|FALSE----------
```

08/04/2025 One-time passwords 7

7

Problem: C_p depends on time of prover, but clocks are not synchronised and there is delay. So verifier sets a window in which we try the possible values we have. Value of X tells me distance between two different times. And I can play on acceptance window. Larger windows are less secure, smaller ones increase prob. of denial of service.

Time synchronization

UNIVERSITÀ DI PISA

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

08/04/2025 One-time passwords 8

8

One-Time Password

OTP BASED ON THE PREVIOUS PASSWORD

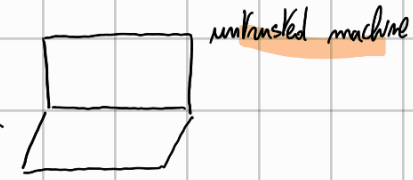
9

The Lamport's scheme

- Hash List
  - Setup
    - Seed $p_0 \leftarrow$ random()
    - $p_i = H(p_{i-1})$, i = 1, …, 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

10

Imagine you trust and you use a machine you don't trust:



If you type in your pwd, it could be compromised on untrusted machine.

One solution: Lamport's scheme: Suppose you are user.

You generate p_0 which is a random number. You start computing p_0 and compute $p_1 = H(p_0)$, then $p_2 \xrightarrow{H} p_3 \xrightarrow{H} \dots \to p_m$ Those are each a number of pwds you can use.

So pwd generation goes from $p_0 \to p_m$. And you give p_m to the home by offline means. And then you start using pwd in reverse order.

First login uses p_{m-1}, that is received by home server. home checks wether $H(p_{m-1}) = p_m$.

If p_{m-1} is compromised, it cannot be reused and you cannot compute p_{m-2} because that is the preimage of p_{m-1}.

Notice also that if one pwd gets lost, I don't need to follow a strict order. I could send p_i, and hash p_0 i times to see if it is correct.

One-Time Password

OTP BASED ON A CHALLENGE

11

Challenge-response

UNIVERSITÀ DI PISA

- Prover and Verifier share a key K

Verifier		Prover
chl ← random() generates a random #		
send(Prover, chl) Sends challenge to prover		
	---------->	
		res = H_k(chl) MAC as a response. (prover computes)
		send(Verifier, res)
	< ----------	
return (res == H_k(ch))		

You can also replace it by digital signature

12

NOTE: Prover has no proof that they are talking to a verifier① It is important that challenge is generated in a random, unpredictable way

① If you need mutual auth, use other protocols