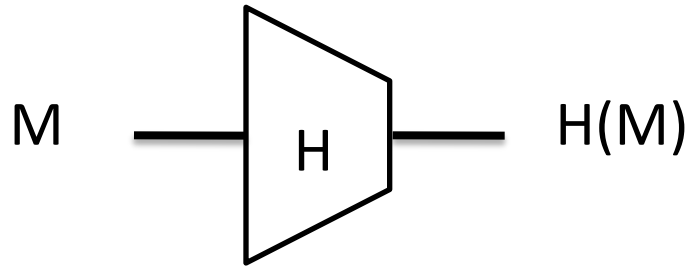
Today in Cryptography (5830)

Hash functions
HMAC
Passwords and password-based key derivation

Cryptographic hash functions

A cryptographich hash function H maps arbitrary bit string to fixed length string of size m



MD5: m = 128 bits
SHA-1: m = 160 bits
SHA-256: m = 256 bits

Some security goals:

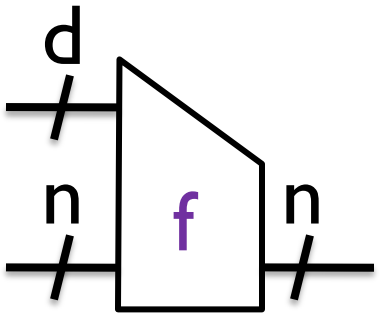
- collision resistance: can't find M != M' such that H(M) = H(M')
- preimage resistance: given H(M), can't find M
- second-preimage resistance: given H(M), can't find M' s.t. H(M') = H(M)
- Behave like a *public*, random function. Sometimes called random oracle model (ROM)

Pseudorandom functions vs. random oracles

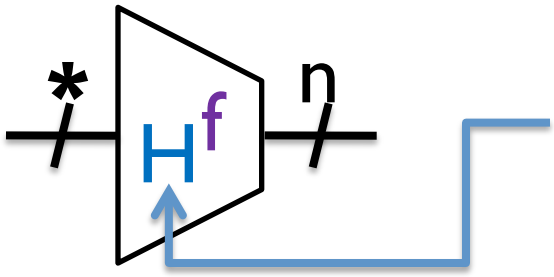
	Inputs	Security	Examples
PRF	Secret key, message	Indistinguishable from random function to any party without key	CBC-MAC HMAC
Random oracle (RO)	Message	Is a random function, but one that everyone can compute	SHA-256 SHA-512 SHA-3

Two-step design for hash functions

Compression Function

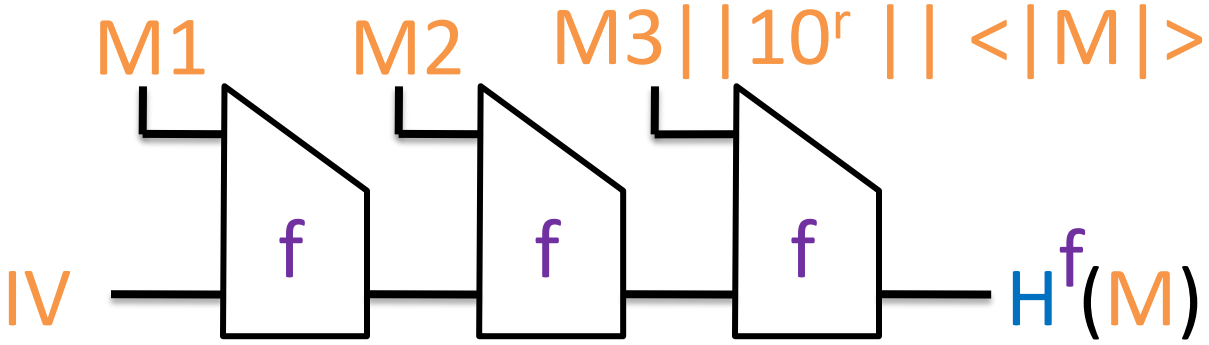


Hash Function



Domain extension transform

E.g., H = "Merkle-Damgard with strengthening"



Used in MD-x, SHA-1, SHA-256, …

Building compression functions

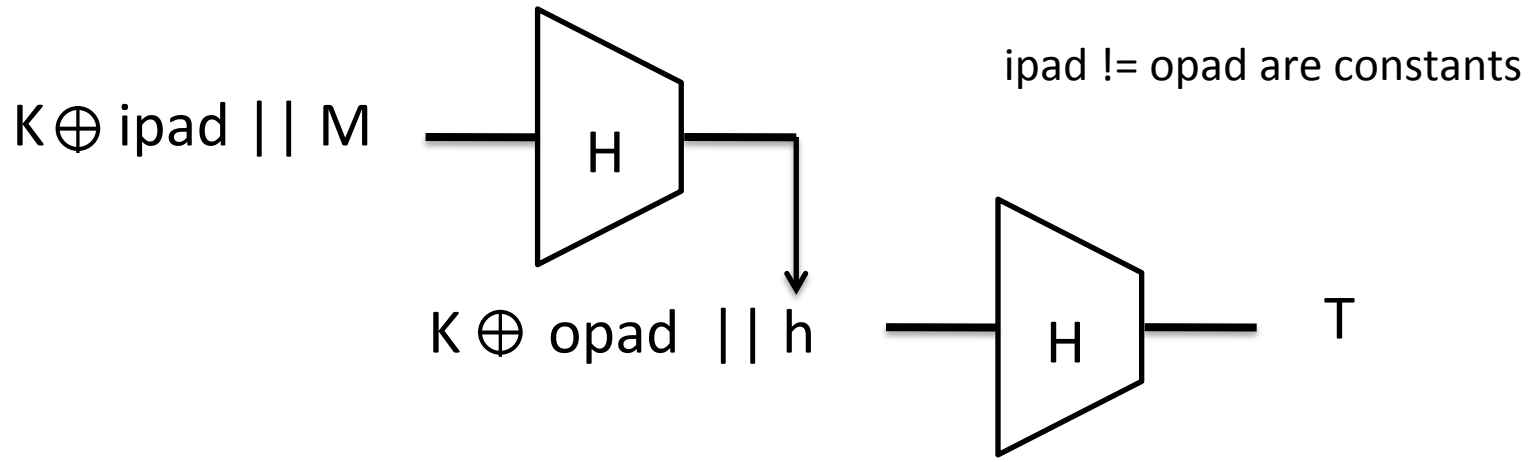
- Can build compression functions from suitable block ciphers

$$f(z,m) = E(m,z) \oplus z$$

- Can use AES, but security too low. Why?

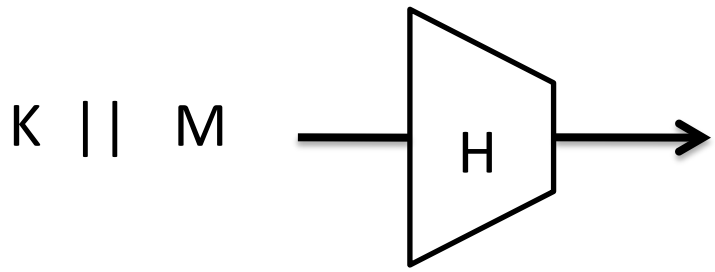
Building PRFs with hash functions: HMAC

Use a hash function H to build a MAC. K is a secret key

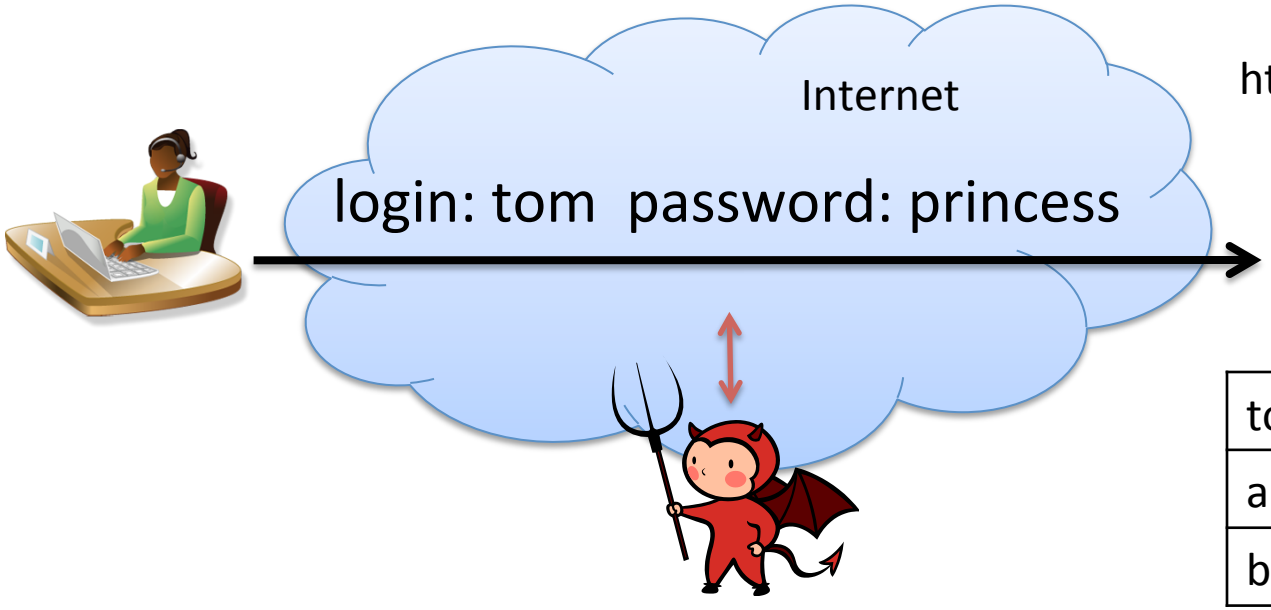


This is slight simplification, assuming |K| < d
(recall d is underlying message block length)

What's wrong with this PRF construction?

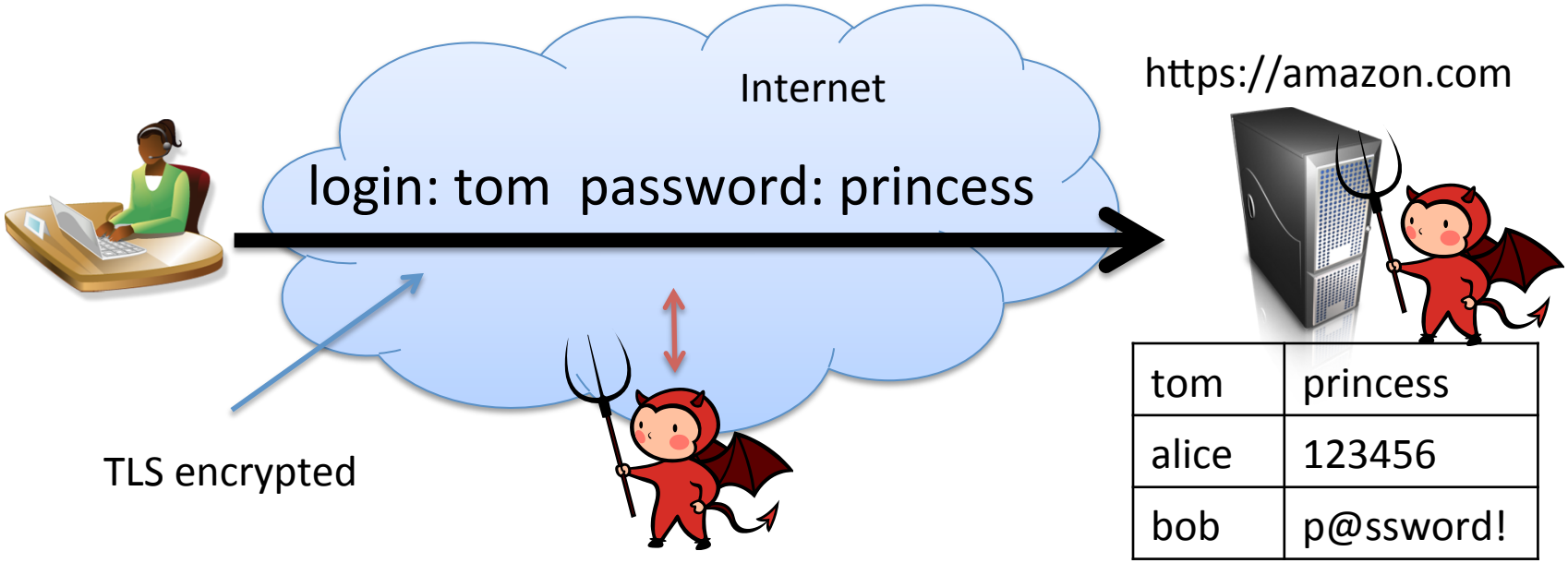


Passwords



tom	princess
alice	123456
bob	p@ssword!

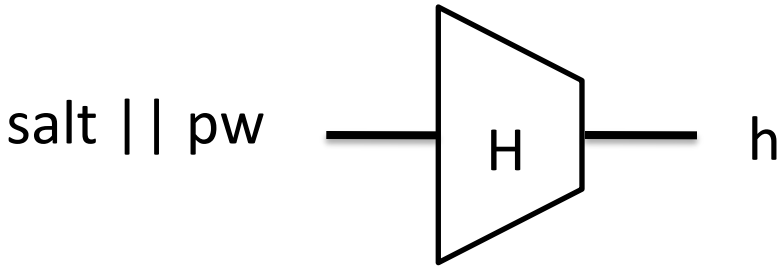
Passwords



tom	princess
alice	123456
bob	p@ssword!

Password hashing

Password hashing. Choose random salt and store (salt,h) where:



The idea: Attacker, given (salt,h), should not be able to recover pw

Or can they?

For each guess pw':
    If H(salt||pw') = h then
        Ret pw'

Rainbow tables speed this up in practice by way of precomputation. Large salts make rainbow tables impractical

```
rist@seclab-laptop1:~/work/teaching/642-fall-2011/slides$ openssl speed sha1
Doing sha1 for 3s on 16 size blocks: 4109047 sha1's in 3.00s
Doing sha1 for 3s on 64 size blocks: 3108267 sha1's in 2.99s
Doing sha1 for 3s on 256 size blocks: 1755265 sha1's in 3.00s
Doing sha1 for 3s on 1024 size blocks: 636540 sha1's in 3.00s
Doing sha1 for 3s on 8192 size blocks: 93850 sha1's in 3.00s
OpenSSL 1.0.0d 8 Feb 2011
```

```
rist@seclab-laptop1:~/work/teaching/642-fall-2011/slides$ openssl speed aes-128-
cbc
Doing aes-128 cbc for 3s on 16 size blocks: 27022606 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 64 size blocks: 6828856 aes-128 cbc's in 2.99s
Doing aes-128 cbc for 3s on 256 size blocks: 1653364 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 1024 size blocks: 438909 aes-128 cbc's in 2.99s
Doing aes-128 cbc for 3s on 8192 size blocks: 54108 aes-128 cbc's in 3.00s
OpenSSL 1.0.0d 8 Feb 2011
```

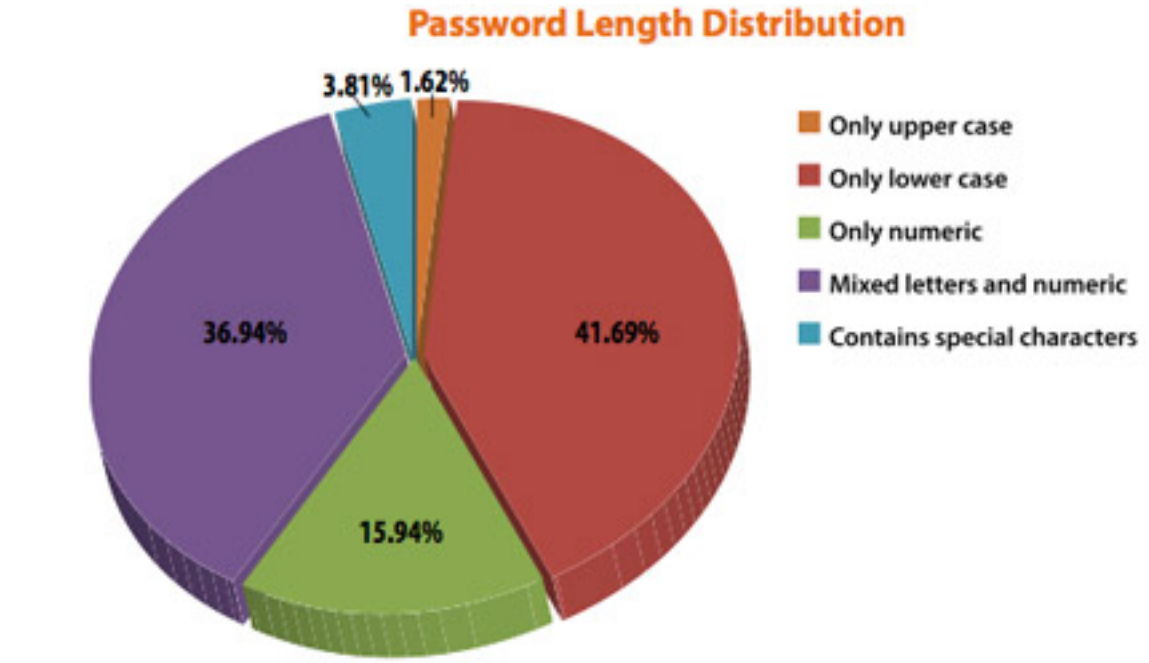
Say c = 4096. Generous back of envelope* suggests that in 1 second, can test 252 passwords and so a naïve brute-force:

6 numerical digits	10^6 = 1,000,000	~ 3968 seconds
6 lower case alphanumeric digits	36^6 = 2,176,782,336	~ 99 days
8 alphanumeric + 10 special symbols	72^8 = 722,204,136,308,736	~ 33million days

* I did the arithmetic…

Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Rank	Password	Number of Users with Password (absolute)
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856



From an Imperva study of released RockMe.com password database 2010

AshleyMadison hack: 36 million user hashes

Salts + Passwords hashed using bcrypt with c = 2^{12} = 4096

4,007 cracked directly with trivial approach

CynoSure analysis: **11 million** hashes cracked

>630,000 people used usernames as passwords

MD5 hashes left lying around accidentally

http://cynosureprime.blogspot.com/2015/09/csp-our-take-on-cracked-am-passwords.html

Ashley Madison in good company

⋮

32.6 million leaked (2012)
32.6 million recovered (plaintext!)

6.5 million leaked (2012)
5.85 million recovered in 2 weeks

442,832 leaked (2012)
442,832 recovered

36 million accounts leaked (2013)
Encrypted, but with ECB mode

⋮

HACKERS RECENTLY LEAKED **153 MILLION** ADOBE USER EMAILS, ENCRYPTED PASSWORDS, AND PASSWORD HINTS.

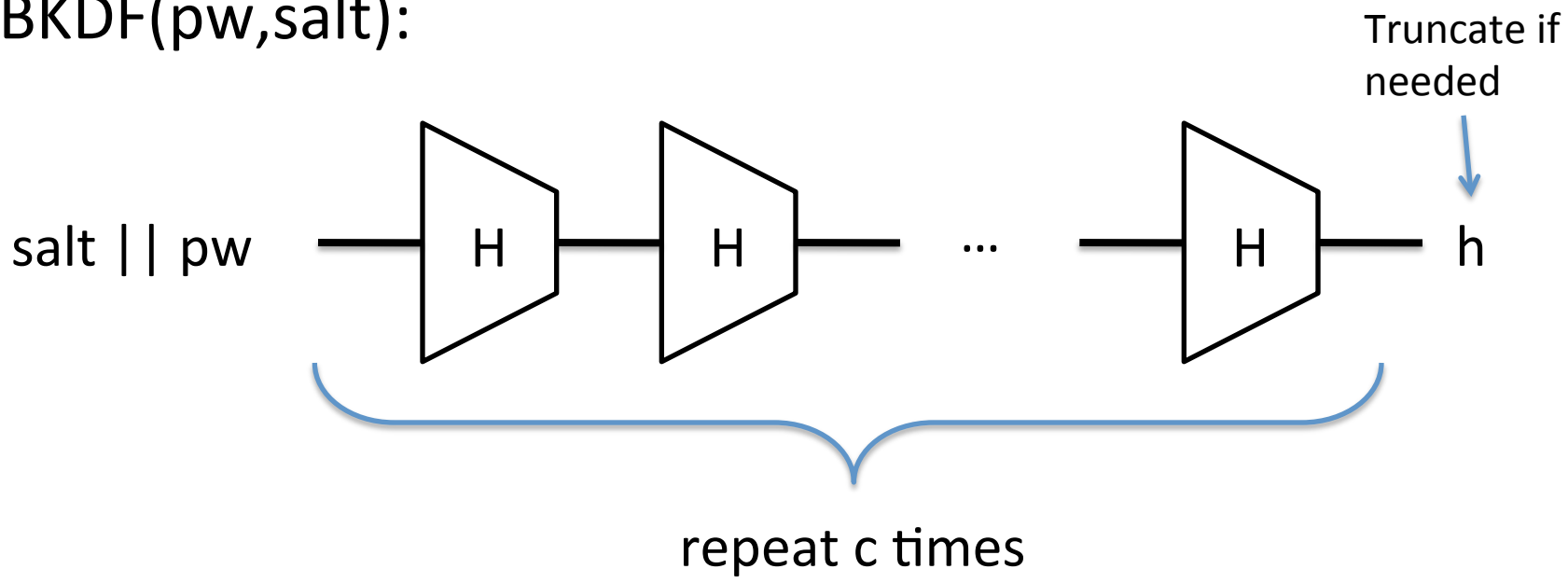
ADOBE ENCRYPTED THE PASSWORDS IMPROPERLY, MISUSING BLOCK-MODE 3DES. THE RESULT IS SOMETHING WONDERFUL:

USER PASSWORD		HINT
4e18acc1ab27a2d6		WEATHER VANE SWORD
4e18acc1ab27a2d6		
4e18acc1ab27a2d6	a0a2876eb1ea1fca	NAME 1
8babb6299e06eb6d		DUH
8babb6299e06eb6d	a0a2876eb1ea1fca	
8babb6299e06eb6d	85e9da81a8a78adc	57
4e18acc1ab27a2d6		FAVORITE OF 12 APOSTLES
1ab29ae86da6e5ca	7a2d6a0a2876eb1e	WITH YOUR OWN HAND YOU HAVE DONE ALL THIS
a1f9b2b6299e7a2b	eadec1e6ab797397	SEXY EARLOBES
a1f9b2b6299e7a2b	617ab0277727ad85	BEST TOS EPISODE
39738b7adb0b8af7	617ab0277727ad85	SUGARLAND
1ab29ae86da6e5ca		NAME + JERSEY #
877ab7889d3862b1		ALPHA
877ab7889d3862b1		
877ab7889d3862b1		
877ab7889d3862b1		OBVIOUS
877ab7889d3862b1		MICHAEL JACKSON
38a7c9279cadeb44	9dca1d79d4dec6d5	
38a7c9279cadeb44	9dca1d79d4dec6d5	HE DID THE MASH, HE DID THE
38a7c9279cadeb44		PURLOINED
e8ae5745a2b7af7a	9dca1d79d4dec6d5	FAV WATER-3 POKEMON

THE GREATEST CROSSWORD PUZZLE IN THE HISTORY OF THE WORLD

Password-based Key Deriviation (PBKDF)

PBKDF(pw,salt):



PKCS#5 standardizes PBKDF1 and PBKDF2, which are both hash-chain based.

Only slows down by a factor of c

scrypt, argon2: memory-hard hashing functions

Facebook password onion

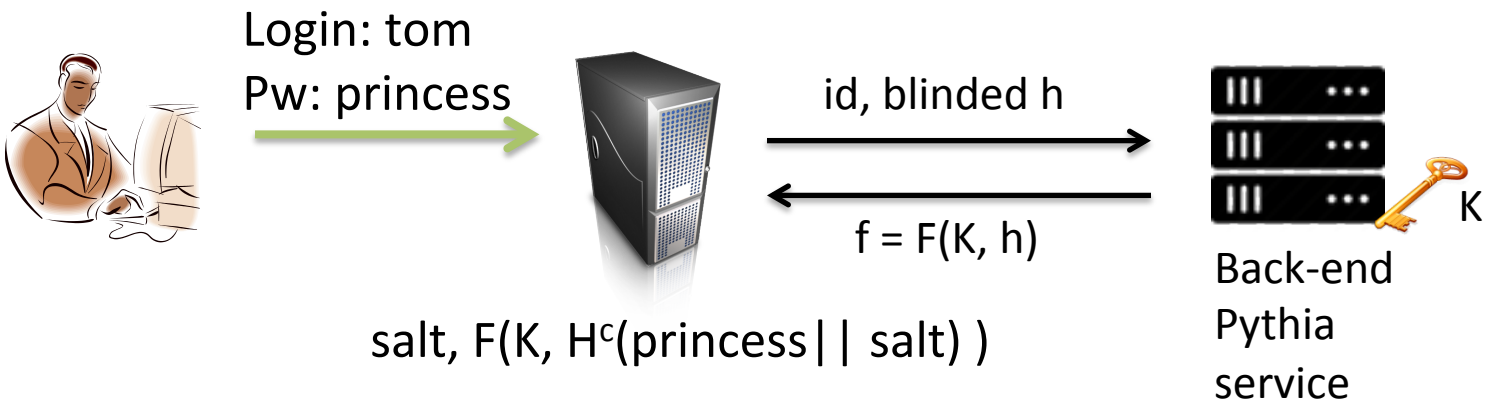
```
$cur  = 'password'
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```

Evolution of their password hashing over time

Limitations:

- Can't rotate secret
- Can't do cryptographic erasure for compromise clean-up

The Pythia PRF Service



- Stronger password privacy properties via blinding of values derived from password
- Prototype and paper:
  http://pages.cs.wisc.edu/~ace/pythia.html

Another application of PBKDFs: PW-based encryption

Enc(pw,M):
salt
K = PBKDF(pw,salt)
C' = AEnc(K,M)
Return (salt,C')

Dec(pw,salt||C):
K = PBKDF(pw,salt)
M = ADec(K,C)
Return M

Here En is a normal symmetric encryption scheme (CBC+HMAC)

Attacks?

Summary

- Hash functions
  - Merkle-Damgard domain extension
  - Compression functions from block ciphers
  - Length-extension attacks & HMAC
- Passwords
  - Brute-force attacks
  - PBKDFs slow down attacks
  - Split-state architectures help (Facebook & Pythia)
  - PW-based encryption (PBKDF + AEnc)