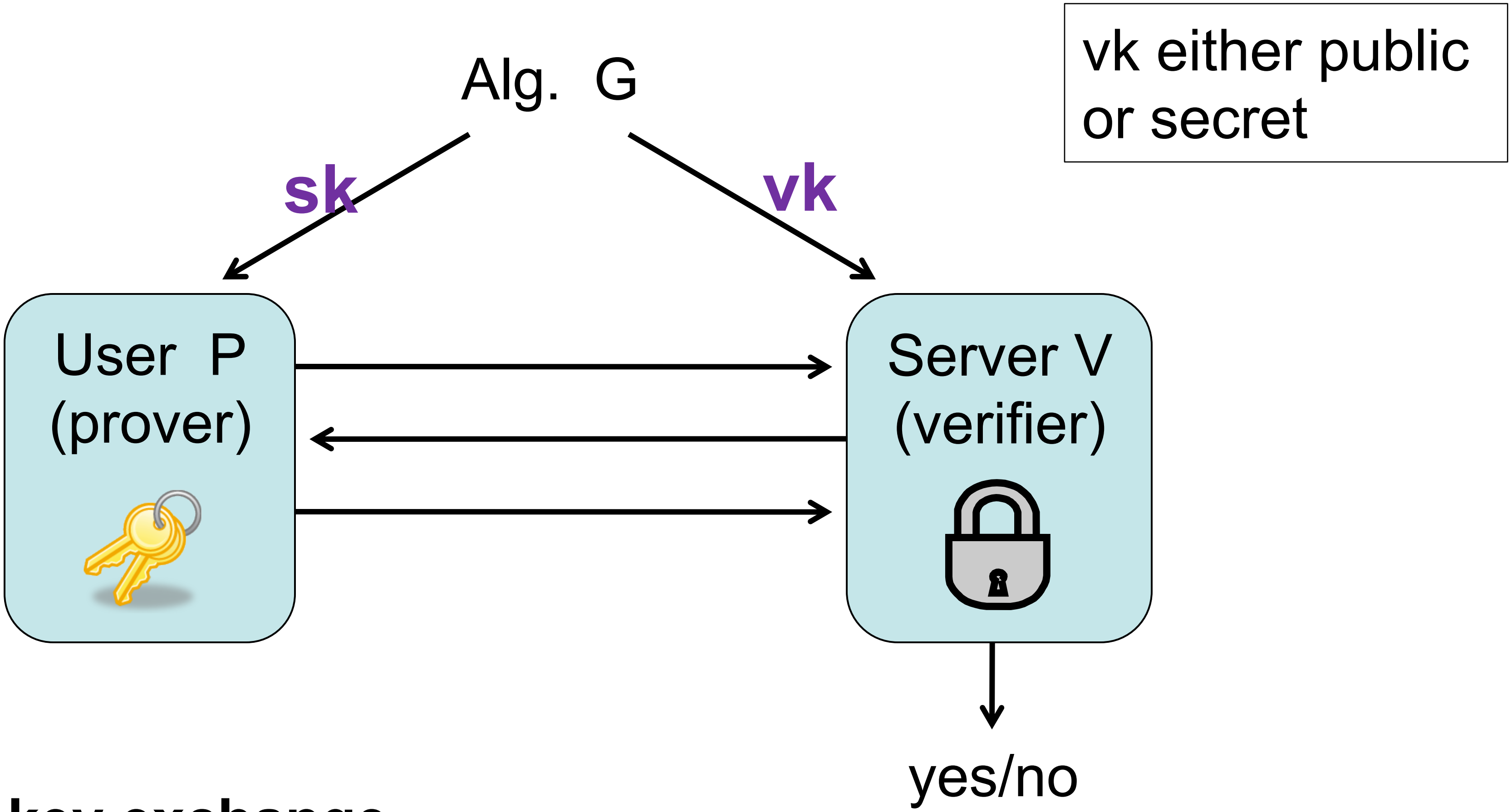
CS255

User Authentication: ID protocols

D. Boneh

The Setup



vk either public or secret

no key exchange

Applications

- Physical locks: (friend-or-foe)
  - Wireless car entry system (e.g. KeeLoq)
  - Opening an office door or a garage door

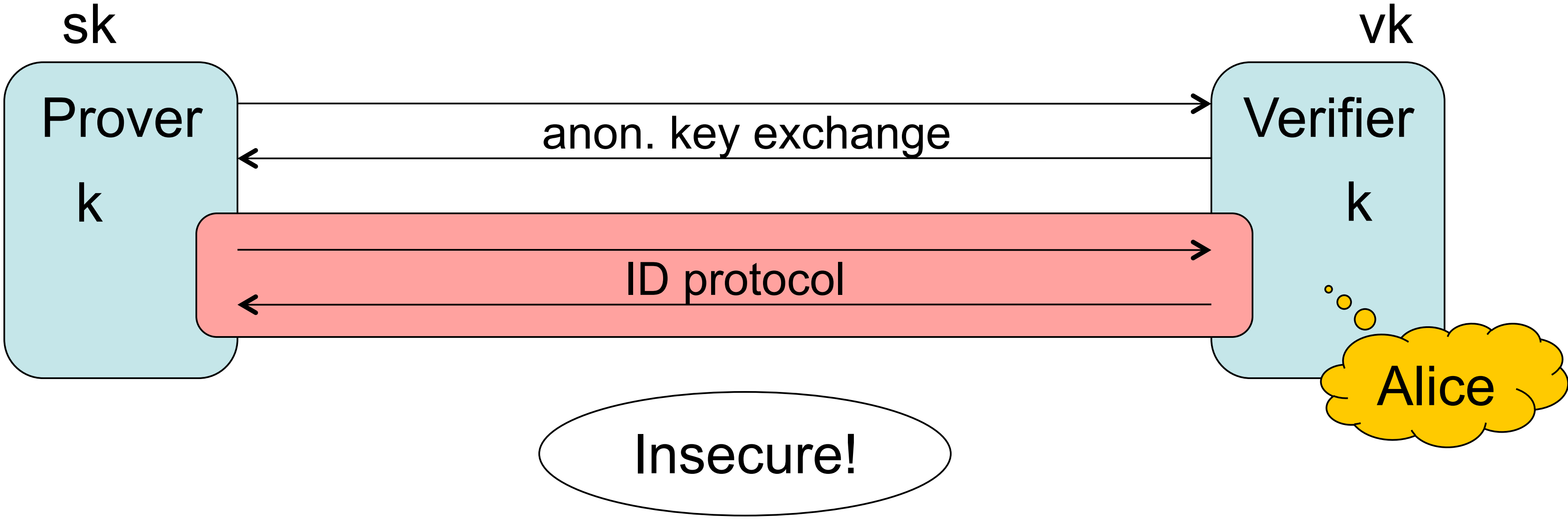
- Login at a bank ATM or a desktop computer

- Login to a remote web site once key-exchange with one-sided authentication completes (e.g. SSL)

ID Protocols: how not to use

ID protocol do not establish a secure session between Alice and Bob !!

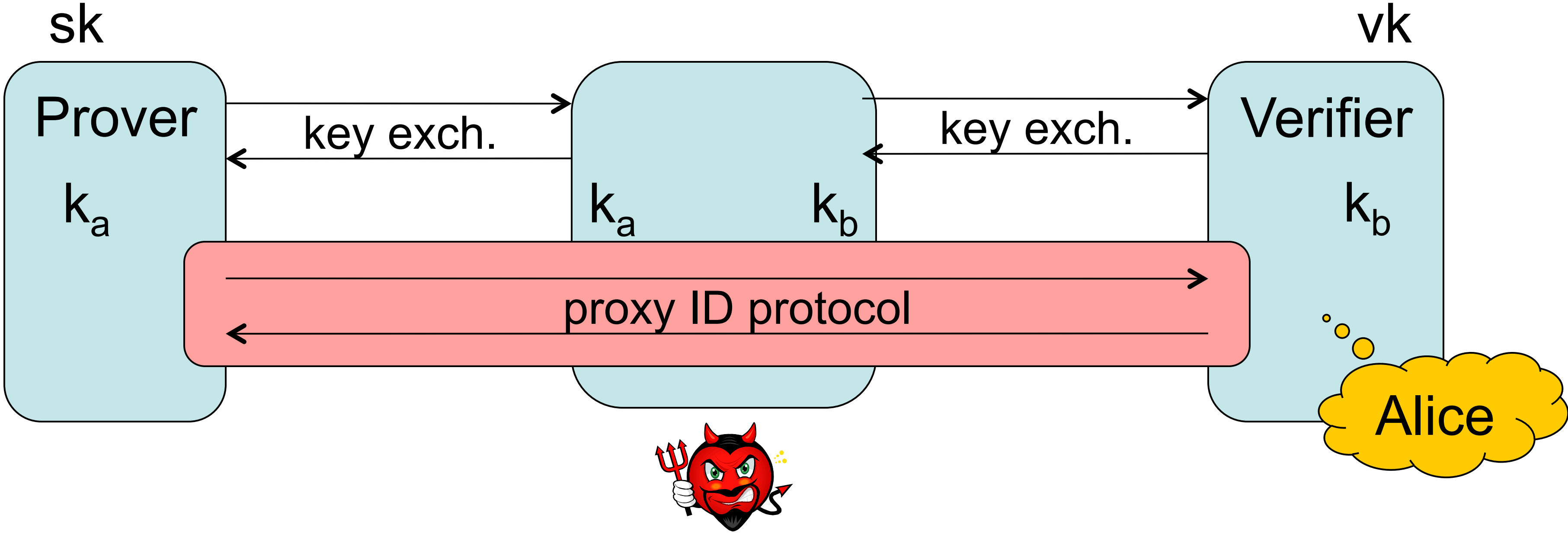
- Not even when combined with anonymous key exch.
- Vulnerable to man in to the middle attacks



ID Protocols: how not to use

ID protocol do not set up a secure session between Alice and Bob !!

- Not even when combined with anonymous key exch.
- Vulnerable to man in to the middle attack



ID Protocols: Security Models

1. **Direct Attacker**: impersonates prover with no additional information (other than vk)
   - Door lock

2. **Eavesdropping attacker**: impersonates prover after eavesdropping on a few conversations between prover and verifier
   - Wireless car entry system

3. **Active attacker**: interrogates prover and then attempts to impersonate prover
   - Fake ATM in shopping mall

ID protocols secure against direct attacks

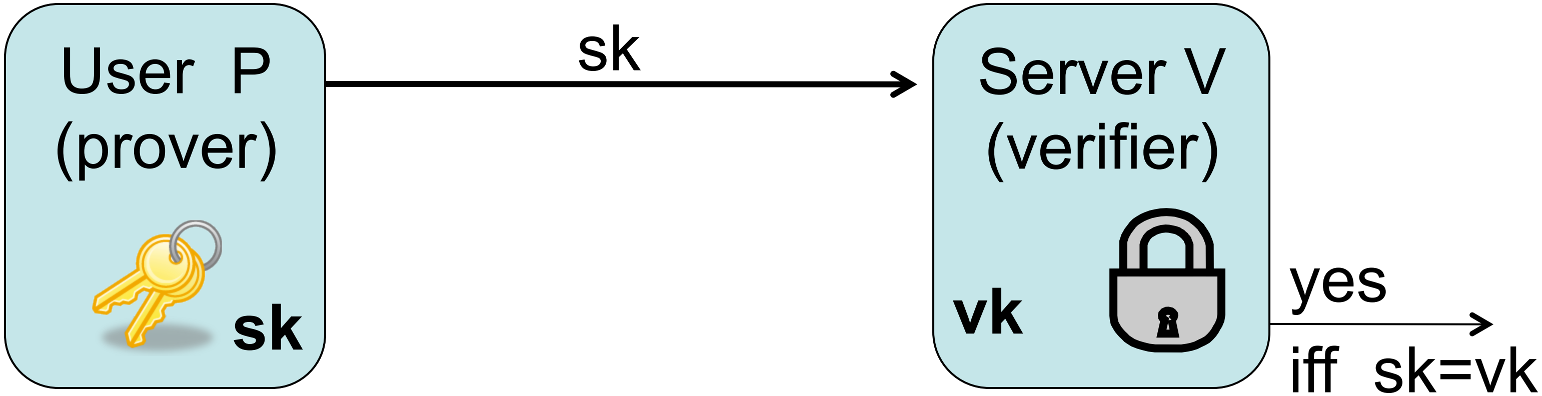
a.k.a Password Systems

Basic Password Protocol (incorrect version)

PWD: finite set of passwords

Algorithm G (KeyGen):

- choose pw ← PWD. output sk = vk = pw.



Basic Password Protocol (incorrect version)

Problem: VK must be kept secret

- Compromise of server exposes all passwords
- Never store passwords in the clear!

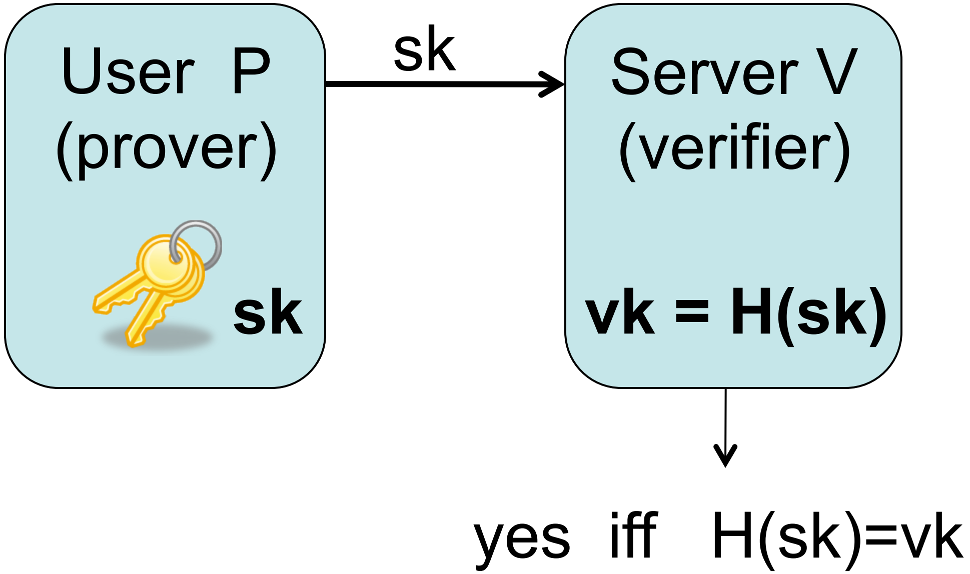
password file on server

Alice	pw_{alice}
Bob	pw_{bob}
…	…

Basic Password Protocol: version 1

H: one-way hash function from PWD to X

- "Given H(x) it is difficult to find y such that H(y)=H(x)"



User P (prover)

sk

sk

Server V (verifier)

vk = H(sk)

yes iff H(sk)=vk

password file on server

Alice	H(pw_A)
Bob	H(pw_B)
…	…

Weak Passwords and Dictionary Attacks

People often choose passwords from a small set:

- The 6 most common passwords (sample of 32×10^6 pwds):

  123456, 12345, Password, iloveyou, princess, abc123

  (‘123456’ appeared 0.90% of the time)

- 23% of users choose passwords in a dictionary of size 360,000,000

Online dictionary attacks:

- Defeated by doubling response time after every failure
- Harder to block when attacker commands a bot-net

Offline Dictionary Attacks

Suppose attacker obtains vk = H(pw) from server

- **Offline** attack: hash all words in Dict until a word w is found such that H(w) = vk
- Time O(|Dict|) per password

Off the shelf tools

- 2,000,000 guesses/sec
- Scan through 360,000,000 guesses in few minutes
  - Will recover 23% of passwords

Password Crackers

Algorithm	Speed/sec
DES	2 383 000
MD5	4 905 000
LanMan	12 114 000

Many tools for this

- John the ripper
- Cain and Abel
- Passware(Commercial)

Using CUDA: 5x speed-up

Batch Offline Dictionary Attacks

Alice	H(pw_A)
Bob	H(pw_B)
…	…

Suppose attacker steals pwd file F

- Obtains hashed pwds for **all** users

Batch dict. attack:

- Build list L containing **(w, H(w))** for all w ∈ Dict
- Find intersection of L and F

Total time: **O(|Dict| + |F|)**

Much better than a dictionary attack on each password

Preventing Batch Dictionary Attacks

id	S	h
Alice	$\mathbf{S_A}$	$H(pw_A , \mathbf{S_A})$
Bob	$\mathbf{S_B}$	$H(pw_B , \mathbf{S_B})$
…	…	…

Public salt:

- When setting password, pick a random n-bit salt S
- When verifying pw for A, test if **$H(pw, S_A) = h_A$**

Recommended salt length, n = 64 bits

- Pre-hashing dictionary does not help

Batch attack time is now: O(**|Dict| × |F|**)

Further Defenses

Slow hash function H: (0.1 sec to hash pw)

- Example: H(pw) = SHA1(SHA1(… SHA1(pw) …))
- Unnoticeable to user, but makes offline dictionary attack harder

Secret salts:

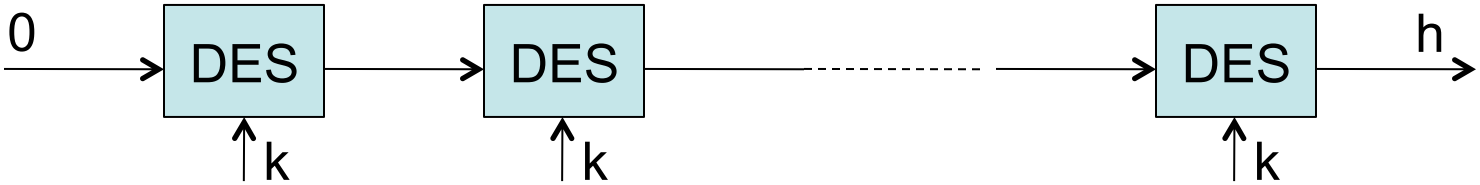
- When setting pwd choose short random r (8 bits)
- When verifying pw for A, try all values of r_A: 128 times slow down on average
- 256 times slow down for attacker

Alice	S_A	$H(pw_A , S_A , \mathbf{r_A})$
Bob	S_B	$H(pw_B , S_B , \mathbf{r_B})$
…	…	…

Case study: UNIX and Windows

UNIX: 12-bit public salt

- Hash function H:
  - Convert pw and salt and a DES key k
  - Iterate DES (or DES') 25 times:



Windows: NT and later use MD4

- Outputs a 16 byte hash
- No public or secret salts

Biometrics

Examples:

- Fingerprints, retina, facial recognition, …
- Benefit: hard to forget

Problems:

- Biometrics are not generally secret
- Cannot be changed, unlike passwords

⇒ Primarily used as a second factor authentication

The Common Password Problem

Users tend to use the same password at many sites

- Password at a high security site can be exposed by a break-in at a low security site

Standard solution:

- Client side software that converts a common password pw into a unique site password

  $$pw' \leftarrow H(pw, \text{user-id}, \text{server-id})$$

  pw' is sent to server

ID protocols secure against <u>eavesdropping</u> attacks

a.k.a   One-time Password Systems

Eavesdropping Security Model

Adversary is given:

- vk, and
- the transcript of several interactions between honest prover and verifier.

adv. goal is to then impersonate prover to verifier

A protocol is “secure against eavesdropping” if no efficient adversary can win this game

The password protocol is clearly insecure

- We discuss two secure stateful protocols (one-time pwd), and
- one stateless protocol (challenge-response)

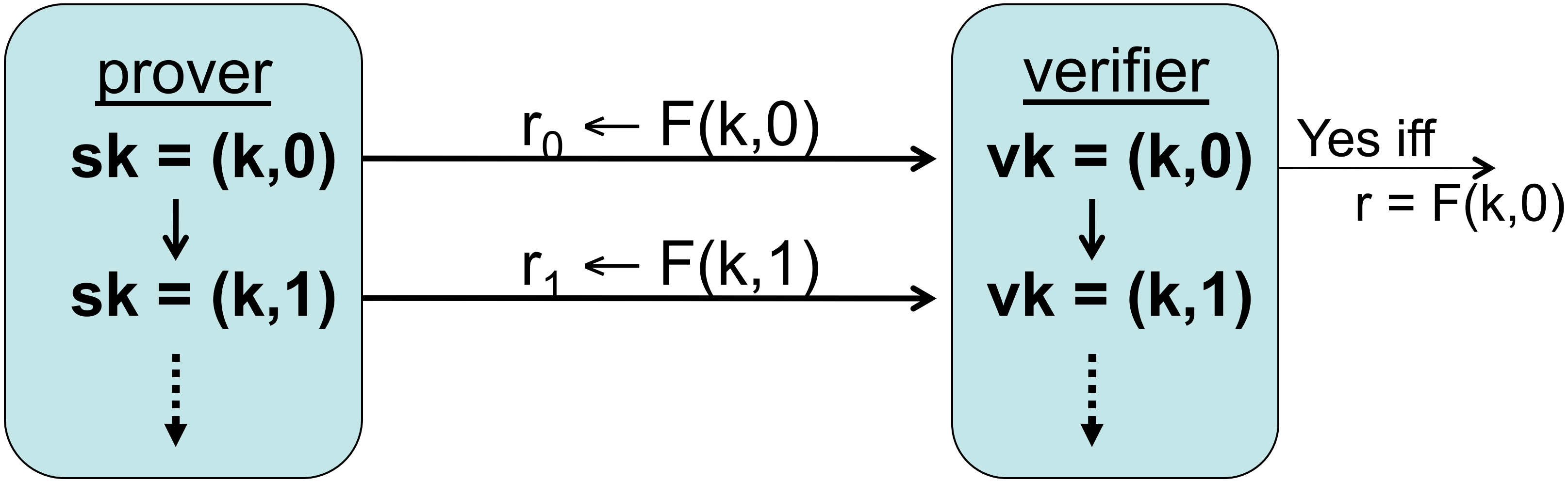
The SecurID system (secret vk, stateful)

vasco

Algorithm G: (setup)

- Choose random key $k \leftarrow K$
- Output **sk = (k,0)** ; **vk = (k,0)**

Identification:



prover

sk = (k,0)

↓

sk = (k,1)

$r_0 \leftarrow F(k,0)$

$r_1 \leftarrow F(k,1)$

verifier

vk = (k,0)

↓

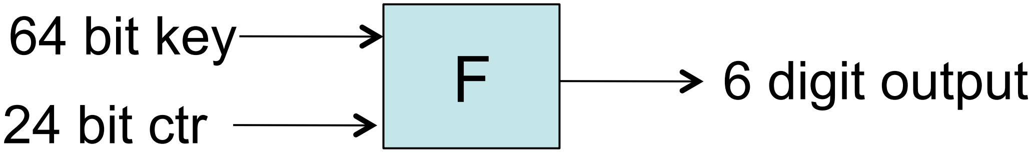
vk = (k,1)

Yes iff $r = F(k,0)$

The SecurID system (secret vk, stateful)

“Thm”: if F is a secure PRF then protocol is secure against eavesdropping

RSA SecurID uses a custom PRF:



64 bit key

24 bit ctr

F

6 digit output

vasco

Advancing state: sk ← (k, i+1)

- Time based: every 60 seconds
- User action: every button press

Both systems allow for skew in the counter value

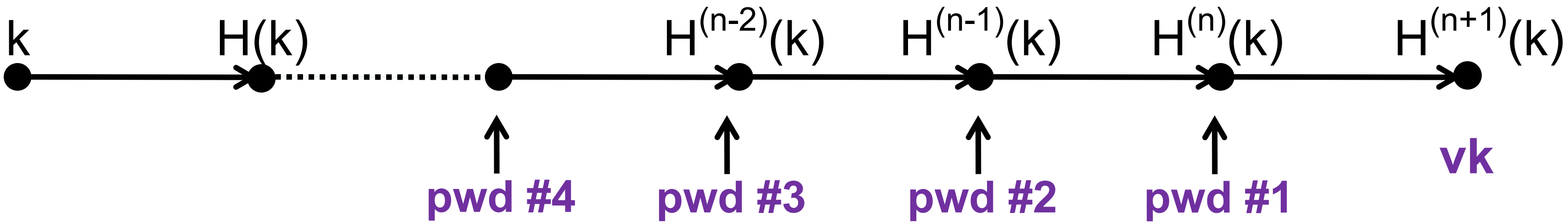
The S/Key system (public vk, stateful)

Notation: $H^{(n)}(x) = \underbrace{H(H(\ldots H(x)\ldots))}_{\text{n times}}$

<u>Algorithm G:</u> (setup)

- Choose random key $k \leftarrow K$
- Output **sk = (k,n)** ; **vk = $H^{(n+1)}(k)$**

<u>Identification</u>:



The S/Key system (public vk, stateful)

Identification (in detail):

- Prover (**sk=(k,i)**): send **$t \leftarrow H^{(i)}(k)$** ; set **$sk \leftarrow (k,i-1)$**
- Verifier(**$vk=H^{(i+1)}(k)$**): if **$H(t)=vk$** then **$vk \leftarrow t$**, output “yes”

Notes: vk can be made public;
but need to generate new sk after n logins ($n \approx 10^6$)

“Thm”: S/Key_n is secure against eavesdropping (public vk)
provided H is one-way on n-iterates

SecurID vs. S/Key

S/Key:

- **public** vk, **limited** number of auths
- often implemented using pencil and paper

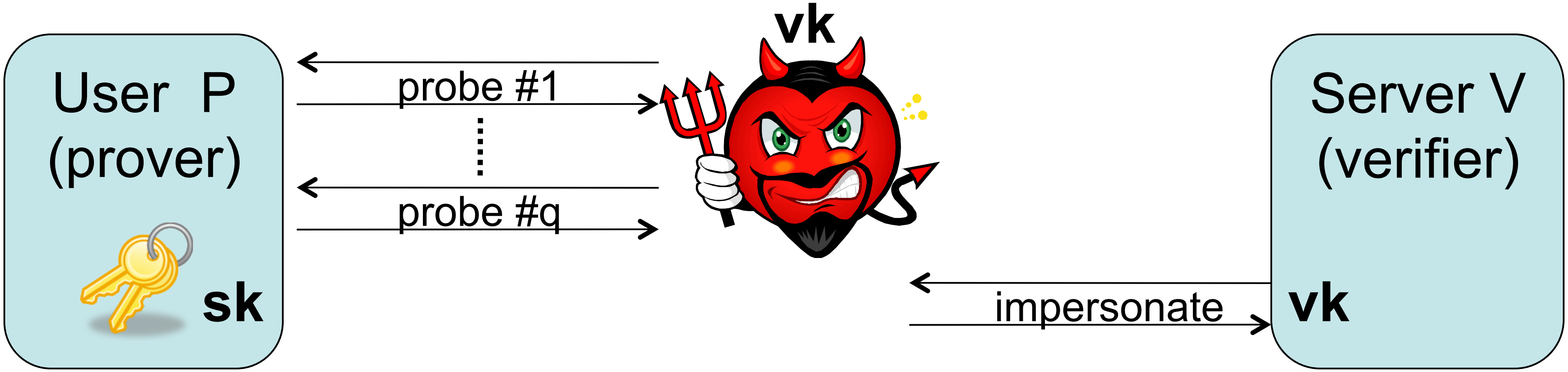
SecurID:

- **secret** vk, **unlimited** number of auths
- often implemented using secure token

ID protocols secure against active attacks

a.k.a Challenge-Response Protocols

Active Attacks

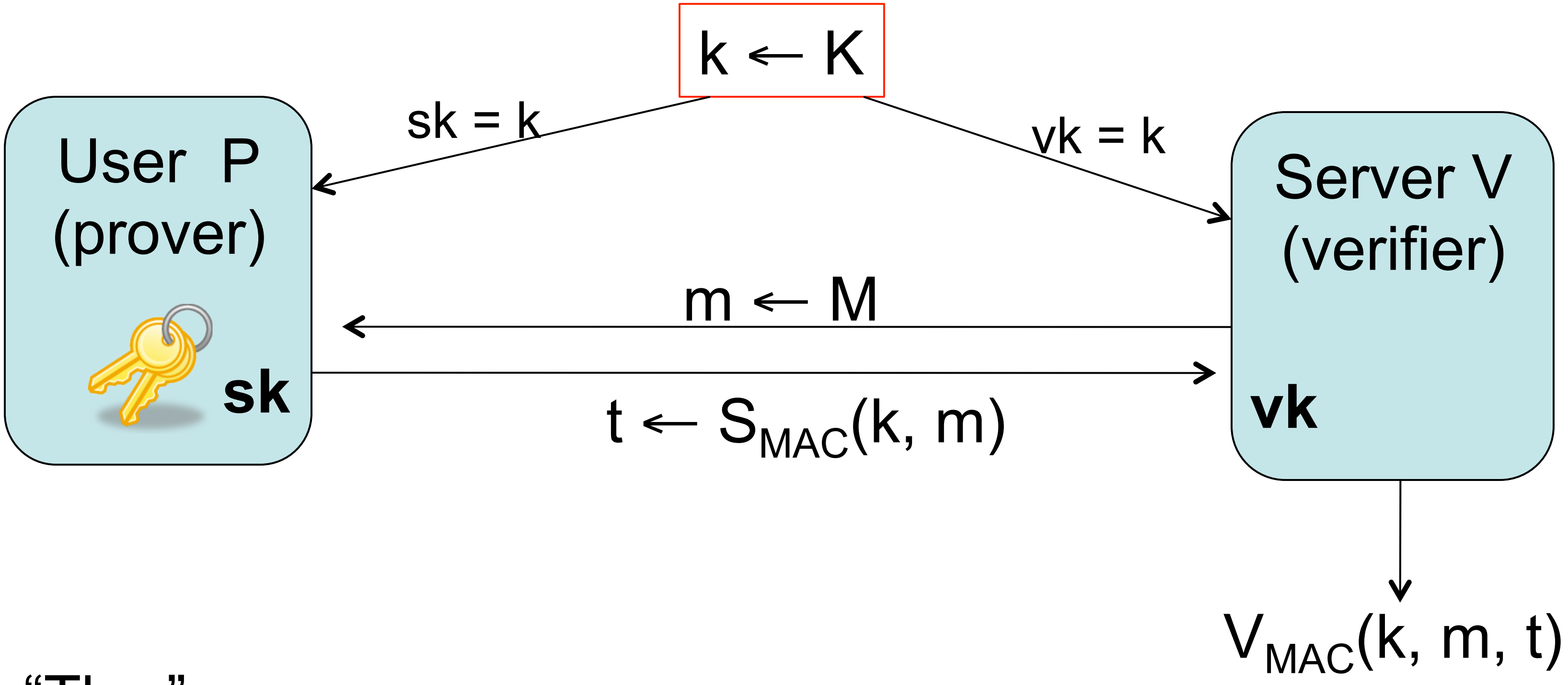


Offline fake ATM: interacts with user; later tries to impersonate to legit. ATM

Offline phishing: phishing site interacts with user; later authenticates to real site

Protocols so far are vulnerable

MAC-based Challenge Response (secret vk)



$k \leftarrow K$

sk = k

vk = k

User P (prover)

sk

Server V (verifier)

vk

$m \leftarrow M$

$t \leftarrow S_{MAC}(k, m)$

$V_{MAC}(k, m, t)$

“Thm”:

Protocol is secure against active attacks (secret vk), provided (S_{MAC}, V_{MAC}) is a secure MAC

MAC-based Challenge Response

Problems:

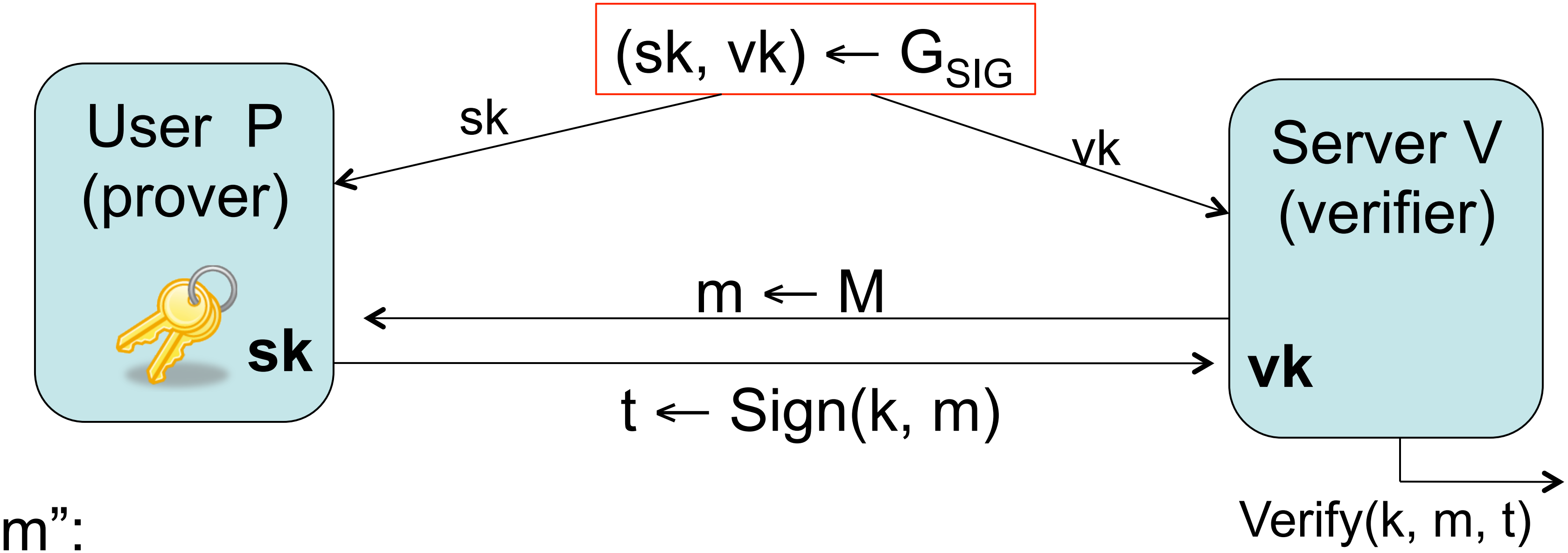
- vk must be kept secret on server
- dictionary attack when k is a human pwd:
  - Given [m , S_{MAC} (pw, m)] eavesdropper can try all pw ∈ Dict to recover pw

Main benefit:

- Both m and t can be short
- CryptoCard: 8 chars each

Sig-based Challenge Response (public vk)

Replace MAC with a digital signature:



$(sk, vk) \leftarrow G_{SIG}$

User P (prover)

sk

sk

vk

Server V (verifier)

$m \leftarrow M$

vk

$t \leftarrow Sign(k, m)$

Verify(k, m, t)

“Thm”:

Protocol is secure against active attacks **(public vk),** provided (G_{SIG} ,Sign,Verify) is a secure digital sig.

but t is long (≥20 bytes)

Summary

- ID protocols: useful in settings where adversary cannot interact with prover during impersonation attempt

- Three security models:
  - **Direct**: passwords (properly salted and hashed)
  - **Eavesdropping attacks**: One time passwords
    - SecurID: secret vk, unbounded logins
    - S/Key: public vk, bounded logins
  - **Active attacks**: challenge-response

Advanced Topics

- Anonymous digital cash
- Zero knowledge protocols and Dlog signatures
- Quantum computing
- Elliptic curve cryptography
- Factoring algorithms
- Advanced pub-key techniques: IBE, ABE, functional

THE END