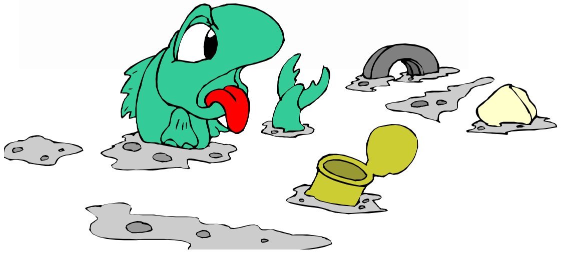
KEY DERIVATION WITHOUT ENTROPY WASTE

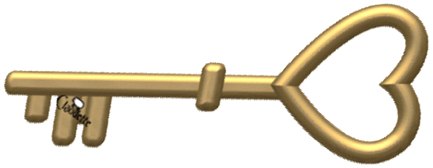


Yevgeniy Dodis

New York University

Based on joint works with B. Barak, H. Krawczyk, O. Pereira, K. Pietrzak, F-X. Standaert, D. Wichs and Y. Yu

Key Derivation



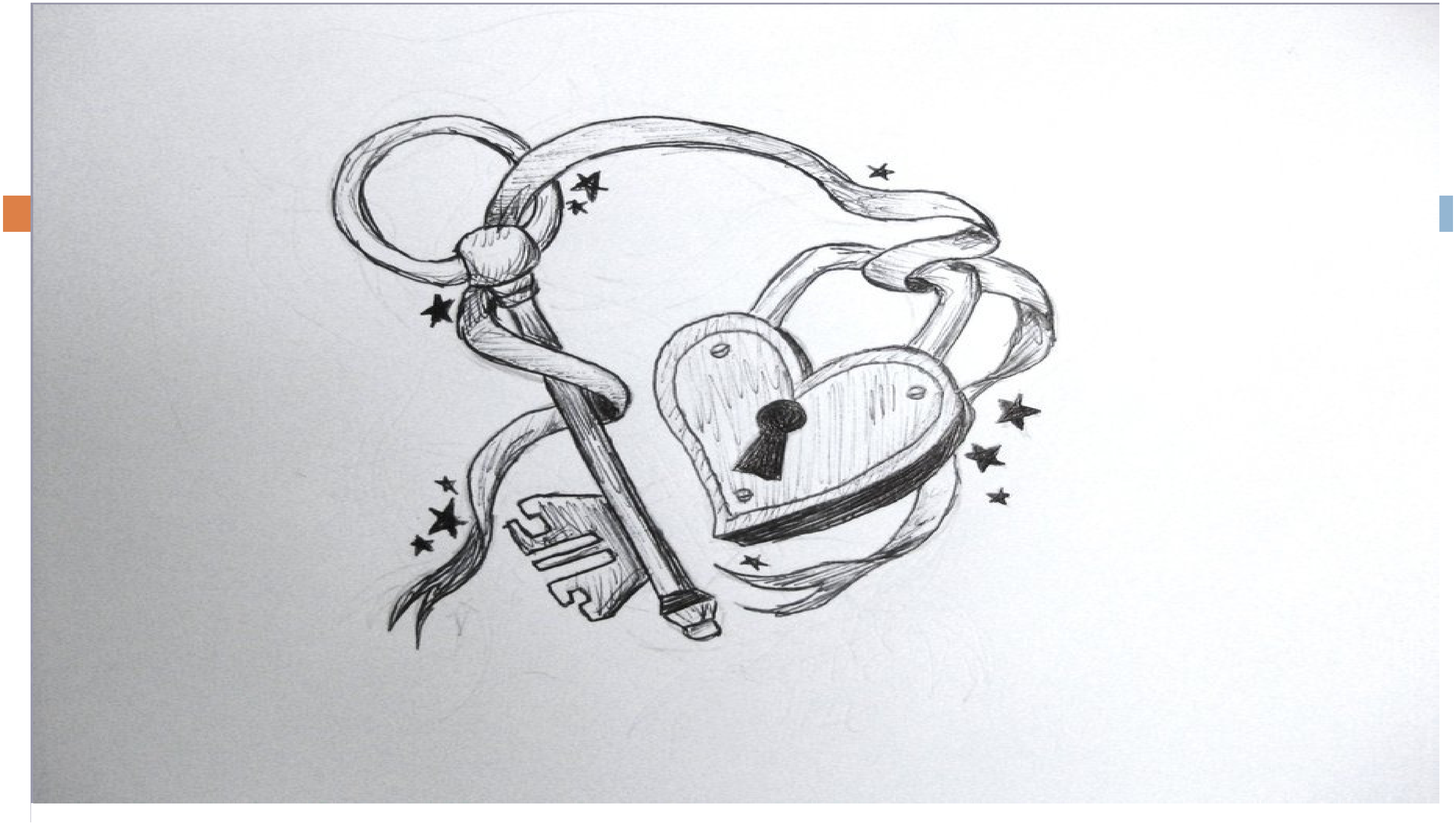
- Setting: application P needs m–bit secret key R
- Theory: pick *uniformly random R* $\leftarrow \{0,1\}^m$
- Practice: have ”imperfect randomness” $X \in \{0,1\}^n$
  - physical sources, biometric data, partial key leakage, extracting from group elements (DH key exchange), …
- Need a “bridge”: *key derivation function* (KDF) h: $\{0,1\}^n \rightarrow \{0,1\}^m$ s.t. $R = h(X)$ is “good” for P
  - … *only* assuming X has **“minimal entropy”** k

Dreaming Big

- Question 1: minimal entropy k enough to achieve "real security" ≈ "ideal security" for P?
  - Dream 1: can get $k \approx m$ (no "entropy loss") !

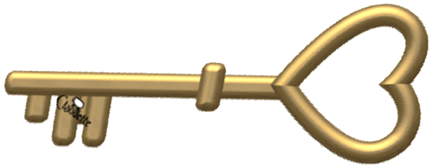
Question/Dream 3: can we ever hope to achieve **comparable** security **without entropy loss** ?!

- Question 2: best security degradation when $k \approx m$?
  - Dream 2: (almost) no security degradation !



- Note: we design h but must work for any (n, k)-source X

Formalizing the Problem



- Ideal Model: pick *uniform* $R \leftarrow U_m$ as the key
  - Assume P is ε–secure against certain class of attackers ***A***
- Real Model: use $R = h(X)$ as the key, where
  - min-entropy$(X) = \mathbb{H}_\infty(X) \geq k$ $\quad$ ($\Pr[X = x] \leq 2^{-k}$, for all x)
  - h: $\{0,1\}^n \rightarrow \{0,1\}^m$ is a (*carefully designed*) KDF
- Goal: prove that P is ε'–secure in the real model (against same/similar class of attackers ***A***)
  - Note: we design h but must work for any (n, k)-source X
- What is the smallest ε'???

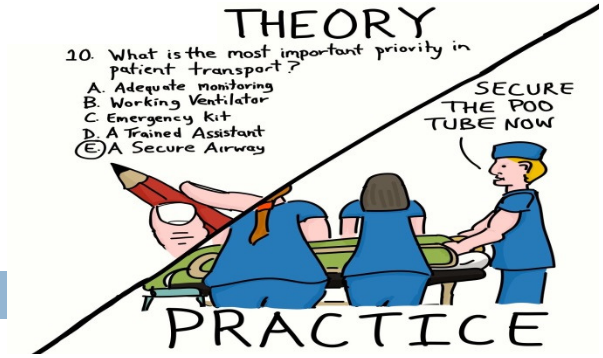
Dreaming Big, formally

- Question 1: minimal k (call it k^*) to get $\varepsilon' = 2\varepsilon$?
  - Dream 1: can get $k^* \approx m$ (no "entropy loss") !

Question/Dream 3: can we ever hope to achieve $\varepsilon' = O(\varepsilon)$ security when $k \approx m$ (no entropy loss) ?!

- Question 2: smallest ε' (call it ε^*) when $k = m$?
  - Dream 2: can get $\varepsilon^* = O(\varepsilon)$ (no security degradation) !

Theory vs. Practice



- Practice: heuristic key derivation (h = SHA,MD5,…)
  - common belief among practitioner: Dream 3 is TRUE !
- *Amazing* (heuristic) bound in "random oracle" model:

$$\boxed{\varepsilon' \leq \varepsilon + \varepsilon\cdot 2^{m-k}}$$

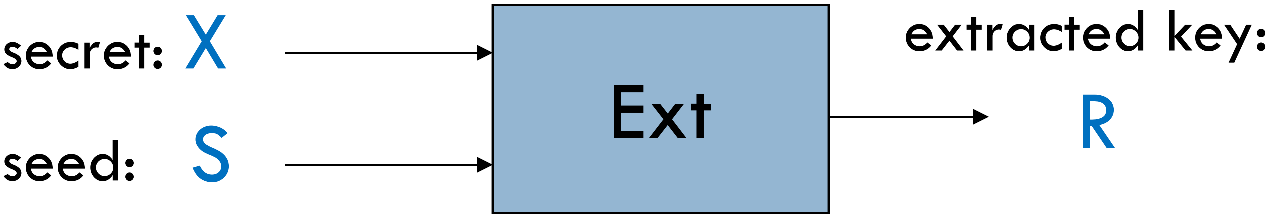
  - "implies" $\varepsilon^* = 2\varepsilon$ and $k^* = m$ at the same time!
- Despite lack of "practical" attacks, lots of (valid) criticism [DHK$^+$04,Kra10,BDK$^+$11]
- How close can we come in theory (and practice ☺)?

Extractors

- <u>Tool:</u> Randomness Extractor [NZ96].
  - Input: a *weak secret* X and a *uniformly random seed* S.
  - Output: *extracted key* $R = \mathbf{Ext}(X; S)$.
  - R is uniformly random, even conditioned on the seed S.

$$(\mathbf{Ext}(X; S), S) \approx (\text{Uniform}, S)$$

- **Many uses in complexity theory and cryptography.**
  - Well beyond key derivation (de-randomization, etc.)



(Seeded) Extractors

- Tool: Randomness Extractor [NZ96].
  - Input: a *weak secret* X and a *uniformly random seed* S.
  - Output: *extracted key* R = **Ext**(X; S).
  - R is uniformly random, even conditioned on the seed S.

$$(\mathbf{Ext}(X; S), S) \approx (\text{Uniform}, S)$$

- (k,δ)-extractor: given any secret (n,k)-source X, outputs m secret bits "δ–fooling" any distinguisher ***D***:

statistical distance

$$|\Pr[D(\mathbf{Ext}(X; S), S) = 1] - \Pr[D(U_m, S) = 1]| \leq \delta$$

Extractors as KDFs

- Lemma: for any ε-secure P needing an m–bit key, (k,δ)-extractor is a KDF yielding security $\varepsilon' \leq \varepsilon + \delta$
- Note: use potentially restricted distinguishers ***D***
  - ***D*** = combination of attacker ***A*** and challenger ***C***
  - ***D*** outputs 1 ⇔ ***A*** "won" (e.g., forged signature) against ***C***
- Best tradeoff between m, k & δ in a (k,δ)-extractor?

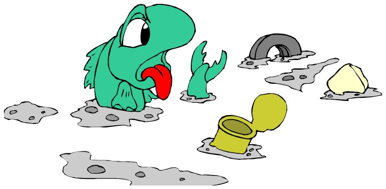
Leftover Hash Lemma

- □ LHL [HILL]: universal hash functions are (k, δ)-extractors where $\boxed{\delta = \sqrt{2^{m-k}}}$
- □ Corollary: For any P, $\boxed{\varepsilon' \leq \varepsilon + \sqrt{2^{m-k}}.}$ In particular,
  - $\boxed{k^* = m + 2\log(1/\varepsilon)}$ (entropy loss $2\log(1/\varepsilon)$ enough)
  - $\boxed{\varepsilon^* = 1}$ (no meaningful security when $k = m$ ☹)
- □ RT-bound [RT]: Any (k, δ)-extractor $\Rightarrow$ $\boxed{\delta \geq \sqrt{2^{m-k}}}$
  - Above bounds are optimal (in this level of generality) ☹

Theory vs. Practice:

Application P	KDF h	Sec. Loss $\varepsilon' - \varepsilon$	ε^* $(k=m)$	Entr. Loss $k^* - m$	Provable?
Computat. Secure	SHA/RO	$\varepsilon \cdot 2^{m-k}$	2ε	0	no
ANY	universal hash	$\sqrt{2^{m-k}}$	1	$2\log(1/\varepsilon)$	yes

How Bad is $2\log(1/\varepsilon)$ Entropy Loss?



- Many sources do not have "extra" $2\log(1/\varepsilon)$ bits
  - Biometrics, physical sources, DH keys on elliptic curves
    - DH: lower "start-up" min-entropy improves efficiency
  - AES-based P: $\varepsilon = 2^{-64}$, $m = 128 \Rightarrow k^* = 256 = 2m$ ☹
- Heuristic extractors have "no entropy loss": $k^* = m$
- End Result: practitioners prefer heuristic key derivation to provable key derivation [DGH+,Kra]
- Can we provably reduce it, despite RT-bound?

Options for Avoiding RT

- Route 1: restrict the power of distinguisher ***D*** or the class of (n, k)-sources X
  - Ex. 1: efficiently samplable sources X [DGKM12]
  - Ex. 2: computationally bounded ***D*** (pseudo-randomness)
  - Ex. 3: *implicitly* restrict ***D*** by considering special classes of applications P [BDK+11,DRV12,DY13,DPW13]
- Route 2: do we need to derive statist. random *R*?
  - Yes for OTP; No for many (most?) other applications P!

Options for Avoiding RT

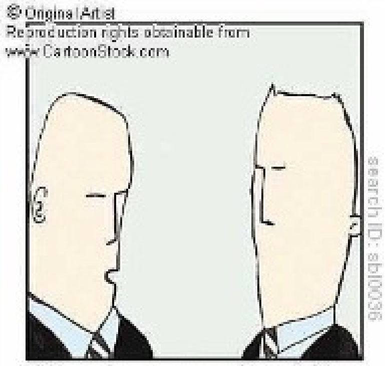
Punch line: Difference between Extraction and Key Derivation !

- Ex. 3: *implicitly* restrict ***D*** by considering special classes of applications P [BDK^{+}11,DRV12,DY13,DPW13]

□ Route 2: do we need to derive statist. random *R*?

- □ Yes for OTP; No for many (most?) other applications P!

Unpredictability Applications



"Massive unpredictablity is absolutely certain, maybe."

- Adv(***A***) = Pr[***A*** wins] = Pr[***D*** out. 1] $\in$ [0,1]
  - signatures, MACs, one-way functions, … (*not* encryption!)
- Case Study: key derivation for signature/MAC
  - Assume: Pr[A forges sig with uniform key] $\leq \varepsilon$ (= negl)
  - Hope: Pr[A forges sig with extracted key] $\leq \varepsilon'$ ($\approx \varepsilon$)
- Key Insight: only care about distinguishers ***D*** which almost never succeed on uniform keys (Pr[.] $\leq \varepsilon$) !
  - E.g., small multiplicative security loss is OK now

Unpredictability Extractors

- **UExt** is $(k, \varepsilon, \varepsilon')$-unpredictability extractor if

$$\Pr[\mathbf{D}(U_m, S) = 1] \le \varepsilon \Rightarrow \Pr[\mathbf{D}(\mathbf{UExt}(X;S), S) = 1] \le \varepsilon'$$

- **Theorem** [DPW13]: efficient $(k, \varepsilon, \varepsilon')$-**UExt** with
  - **Option 1:** $\varepsilon' = 3\varepsilon$ and $k = m + \log\log(1/\varepsilon) + 4$
  - **Option 2:** $\varepsilon' = \varepsilon \cdot (1 + \log(1/\varepsilon))$ and $k = m$

Plan of Attack

- **Step1.** Argue *any* unpredictability applic. P works well with (*only*) a high-entropy key R
  - Of independent interest !

E.g., random *R* except first bit 0 $\Rightarrow \varepsilon' \leq 2\varepsilon$

1: Security with Weak Keys

Entropy deficiency

- □ Fix P and any “legal” ***A***
- □ Let f(r) = [Advantage of ***A*** on key r] ∈ [0,1]
- □ Ideal Adv. ε = $\mathbb{E}[f(U_m)] = \sum_r \frac{1}{2^m}\cdot f(r)$
- □ Real Adv. ε’ = $\mathbb{E}[f(R)] = \sum_r p(r)\cdot f(r)$
- □ **Lemma**: If f(r) ≥ 0 and $\mathbb{H}_\infty(R) \ge m - d$,

  $$\boxed{\mathbb{E}[\ f(R)\ ] \le 2^d\cdot \mathbb{E}[\ f(U_m)\ ]}$$

  - Proof: $\sum p(r)\cdot f(r) \le 2^m\cdot \max_r(p(r))\cdot(\sum \frac{1}{2^m}\cdot f(r))$ ■
- □ **Corollary**: $\mathbb{H}_\infty(R) \ge m - d \Rightarrow \boxed{\varepsilon' \le 2^d\cdot\varepsilon}$

Plan of Attack

Achieve extremely low 2^d to compose with **Step1**!

Option 1: $2^d = 2$ and $k = m + \log\log(1/\varepsilon) + 4$

Option 2: $2^d = \log(1/\varepsilon)$ and $k = m$

- **Step2.** Build good condenser: relaxation of extractor producing high-entropy (but *non-uniform*!) derived key $R = h(X)$

2: Randomness Condensers

- (k,d,ε)-**condenser**: given $(n,\ k)$-source X, outputs m bits R "ε–close" to some $(m,\ m-d)$-source Y :

  $\boxed{(\mathbf{Cond}(X;\ S),\ S) \approx_{\varepsilon} (Y,\ S)}$ and $\boxed{\mathbb{H}_{\infty}(Y \mid S) \geq \boldsymbol{m - d}}$

  - **Cond** + Step1 ⇒ $\boxed{\varepsilon' \leq (1 + 2^{d}) \cdot \varepsilon}$

- Extractors: $d = 0$ but only for $k \geq m + 2\log(1/\varepsilon)$ ☹

- **Theorem** [DPW13]: efficient (k,d,ε)-condenser with
  - **Option 1:** $d = 1$ and $k = m + \log\log(1/\varepsilon) + 4$
  - **Option 2:** $d = \log\log(1/\varepsilon)$ and $k = m$

Balls and Bins

- Reduces to simple balls-and-bins game:
  - Throw 2^k balls into 2^m bins
  - Pick a random ball x
  - Lose if $|Bin(x)| > 2^d \cdot 2^{k-m}$
- **<u>Goal</u>**: given d, m, ε ⇒ min k s.t. Pr[Lose] ≤ ε
- Easy calculation ⇒ parameters of theorem if throw balls totally independently
- <u>Observation</u>: $\log(1/\varepsilon)$-independence suffices!

Balls and Bins

- Reduces to simple balls-and-bins game:
  - Throw 2^k balls into 2^m b
  - Pick a random ball x
  - Lose if $|Bin(x)| > 2^d \cdot 2^{k-}$
- **<u>Goal</u>**: given d, m, ε ⇒ min k Pr[Lose] ≤ ε
- Easy calculation ⇒ parameters of theorem if throw balls totally independently
- <u>Observation</u>: $\log(1/\varepsilon)$-independence suffices!

improve |S| to O(n log k) using “gradual increase of independence” [CRSW11]

Theory vs. Practice:

Application P	KDF h	Sec. Loss $\varepsilon' - \varepsilon$	ε^* $(k{=}m)$	Entr. Loss $k^* - m$	Provable?
Computat. Secure	SHA/RO	$\varepsilon \cdot 2^{m-k}$	2ε	0	no
Unpredict.	$\log(1/\varepsilon)$-wise hash	$\varepsilon \cdot \log(1/\varepsilon) \cdot 2^{m-k}$	$\varepsilon \cdot \log(1/\varepsilon)$	$\log\log(1/\varepsilon)$	yes
ANY	universal hash	$\sqrt{2^{m-k}}$	1	$2\log(1/\varepsilon)$	yes

Theory vs. Practice:

- Example: CBC-MAC, $\varepsilon = 2^{-64}$, $m = 128$

LHL: $\varepsilon^* = 1$ and $k^* = 256$

Now: $\varepsilon^* = 2^{-57.9}$ and $k^* = 138$

Heuristic: $\varepsilon^* = 2^{-63}$ and $k^* = 128$

Sometimes Dreams Come True!

Indistinguishability Apps?

□ Impossible for one-time pad ☹

□ Still, similar plan of attack:

- **Step1.** Identify *sub-class* of indist. applications P which work well with (*only*) a high-entropy key R
  - Will use Renyi entropy instead of min-entropy
  - Weaker inequality, but still beat LHL

- **Step2.** Build good *condensers* for Renyi entropy

Simple Inequality

- **Col**$(R) = \Pr[R_1{=}R_2] = \sum p(r)^2$
  - Renyi: $\mathbb{H}_2(R) = -\log$ **Col**$(R) \geq \mathbb{H}_\infty(R)$
- **Lemma**: For all f and $\mathbb{H}_2(R) \geq m{-}d$,

$$|\mathbb{E}[f(R)]-\mathbb{E}[f(U_m)]| \leq \sqrt{2^d-1}\cdot\sqrt{\mathbb{E}[f(U_m)^2]}$$

- Proof: LHS = $\left|\sum_r \left(p(r) - \frac{1}{2^m}\right)\cdot f(r)\right|$
- CS: $\leq \sqrt{2^m \sum \left(p(r) - \frac{1}{2^m}\right)^2}\cdot\sqrt{\frac{1}{2^m}\sum f(r)^2}$ …

■

Why is it Nice?

- **Lemma**: For all f and $\mathbb{H}_2(R) \geq \boldsymbol{m} - \boldsymbol{d}$,

$$\boxed{|\,\mathbb{E}[f(R)] - \mathbb{E}[f(U_m)]\,| \leq \sqrt{2^d - 1} \cdot \sqrt{\mathbb{E}[f(U_m)^2]}}$$

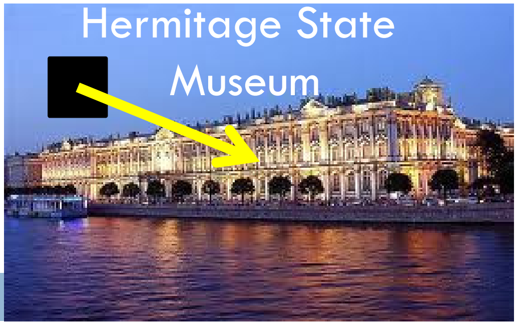
  - Works even if f(r) can be negative (indist. OK)
  - First term does not depend on f (i.e., appl. P)
  - Second term is for uniform distribution
  - Nicer entropy for condenser: $\mathbb{H}_2(R) \geq \mathbb{H}_\infty(R)$
- **Question**: $|\,\mathbb{E}[f(U_m)]\,| = \varepsilon$, what is $\mathbb{E}[f(U_m)^2]$?

Square Security

Malevich

- Def: P is σ-square secure (against a class of attackers A), if for any A ⇒ $\mathbb{E}[f_A(U_m)^2] \leq \sigma$
- **Lemma**: If P is ε-secure and σ-square secure, then P is ε'-secure in "(m-d)-real model", where $\boxed{\varepsilon' \leq \varepsilon + \sqrt{\sigma \cdot (2^d - 1)}}$
- Motivates studying square security!
- Question: how does square security σ relate to regular security ε?

Square-Friendly Applications



- P is square-friendly* (SQF) if $\sigma \leq \varepsilon$
- **<u>Example</u>**: all unpredictability applications P
  - $f \in [0,1] \Rightarrow \sigma = \mathbb{E}[f^2] \leq \mathbb{E}[f] = \varepsilon$ ▶
- Non-SQF applications: OTP, PRF, PRP, PRG ☹
- **[BDK⁺11,DY13]**: many natural indistinguishability applications are square-friendly !

▶ CPA/CCA-encryption, weak PRFs, q-wise independent hash functions, …

* Allow for small (say, factor of 2) degradation in the efficiency of the attacker A

Indistinguishability Apps?

ARE YOU CAPABLE OF DISTINGUISHING RIGHT FROM WRONG?
CAN YOU GIVE ME A *HINT?*
Bela

- Impossible for one-time pad ☹
- Still, similar plan of attack:
  - **Step1.** Identify *sub-class* of indist. applications P which work well with (*only*) a high-entropy key R
    - Will use Renyi entropy instead of min-entropy
    - Weaker inequality, but still beat LHL
  - **Step2.** Build good *condensers* for Renyi entropy

Universal Hash Functions

- Universal Hash Family $\mathcal{H} = \{\, h\colon \{0,1\}^n \to \{0,1\}^m \,\}$:

$$\forall\, x \neq x',\ \ \Pr_h[\, h(x) = h(x')\,] = \frac{1}{2^m}$$

- **LHL'**. Universal family $\mathcal{H}$ defines $(k,d,0)$-**condenser**$_2$ with m–bit output, where $\boxed{2^d - 1 \;=\; 2^{m-k}}$
  - $\Pr[h(X) = h(X')] \leq \Pr[X = X'] + \Pr[h(X) = h(X') \;\&\; X \neq X']$

$$= \boxed{2^{d-m} \quad \leq \quad 2^{-k} + 2^{-m}} \qquad \blacksquare$$

- **Corollary**: If P is ε-secure and square-friendly, then universal hashing yields KDF with $\boxed{\varepsilon' \leq \varepsilon + \sqrt{\varepsilon \cdot 2^{m-k}}}$

Theory vs. Practice:

Application P	KDF h	Sec. Loss $\varepsilon' - \varepsilon$	ε^* $(k=m)$	Entr. Loss $k^* - m$	Provable?
Computat. Secure	SHA/RO	$\varepsilon \cdot 2^{m-k}$	2ε	0	no
Unpredict.	$\log(1/\varepsilon)$-wise hash	$\varepsilon{\cdot}\log(1/\varepsilon){\cdot}2^{m-k}$	$\varepsilon{\cdot}\log(1/\varepsilon)$	$\log\log(1/\varepsilon)$	yes
Square-Friendly	universal hash	$\sqrt{\varepsilon \cdot 2^{m-k}}$	$\sqrt{\varepsilon}$	$\log(1/\varepsilon)$	yes
ANY	universal hash	$\sqrt{2^{m-k}}$	1	$2\log(1/\varepsilon)$	yes

Theory vs. Practice:

□ <u>Example</u>: CBC Encryption, $\varepsilon = 2^{-64}$, $m = 128$

LHL: $\varepsilon^* = 1$ and $k^* = 256$

LHL': $\varepsilon^* = 2^{-32}$ and $k^* = 192$

Heuristic: $\varepsilon^* = 2^{-63}$ and $k^* = 128$

Options for Avoiding RT

- Route 1: restrict the power of distinguisher ***D*** or the class of (n, k)-sources X
  - Ex. 1: efficiently samplable sources X [DGKM12]
  - Ex. 2: computationally bounded ***D*** (pseudo-randomness)
  - ✓ Ex. 3: *implicitly* restrict ***D*** by considering special classes of applications P [BDK+11,DRV12,DY13,DPW13]
- Route 2: do we need to derive statist. random *R*?
  - ✓ Yes for OTP; No for many (most?) other applications P!

Efficient Samplability

- **Theorem** [DPW13]: efficient samplability of X does *not* help to improve entropy loss below
  - $2\log(1/\varepsilon)$ for all applications P (RT-bound)
    - Affirmatively resolves “SRT-conjecture” from [DGKM12]
  - $\log(1/\varepsilon)$ for all square-friendly applications P
  - $\log\log(1/\varepsilon)$ for all unpredictability applications P
- **Idea**: bounded independent (n, k)-source X is enough to fool any extractor/condenser/…

Options for Avoiding RT

- Route 1: restrict the power of distinguisher ***D*** or the class of (n, k)-sources X
  - ✓ Ex. 1: efficiently samplable sources X [DGKM12]
  - Ex. 2: computationally bounded ***D*** (pseudo-randomness)
  - ✓ Ex. 3: *implicitly* restrict ***D*** by considering special classes of applications P [BDK$^+$11,DRV12,DY13,DPW13]
- Route 2: do we need to derive statist. random *R*?
  - ✓ Yes for OTP; No for many (most?) other applications P!

Minimal Assumptions

- □ **Theorem** [DGKM12 ,DPW13]: ~~SRT-conjecture ⇒~~ efficient **Ext** beating RT-bound for all *computationally bounded* ***D*** ⇒ **OWFs exist**
- □ How far can we go with OWFs/PRGs?
- □ Extract-then-Expand [Kra10]: Beats RT-bound, but only for medium-to-high values of k ☹
- □ Expand-then-Extract (aka “dense-model thm”): horrible run-time degradation in reduction ☹

Computational Extractor

- **Idea**: Design square-friendly key derivation
  - Good KDF for any computationally secure P
- **Solution**: Use weak PRF f: set $R = f_X(S)$
  - wPRF: secure for *random* (but public) inputs
- **Note**: f only needs security against 2 queries!
  - [DY13]: Can easily build using one PRG call: "expand-then-extract w/o time degradation"!
  - New alternative to "dense model" theorem ▶

Theory vs. Practice:

Application P	KDF h	Sec. Loss $\varepsilon' - \varepsilon$	ε^* $(k=m)$	Entr. Loss $k^* - m$	Provable?
Computat. Secure	SHA/RO	$\varepsilon \cdot 2^{m-k}$	2ε	0	no
Unpredict.	log(1/ε)-wise hash	$\varepsilon \cdot \log(1/\varepsilon) \cdot 2^{m-k}$	$\varepsilon \cdot \log(1/\varepsilon)$	$\log\log(1/\varepsilon)$	yes
Square-Friendly	universal hash	$\sqrt{\varepsilon \cdot 2^{m-k}}$	$\sqrt{\varepsilon}$	$\log(1/\varepsilon)$	yes
Computat. Secure	PRG + pairwise hash	$\sqrt{\varepsilon_{PRG} \cdot 2^{m-k}}$	$\varepsilon + \sqrt{\varepsilon_{PRG}}$	$\log(\varepsilon_{PRG}/\varepsilon^2)$	yes*
ANY	universal hash	$\sqrt{2^{m-k}}$	1	$2\log(1/\varepsilon)$	yes

* Under standard and minimal cryptographic assumptions (OWFs)

Summary

- Difference between extraction and KDF
  - $\log\log(1/\varepsilon)$ loss for all unpredictability apps
  - $\log(1/\varepsilon)$ loss for all square-friendly apps

    (+ motivation to study "square security")
- Efficient samplability does not help ☹
- Good computational KDFs require OWFs ☹
- **Main challenge**: better computational KDFs to close theory-vs-practice gap even further

Questions?

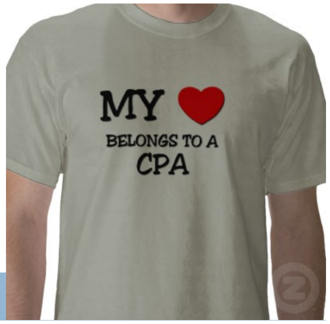
One-Time Pad

Plaintext
+ Keyword

Ciphertext

- Expect to fail even for min-entropy $m - 1$
  - $A(c) = c \Rightarrow f(0) = \frac{1}{2}, f(1) = -\frac{1}{2} \Rightarrow \varepsilon = 0, \sigma = \frac{1}{4}$
- Similar problem for PRGs/PRFs/PRPs ☹

CPA Security of Encryption



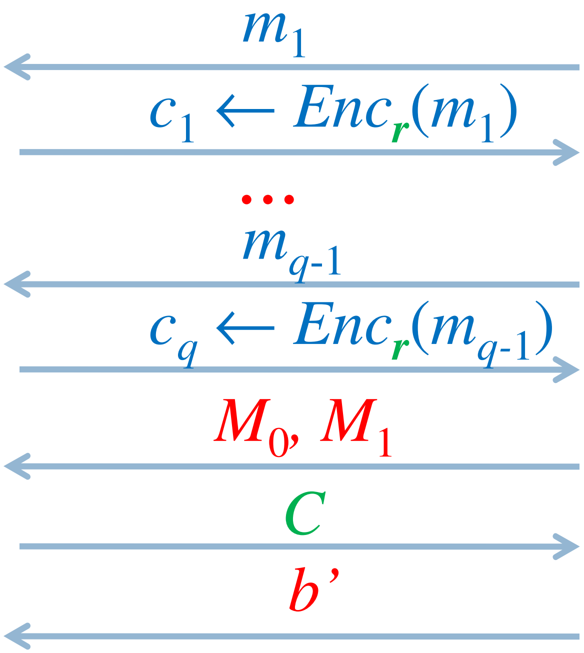
- *Probabilistic* Enc/Dec: $c \leftarrow Enc_r(m)\ ;\ m = Dec_r(c)$

Charlie

Alice

(run-time at most T)

$r \leftarrow U_m$


m_1
$c_1 \leftarrow Enc_r(m_1)$
…
$m_{q\text{-}1}$
$c_q \leftarrow Enc_r(m_{q\text{-}1})$
M_0, M_1
C
b'

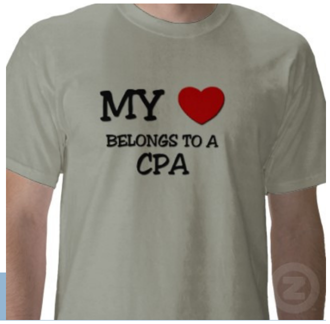
Pick $b \leftarrow \{0,1\}$

Set $C \leftarrow Enc_r(M_b)$

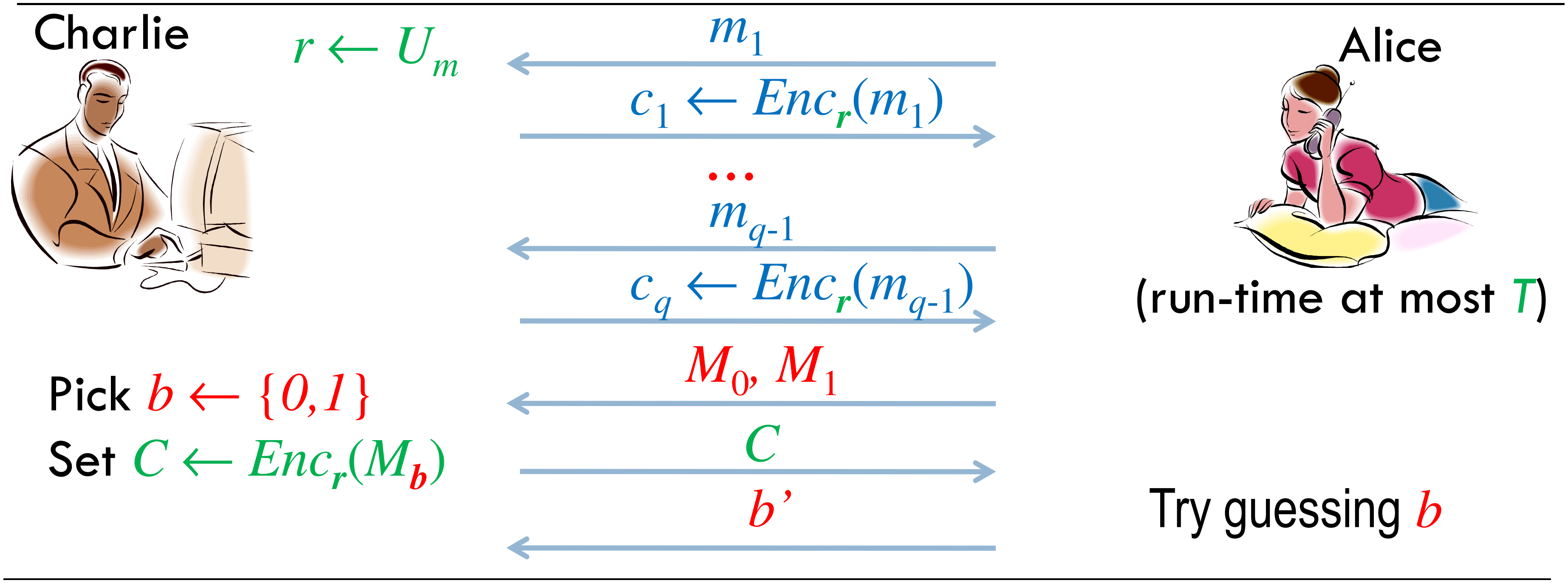
Try guessing b

- Define $f(r) = Adv(\ A, r\ ) = \Pr[b = b'] - \frac{1}{2} \in [-\frac{1}{2}, \frac{1}{2}]$
  - Leads to (T, q, ε)-security/(T, q, σ)-square security

CPA Security of Encryption



- *Probabilistic* Enc/Dec: $c \leftarrow Enc_r(m)$; $m = Dec_r(c)$



Charlie

$r \leftarrow U_m$

m_1

$c_1 \leftarrow Enc_r(m_1)$

…

$m_{q\text{-}1}$

$c_q \leftarrow Enc_r(m_{q\text{-}1})$

Pick $b \leftarrow \{0,1\}$

Set $C \leftarrow Enc_r(M_b)$

M_0, M_1

C

b'

Alice

(run-time at most T)

Try guessing b

- **Lemma**: if Enc is $(2T, 2q, 2\varepsilon)$-secure, then

  Enc is (T, q, ε)-square secure ("$\sigma \approx \varepsilon$")

Square Security of CPA

- **Insight**: for any A making q encryption queries, there exists B making 2q encryption queried s.t. $\forall \boldsymbol{r}$ $\quad Adv(\, B, \boldsymbol{r}\,) = 2Adv(\, A, \boldsymbol{r}\,)^2 \geq 0$ (**)
- Here's B:

1. Run A once against *simulated* challenger C
   - Choose selection bit yourself ⇒ can check if A "won"
   - Spend q queries to simulate *both* A and C
2. Run A again against *real* challenger C (+ q queries)
3. If A lost in Step 1., reverse A's guess in Step 2.
   - Intuition: Step 3. ensures B has advantage ≥ 0

Square Security of CPA

- **Insight**: for any *A* making *q* encryption queries, there exists *B* making 2*q* encryption queried s.t. $\forall \boldsymbol{r}$ $\quad Adv(B, \boldsymbol{r}) = 2Adv(A, \boldsymbol{r})^2 \geq 0 \quad (**)$
- Here's *B*:
  1. Run *A* once against *simulated* challenger C
  2. Run *A* again against *real* challenger C
  3. If *A* lost in Step 1., reverse *A*'s guess in Step 2.
- $\Pr[B \text{ wins}] = \Pr[A \text{ wins twice}] + \Pr[A \text{ looses twice}]$

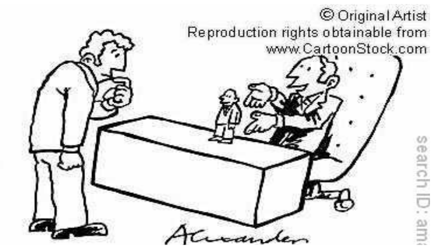
$$= \left(\frac{1}{2} \pm \varepsilon\right)^2 + \left(\frac{1}{2} \mp \varepsilon\right)^2 = \frac{1}{2} + 2\varepsilon^2$$

■

Square Security of CPA

- **Insight**: for any A making q encryption queries, there exists B making $2q$ encryption queried s.t. $\forall \boldsymbol{r}$ $\quad Adv(\,B, \boldsymbol{r}\,) = 2Adv(\,A, \boldsymbol{r}\,)^2 \geq 0 \quad (**)$
  - Hence, $\sigma = \mathbb{E}[Adv(\,A, \boldsymbol{r}\,)^2] \leq ½\, \mathbb{E}[Adv(\,B, \boldsymbol{r}\,)] \leq \varepsilon$ ■
- **Corollary**: if Enc is $(2T, 2q, 2\varepsilon)$-secure, then Enc is $(T, q, \sqrt{\varepsilon \cdot 2^d})$-secure in the $(m\text{-}d)$-real model
  - [BG09]: $((1+c^4)T, (1+c^4)q, \varepsilon) \Rightarrow (T, q, O(\frac{1}{c} \cdot \sqrt{\varepsilon \cdot 2^d}))$
- Same argument works for *weak* PRFs, *greatly* simplifying [Pie09]

New Dense Model Theorem



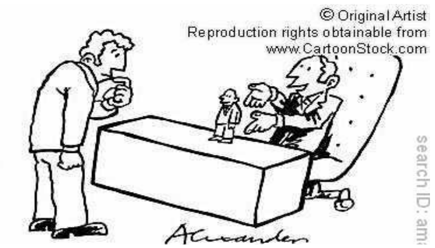
I had my people and your people crushed together to create this one superdense person

- How to build PRG with weak seed?
  - Naïve: $G(X)$ not pseudorandom, even if $\mathbb{H}_2(X) = m - 1$
- **<u>Dense Model Theorem</u>**: if $\mathbb{H}_\infty(X) \geq m - d$, then $G(X)$ has "pseudo-entropy" $2m{-}d \gg m$
  - Implies $G(\mathrm{Ext}(G(X); S))$ is psedorandom given S
  - <u>Problem</u>: bad degradation in run-time t

Pairwise independent hash

- **<u>Our Version</u>**: if $\mathbb{H}_2(X) \geq m - d$, then $G\left(\underbrace{\mathrm{PIH}_{G(X)}(S)}_{\text{wprf}}\right)$ is psedorandom given S
  - No degradation in t, security $\sqrt{\varepsilon \cdot 2^d}$ (vs. $\varepsilon \cdot 2^d$)

New Dense Model Theorem



I had my people and your people crushed together to create this

Leads to **same** concrete instantiation:
pairwise independent hash *is* a good extractor!

Open: a **single** unified proof, giving a smooth transition between these two "extreme" bounds

- Implies $G(\mathrm{Ext}(G(X);S))$ is psedorandom given S
- Problem: bad degradation in run-time t

Pairwise independent hash

- **Our Version**: if $\mathbb{H}_2(X) \geq \boldsymbol{m} - \boldsymbol{d}$, then $G\left(\mathrm{PIH}_{G(X)}(S)\right)$ is psedorandom given S
  - No degradation in t, security $\sqrt{\varepsilon \cdot 2^d}$ (vs. $\varepsilon \cdot 2^d$)