Honey Encryption:

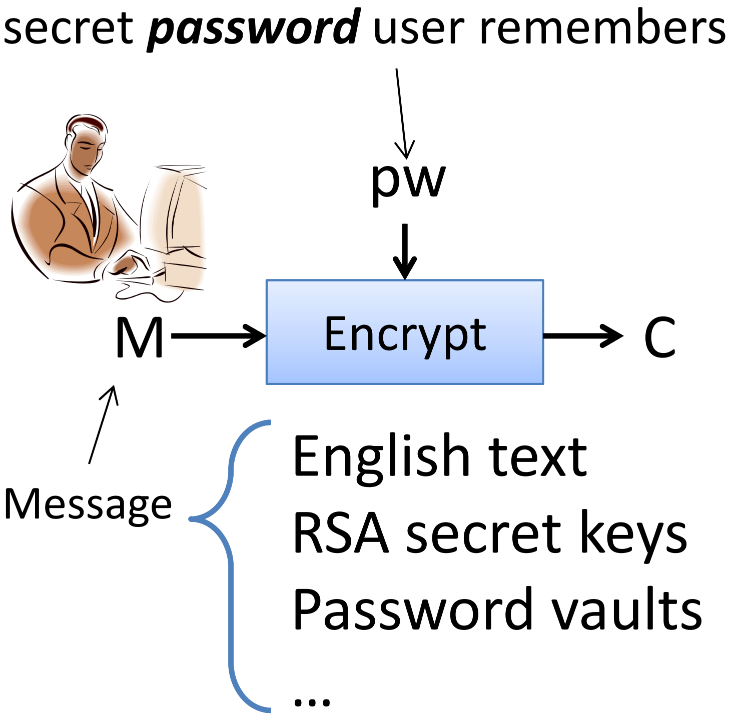
Security Beyond the Brute-force Bound

Ari Juels
Cornell Tech

Thomas Ristenpart
University of Wisconsin

Encryption for which decrypting a ciphertext with any number of ***wrong*** keys yields fake, but plausible, plaintexts

Password-based encryption



secret ***password*** user remembers

pw

M

Encrypt

C

Message

English text

RSA secret keys

Password vaults

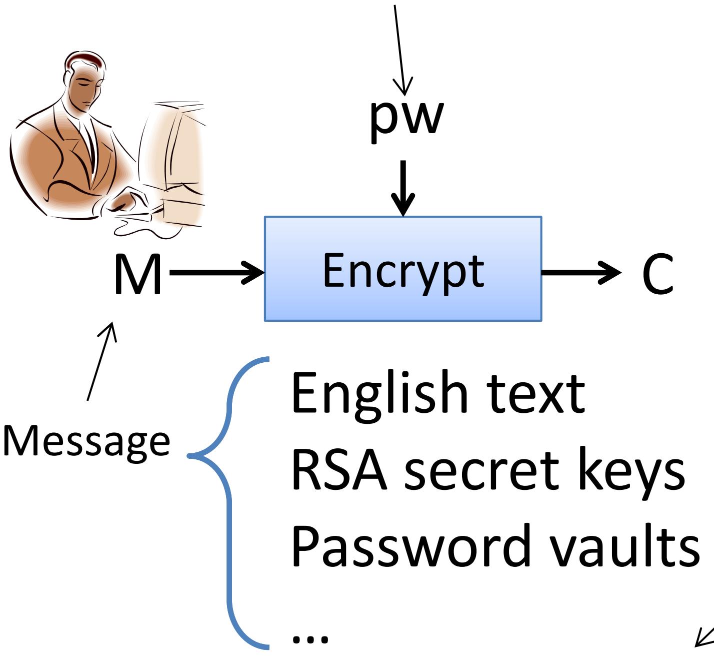
…

PKCS#5 is dominant standard

Password-based encryption

secret ***password*** user remembers

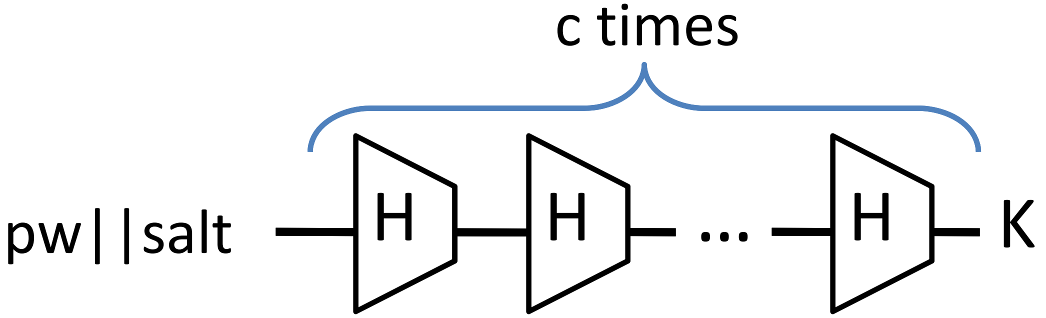
PKCS#5 is dominant standard



English text
RSA secret keys
Password vaults
…

```
Encrypt(pw, M)
    salt $\leftarrow\$ \{0,1\}^{128}$
    $K \leftarrow H^c(pw \,||\, salt)$
    $C \leftarrow K \oplus M$
    Return (salt,C)
```

```
Decrypt(pw, salt,C )
    $K \leftarrow H^c(pw \,||\, salt)$
    $M \leftarrow K \oplus C$
    Return M
```

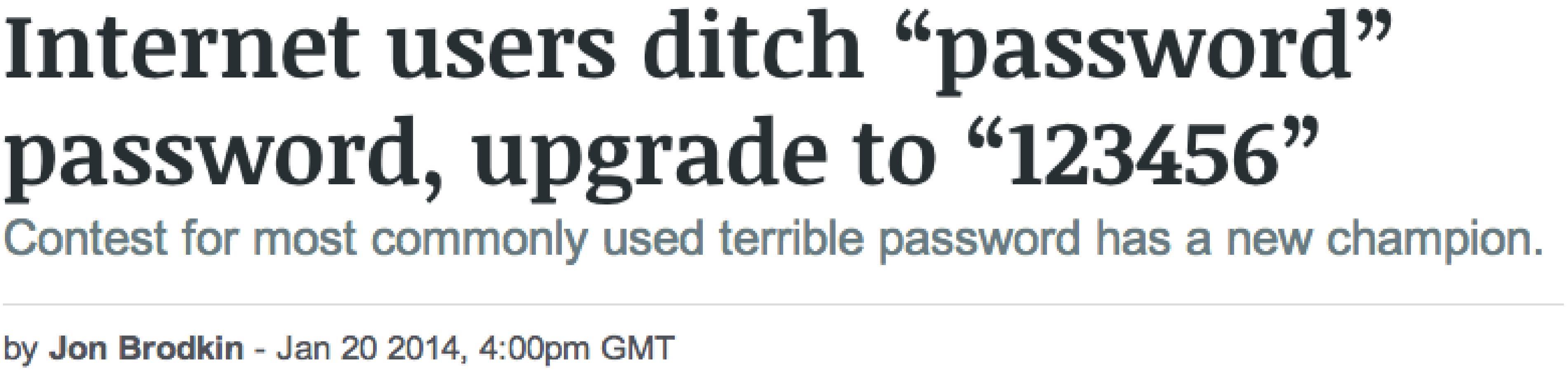


Cryptographic hash function H
(H = SHA-256, SHA-512, etc.)

Common choice is c = 10,000

Why hash chains and salts?

Slow down ***brute-force attacks***



Internet users ditch "password" password, upgrade to "123456"

Contest for most commonly used terrible password has a new champion.

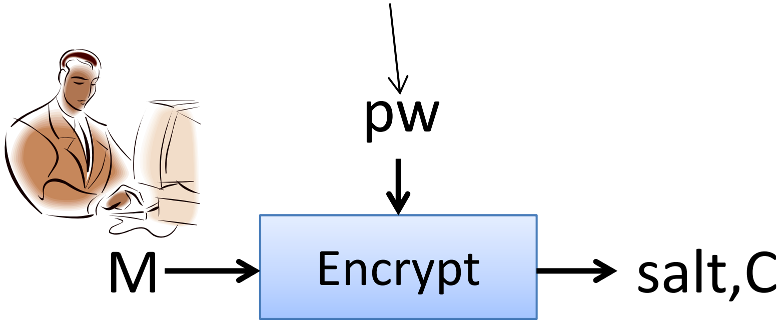
by **Jon Brodkin** - Jan 20 2014, 4:00pm GMT

[Bonneau 2012] studied 69 million Yahoo! Passwords
1.1% of users pick same password

People choose weak passwords

Brute-force attacks

pw likely to fall in short sequence of guesses pw_1,pw_2,pw_3, …



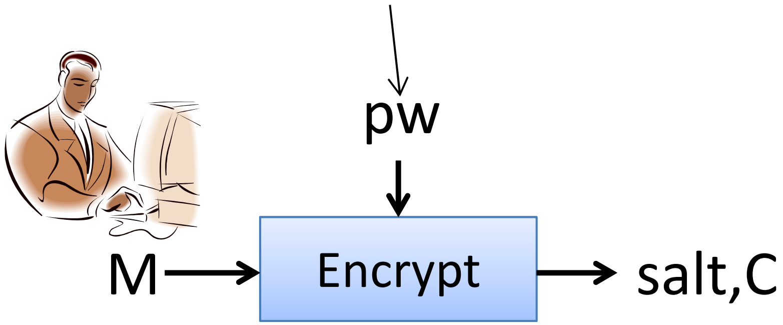
salt,C

Step 1: Trial decryptions

M_1 <- Decrypt(pw_1,salt,C)
M_2 <- Decrypt(pw_2,salt,C)
M_3 <- Decrypt(pw_3,salt,C)
…

Brute-force attacks

pw likely to fall in short sequence of guesses pw_1, pw_2, pw_3, …



Say M is unknown ASCII text encoded in binary

Many bytes won’t be valid ASCII characters, let alone “look” like English text.

salt,C

Step 1: Trial decryptions

M_1 <- $H^c(pw_1 \,||\, salt) \oplus C$

M_2 <- $H^c(pw_2 \,||\, salt) \oplus C$

M_3 <- $H^c(pw_3 \,||\, salt) \oplus C$

…

Step 2: Find true plaintext

~~M_1 = $&%ff1 31f^~~

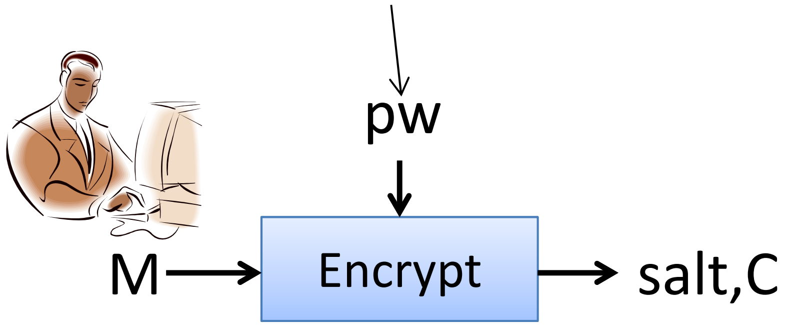
~~M_2 = hgjk!alc&ewj~~

M_3 = copenhagen

…

Brute-force attacks

pw likely to fall in short sequence of guesses pw_1, pw_2, pw_3, …



Analyses ignore Step 2, conservatively assuming it is trivial for attacker

Say M is unknown prime number encoded as integer

- Hash chain slows attack by factor of c
- Salt prevents rainbow tables, provide separation between users

Primality tests will eliminate majority of candidate plaintexts

salt,C

Step 1: Trial decryptions

$M_1 \leftarrow H^c(pw_1 \,||\, \text{salt}) \oplus C$

$M_2 \leftarrow H^c(pw_2 \,||\, \text{salt}) \oplus C$

$M_3 \leftarrow H^c(pw_3 \,||\, \text{salt}) \oplus C$

…

Step 2: Find true plaintext

~~$M_1 = 6123410$~~

$M_2 = 1299827$

~~$M_3 = 7321162$~~

…

The Brute-force Bound

Say pw has min-entropy m (most likely password has probability $1/2^m$)

Corollary [BRT12]: Encrypt is such that for all IND-CPA adversaries A

$$\frac{t}{c2^m} \leq \text{Adv}(\text{Encrypt},A) \leq \frac{t}{c2^m}$$

where t = cq for some q is the number of queries to H modeled as a RO, and ignoring small constants and negligible terms

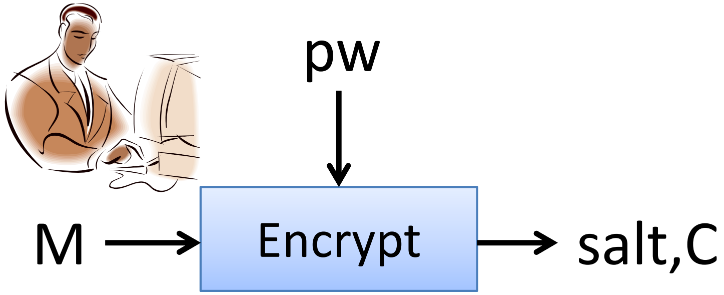
[B12]: most likely password has prob. 1.1% meaning $m \approx 6.5$

So t > 1,000,000 makes the above bound close to 1 for c = 10,000

(A) Existing countermeasures help slow down attacks but only ensure security for high-entropy pw

(B) Best we can do when targeting IND-CPA

Beyond the brute-force bound?



<u>Key intuition:</u>
Step 2 may be hard for attacker for some message distributions

Say M is uniformly distributed bit string

Seems impossible to distinguish!

salt,C

<u>Step 1: Trial decryptions</u>

$M_1 \leftarrow H^c(pw_1 \,||\, salt) \oplus C$

$M_2 \leftarrow H^c(pw_2 \,||\, salt) \oplus C$

$M_3 \leftarrow H^c(pw_3 \,||\, salt) \oplus C$

…

???

<u>Step 2: Find true plaintext</u>

$M_1 = 101010101$

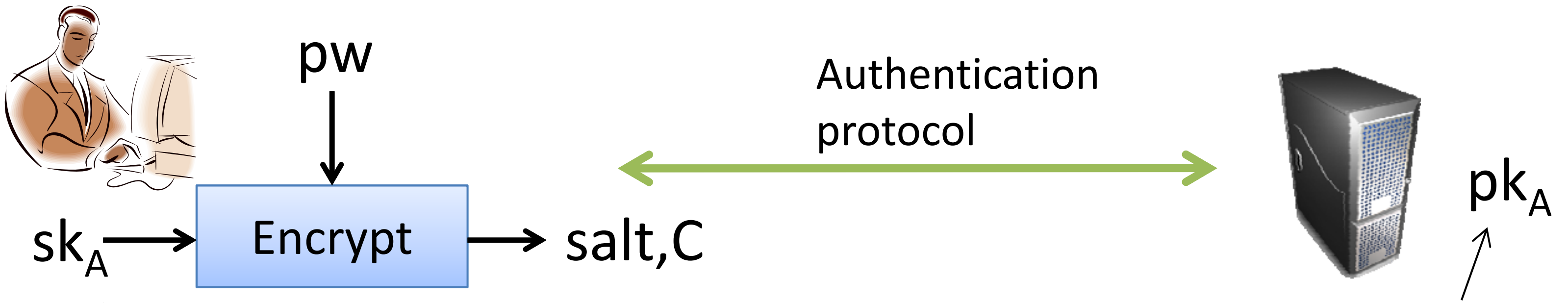
$M_2 = 100111010$

$M_3 = 010101011$

…

Application: compromise resilience for credentials

[Hoover, Kausik 99]



pw

sk_A → Encrypt → salt,C

Authentication protocol

pk_A

__RSA secret exponent d encoded as (uniform) bit string__

Modulus N left in clear.

Decrypt only when user wants to authenticate

Public-key for client stored ***only*** at server

If attacker just obtains C, best strategy is online attack using M_1 , M_2 , … . Significantly harder to mount than offline attack

<u>Step 1: Trial decryptions</u>

M_1 <- $H^c(pw_1 \,||\, salt) \oplus C$

M_2 <- $H^c(pw_2 \,||\, salt) \oplus C$

M_3 <- $H^c(pw_3 \,||\, salt) \oplus C$

…

<u>Step 2: Find true plaintext</u>

???

$M_1 = 101010101$

$M_2 = 100111010$

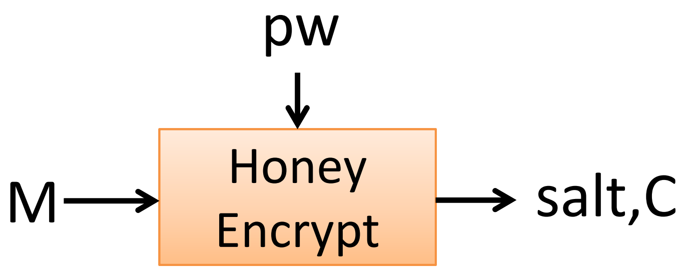
$M_3 = 010101011$

…

Decoys in computer security

- In computer security, we have "honey objects":
  - Honeypots, honeytokens, honey accounts
  - Decoy documents [BHKS09]
  - Kamoflauge system [BBBB10]
  - Honeywords for password hashing [JR13]
- Cryptographic camouflage [Hoover, Kausik 99]

We introduce Honey Encryption (HE)



Encryption schemes tailored to specific message distributions

Secure in [BRT12] sense (use hash chains and salting)

Provable message-recovery security ***beyond brute-force bound.***
We will show ***optimal security*** in some cases***:***

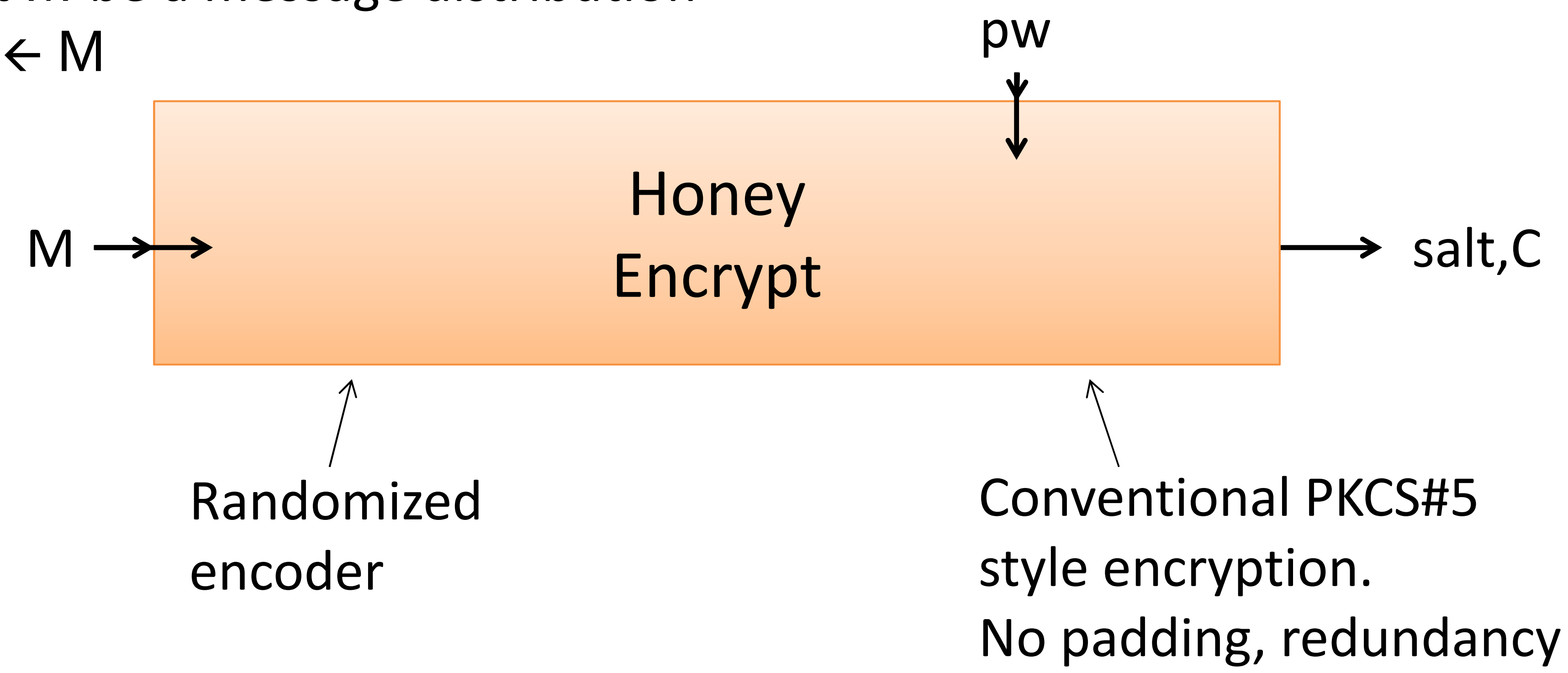
$$\Pr[\text{message recovery}] < \frac{1}{2^m}$$

Probability of guessing password

A framework for HE schemes

Let M be a message distribution

$M \leftarrow M$



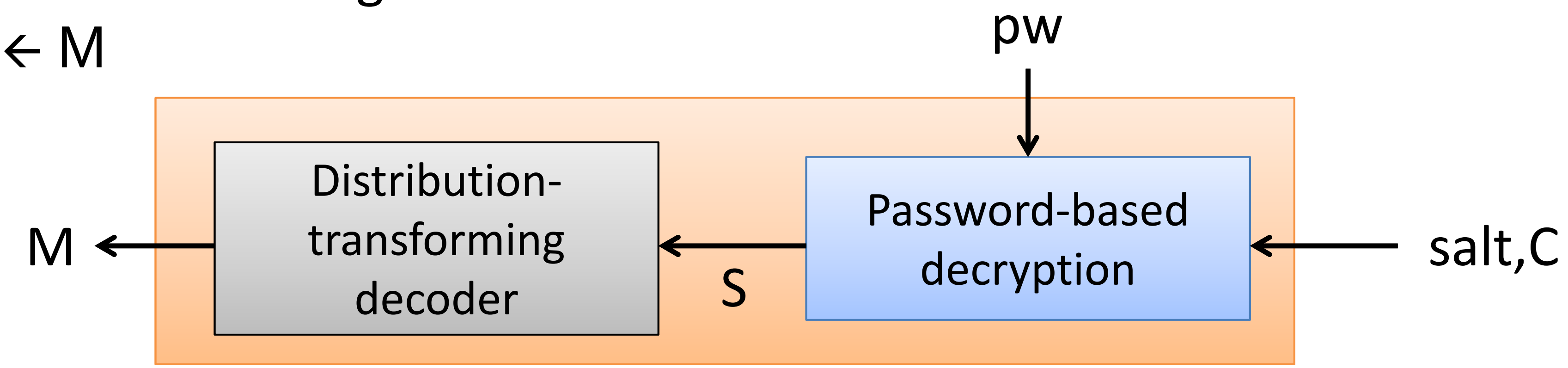
Randomized encoder

Conventional PKCS#5 style encryption.
No padding, redundancy

A framework for HE schemes

Let M be a message distribution

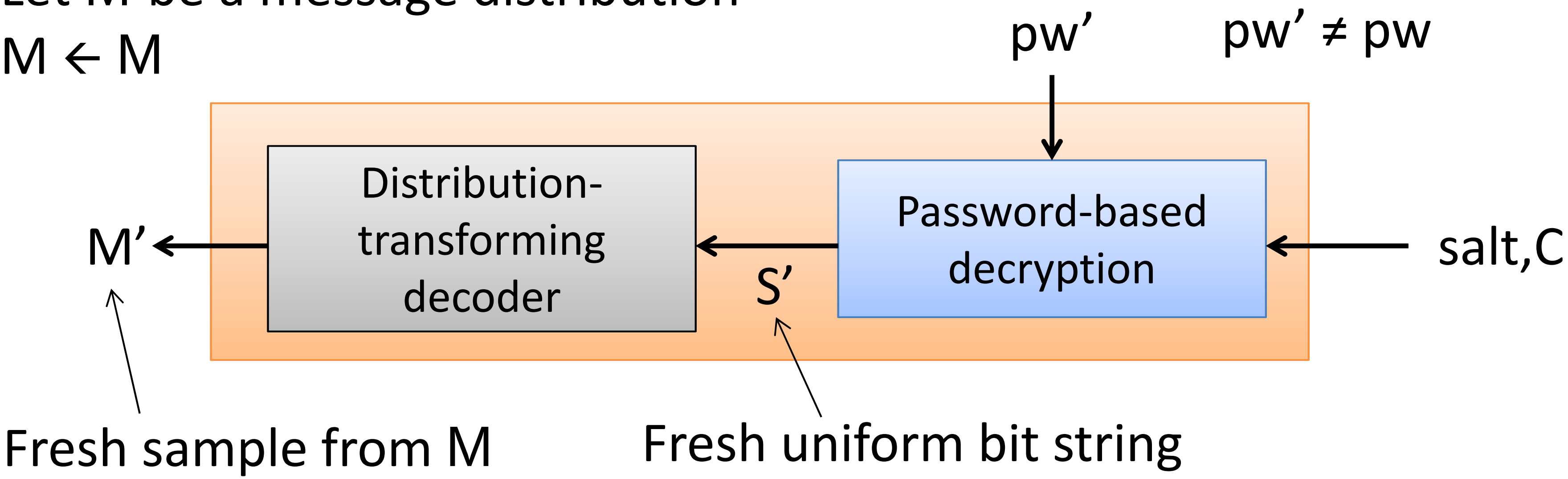
M ← M



A framework for HE schemes

Let M be a message distribution

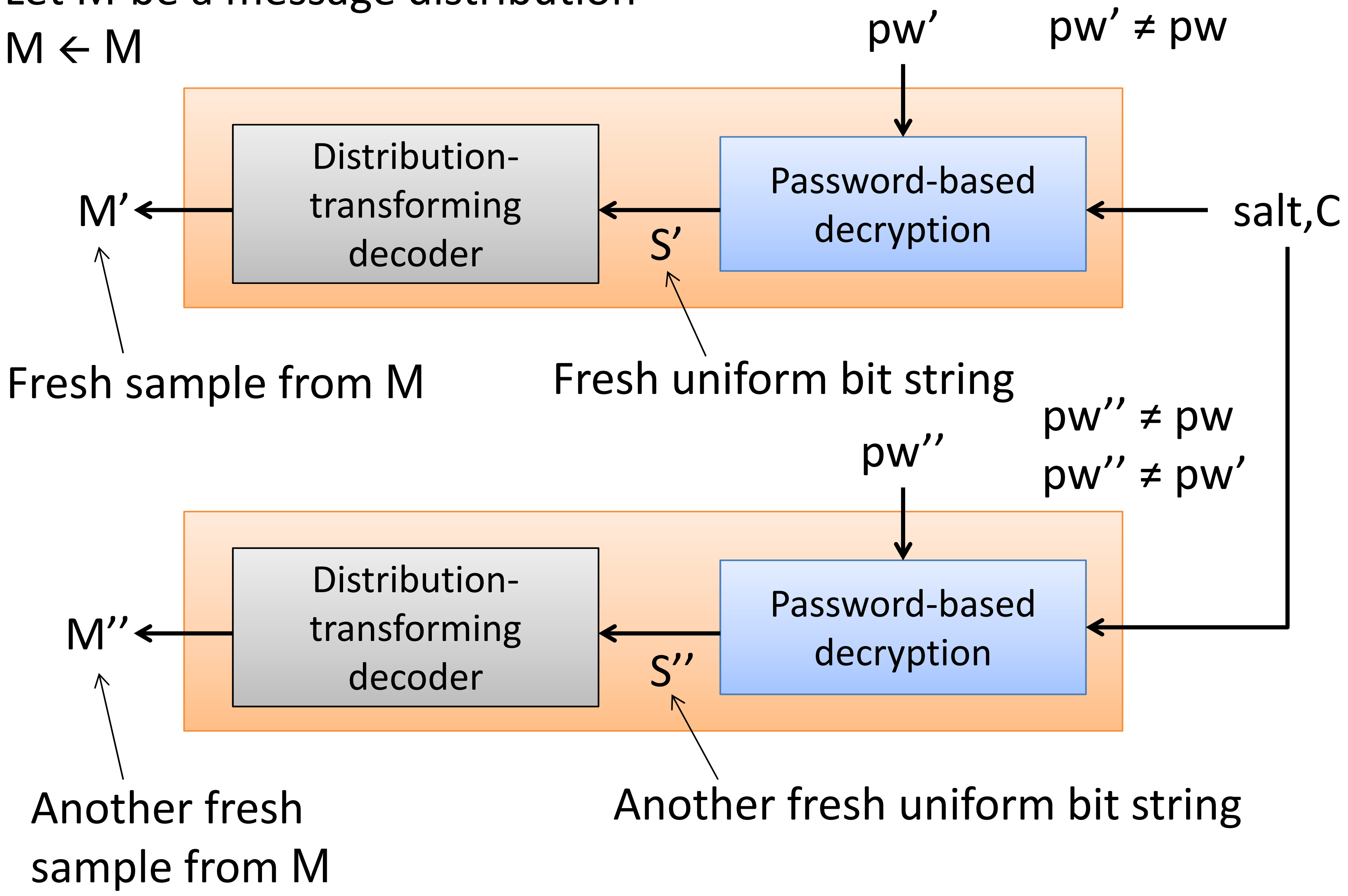
M ← M



A framework for HE schemes

Let M be a message distribution

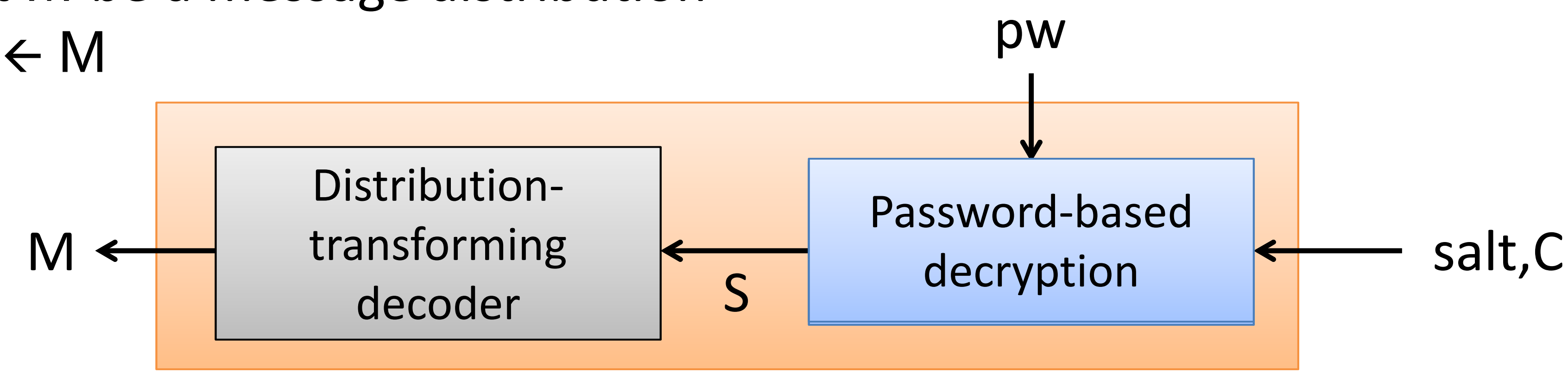
M ← M


pw’
pw’ ≠ pw
M’
Distribution-transforming decoder
S’
Password-based decryption
salt,C
Fresh sample from M
Fresh uniform bit string
pw’’
pw’’ ≠ pw
pw’’ ≠ pw’
M’’
Distribution-transforming decoder
S’’
Password-based decryption
Another fresh sample from M
Another fresh uniform bit string

A framework for HE schemes

Let M be a message distribution

M ← M



<u>Intuition</u>:

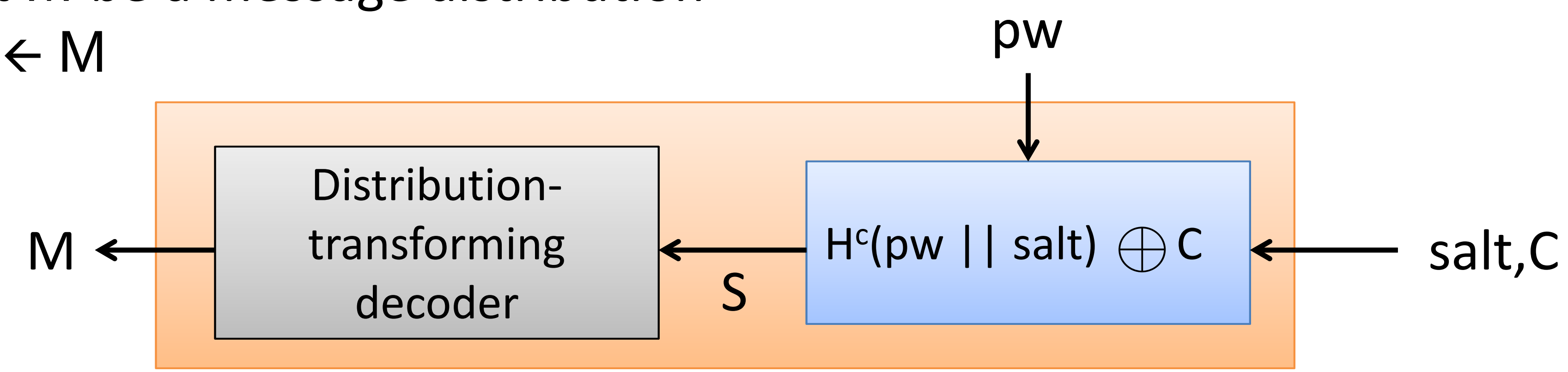
(1) Decoder is sampler using input as string of randomness

✓(2) Decryption under different keys yields uniform bits

A framework for HE schemes

Let M be a message distribution

M ← M


pw
Distribution-transforming decoder
$H^c(pw \,||\, salt) \oplus C$
M
S
salt,C

DTE = (**encode**, **decode**) designed for particular M

encode randomized **decode** deterministic

Toy example M

Message	Probability
eurocrypt	1/4
tivoligarden	1/2
Copenhagen	1/4

encode(M)

```
If M = tivoligarden   then  b ← {0,1} ; Return 0b
If M = eurocrypt      then  Return 11
If M = Copenhagen     then  Return 10
```

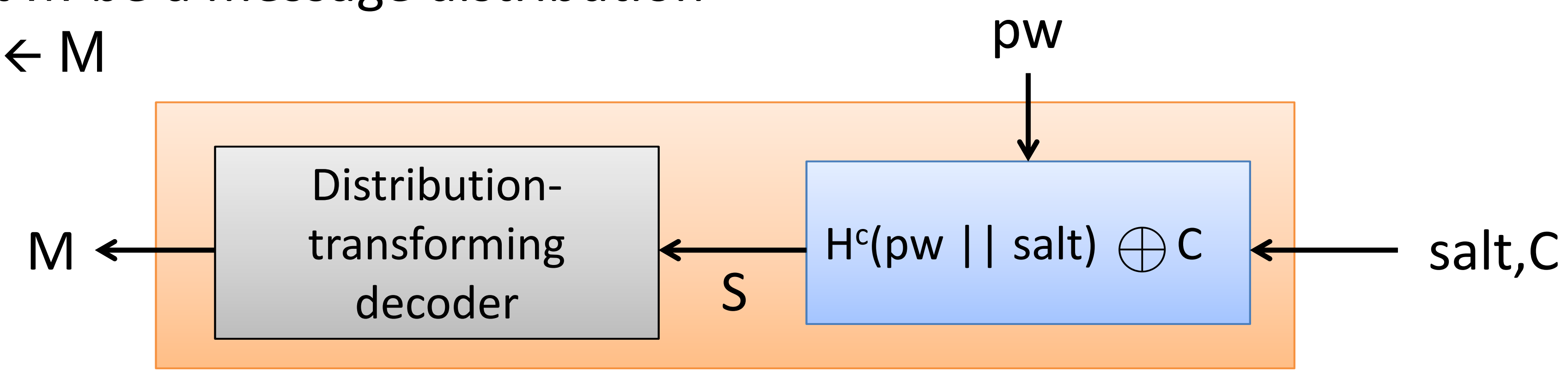
decode via look-up table

Huffman coding without compression

A framework for HE schemes

Let M be a message distribution

M ← M



pw

Distribution-transforming decoder

$H^c(pw \,||\, salt) \oplus C$

M

S

salt,C

DTE = (**encode**, **decode**) designed for particular M

encode randomized **decode** deterministic

DTE for M being uniform n-bit prime numbers

```
Encode(M)
$X_1,\ldots,X_t \leftarrow\$ (Z_n)^t$
Find 1st i with $X_i$ prime
$X_i \leftarrow M$
Return $S = X_1,\ldots,X_t$
```

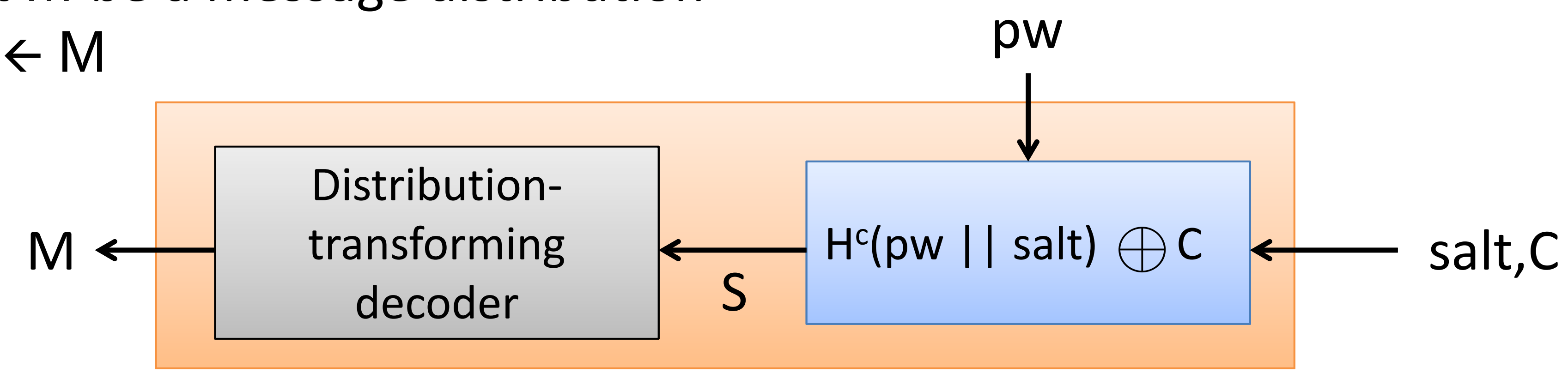
```
Decode(S)
$X_1,\ldots,X_t \leftarrow S$
Find 1st i with $X_i$ prime
$M \leftarrow X_i$
Return M
```

Classic rejection-sampling prime generation

A framework for HE schemes

Let M be a message distribution

M ← M



DTE = (**encode**, **decode**) designed for particular M

encode randomized **decode** deterministic

Many DTEs only approximate correct distribution. Secure if:

```
M ← M
S ←$ encode(M)
Return (M,S)
```

≈

```
S ←$ {0,1}^s
M ← decode(S)
Return (M,S)
```

Honey encryption so far

- Intuition: decryption with wrong password gives plausible plaintext
- Applications in resilience to compromise of encrypted credentials
- Framework:

  (1) Distribution-transforming encoders (DTEs)

  (More examples in paper!)

  (2) Conventional password-based encryption

Security for honey encryption

Never worse than existing password-based encryption

Inherit provable security in sense of [BRT12]

We analyze ***message recovery (MR) security***

MR game:

```
M ←$ M
pw ←$ P
salt,C ←$ HEnc(pw,M)
M’ ←$  A(salt,C)
Ret (M=M’)
```

M is message distribution

P   is password distribution

Example: HE for uniform primes

M is uniform n-bit primes

P   has min-entropy m

HE scheme as described before

Thm (informal). For any MR attacker A

Pr[wins MR game]   < $1/2^m$

(ignoring smaller terms)

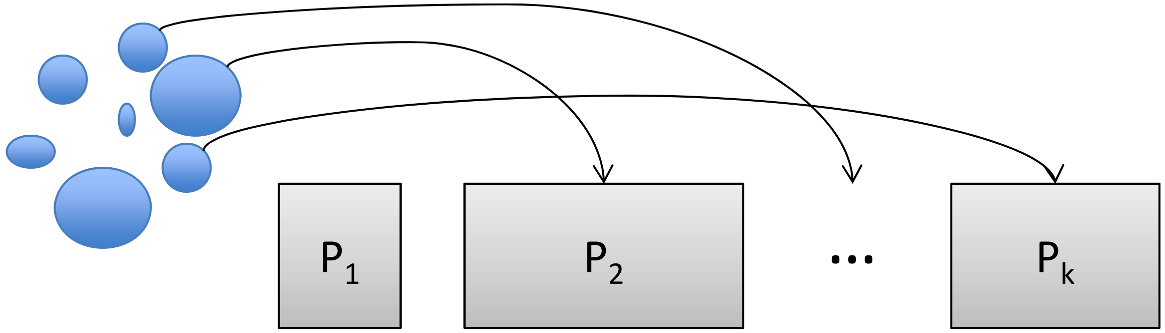
Intuition for proofs

Allow information-theoretic adversaries (also unbounded RO queries)

Adversary outputs most probable message

After applying DTE security, can bound advantage via ***balls-and-bins game***

Decryption of challenge ciphertext with each password
is independent ball throw into bins (when H is RO)



Balls are passwords
of size equal to their
probability

Bins are messages of size
equal to their probability under decode

Adversary’s advantage maximized by
picking heaviest bin at end of game

Expected maximum load E[L] is
expected weight of heaviest bin

Well-studied for some settings

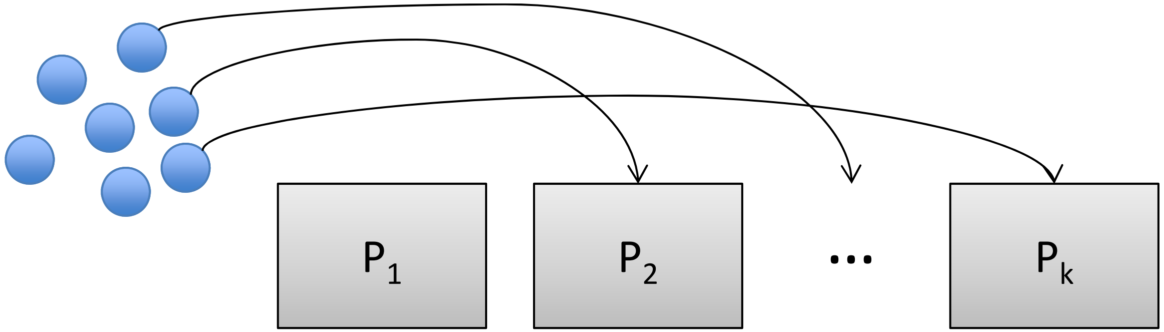
Intuition for proofs

Allow information-theoretic adversaries (also unbounded RO queries)
Adversary outputs most probable message
After applying DTE security, can bound advantage via ***balls-and-bins game***

Decryption of challenge ciphertext with each password is independent ball throw into bins (when H is RO)

Balls are passwords of size equal to their probability

(Equal weight $1/2^m$ for uniform distribution)



Bins are messages of size equal to their probability under decode

(Equal weight $1/2^n$ for uniform distribution)

Adversary's advantage maximized by picking heaviest bin at end of game

Expected maximum load E[L] is expected weight of heaviest bin

Well-studied for some settings

For prime number HE:

$k = 2^n$ and $k^2 << 2^m$

Pr[wins MR game] < E[L] = $1/2^m$ + negl

In the paper…

- More DTEs, more HE constructions
- More general balls-and-bins analyses
- Discussion of extensions
  - dealing with password typos
  - detecting online brute-force attacks
- Discussion of limitations of HE

Summary

Def. ***Honey Encryption***

Encryption for which decrypting a ciphertext with any number of *wrong* keys yields fake, but plausible, plaintexts

A framework for building and analyzing HE schemes using *Distribution-Transforming Encoders*

Moving forward:

DTEs for more complex distributions

- Password vaults

Further analyses, constructions

- Standard model
- Sharpened balls-and-bins bounds