Cryptographic Extraction

Hugo Krawczyk
IBM Research

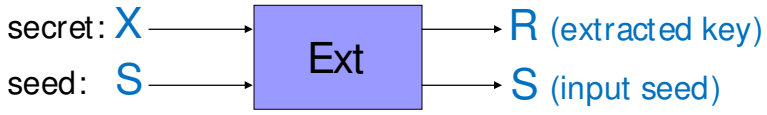
"Is Cryptographic Theory Practically Relevant?"
Cambridge 2-1-12

Imperfect Random Sources

- Uniform randomness is crucial in many areas
  - Cryptography: requires source of secret randomness (secret keys)
- However, often deal with imperfect randomness
  - non-uniform, partial knowledge (i.e., partial secrecy)
  - physical sources, system RNGs, biometric data, extracting from group elements (DH key exchange),…
- Can we convert imperfect sources into (nearly) perfect randomness?

Randomness Extractors [NZ96]

- Input: a *weak secret* X and a *uniformly random seed* S
- Out: *extracted key* R = **Ext**(X; S)



- R is uniformly random, even conditioned on the seed S

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

sufficient min-entropy in X

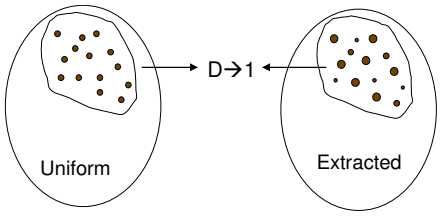
- **Many uses in complexity theory and cryptography**
  - Here: focus on ***key derivation*** but many more applications

3

Parameters

- Min-entropy m ("*m-source*"): $\Pr[X=x] \le 2^{-m}$, for all x
- Output length n (entropy loss $L = m - n$)
- Error ε (measures *statistical* distance from uniform)
  - Equals $\max_D |\Pr[D(\text{Ext}(X; S), S)=1] - \Pr[D(U_n, S)=1]|$

Optimal $\varepsilon = 2^{-L/2}$



D can be unbounded

(later, computationally b'ded distinguishers)

4

Historical Anecdotes (ca.1996)

- Crypto design of IPsec and IKE (IPsec's Key Exchange)
- Authenticated Diffie-Hellman protocols
- Parties agree on g^{xy}: But how to derive session keys?
- The "obvious way": $H(g^{xy} || info)$
- Objection: Let's not treat H as ideally random (e.g., extension attacks)
- Also: bad interaction with other uses of H
  - The Photuris Example

5

Photuris Example

- Photuris (a predecessor of IKE) transmitted $SIG(g^{xy})$ to authenticate g^{xy}. SIG implemented with RSA.
- Then derived key as $K=H(g^{xy} || info)$
- Attack: from $(SIG(g^{xy}))^e$ recover $H(g^{xy})$ and since $|g^{xy}|=1024$ is multiple of 512 then can compute K!
  - $K = H(g^{xy} || info) = h(IV=H(g^{xy}), info)$

6

Using g^{xy} as PRF Key

- K = PRF(key=g^{xy}, info): looks "sounder" than using ideal H
- But what's the PRF security with a semi-random key?
- Also: Does the PRF accept a key of size $|g^{xy}|$?
  - Not if PRF is implemented via AES/DES in CBC mode!
  - OK for HMAC: If $|g^{xy}| > 512$ HMAC replaces key with H(g^{xy})

☹ The HMAC use is an "illusion", the actual KDF is H(g^{xy})

- "Beautifully" illustrated in Photuris again
- K = HMAC(key=g^{xy}, info) = HMAC(key=H(g^{xy}), info)
- But H(g^{xy}) is known!

7

Theory to the Rescue: Extractor

- "Source of Key Material" → almost uniform (short) key
- In IKE case "Source of Key Material" is g^{xy}
- But extractor needs
  - Source with enough min-entropy (recall "entropy loss")
  - A random (but non-secret) "seed"

☺ Under DDH assumption g^{xy} has *computational* entropy

→ g^{xy} is *computationally indisting.* from a high entropy source

☺ Random seed: nonces exchanged during key exchange

8

Extract-then-Expand

- Once a first key K is extracted, use a PRF to *expand* it into more keys (encryption, authentication, etc.)
  - K1, K2, K3 = PRF_K(info, 1) || PRF_K(info, 2) || ….
- **Extract-then-Expand KDF**
  - K_{prf} = Extract(salt, skm) skm= source key material
  - Keys = Expand(K_{prf} , Keys_length, ctxt_info)
- Works for any source (e.g., sampled weak randomness) hence suitable for arbitrary KDF applications
  - DH, PRNGs (s/w, h/w), biometric data, etc.

9

How to Implement the Extractor?

- Efficient implementation: Any Strong Universal Hash F.
  - Leftover Hash Lemma: A pairwise independent hash function is a statistical extractor: Optimal stat. distance $\varepsilon = 2^{-L/2}$, L=m-n
- But inappropriate for IKE
  - Requires high min-entropy: e.g. for $\varepsilon=2^{-80}$ n=128 need m > 288 (e.g. couldn't use Schnorr 160-bit or even 256-bit subgroups)
  - Requires long seeds (salt): ~$|g^{xy}|$
  - Not in crypto libraries and hardware (*)

(*) Also: It is not easy to sell "fancy" crypto today – it was much harder 15 years ago

10

IKE "extractor"

- Extract(salt, g^{xy}) = PRF(key=salt, g^{xy} || info)
- Good: Re-used existing PRF ! (for extract and expand)
- *But a PRF with non-secret key?*
- The hope (my hope) was that for specific PRFs such as HMAC and CBC-MAC the above was a good extractor
  - The inspiration came from theory but took many years to make formal sense of this
    - How much entropy g^{xy} has [GKR03]
    - Analysis of HMAC and CBC as extractors [DGHKR04]
    - Formalization of KDF and the Extract-then-Expand approach [K'10]

11

HKDF

- The HMAC-based implementation now defined as a multi-purpose KDF (called HKDF)
  - RFC 5869, NIST SP 800-56C
- Multi-purpose: *any* source with enough entropy but also:
  - Applications that require a Random Oracle (e.g. some implicit auth'd KE)
  - Applications that extract keys from "computational hardness" without any min-entropy (e.g. CDH vs DDH)
  - Applications that don't have a randomness source for extractor's seed
- Note: We are talking about *computational* extractors not *statistical* extractors (output is comp. indistinguish.)

12

Computational Extractors

- In KDF applications (and most other applications in crypto) we don't need statistical closeness to uniform
- Computational indistinguishability suffices
  - In partic., statistical extraction impossible when source entropy is computational or when using crypt'c hash functions or PRFs
- A *computational extractor* is a randomness extractor that resists computationally bounded distinguishers (i.e, its output is pseudorandom)
- NMAC/HMAC have good computational extraction properties (see next for some examples)

13

HMAC as Extractor (sample results)

- If $\{h_K\}$ is an extractor family and $\{h_K'\}$ a PRF then NMAC is a good extractor on source with blockwise entropy k (e.g., m=160, block=512)
  - Applies, for example, to IKE with safe-prime groups
- If $\{h_k\}$ is universal and $\{H_k\}$ is *mildly* collision-resistant then NMAC truncated by c bits is $(n2^{-c/2})$-stat close to uniform (n=# of blocks)
  - Application: use SHA-512 for extraction, SHA-256 for PRF then c=256 and one gets security of 128 bits w/mild assumptions
  - Note: Computational assumption but extraction is statistical ("mild collision resistance": against generic linear-size circuits)
- Even stronger properties if h (not H) modeled as RO: e.g. hardcore
  - $q2^{-m}$-indistinguishable from U_n with q queries

14

Revisiting Statistical Extractors

- Efficient but some significant practical limitations
  1. Require large gap between entropy and output size
     entropy > |output|+ 2*security (128-bit key + 128 security → ent ≥ 384)
  2. Large number of salt bits (few hundreds and beyond)
  3. Unsuited in general as deterministic extractors and as general hard-core functions (extracting from "unpredictability")
  4. Code (un)availability in crypto implementations and libraries
- In some specific applications of extractors 3 and 4 may not be an issue (enough salt, willing to implement a stat extrac}
- 1-2 are *inherent* to statistical extraction (lower bounds)

15

Inherent

1. RT bound: $\varepsilon \geq 2^{-L/2}$ where L=n-m (i.e. $m \geq n+2\log(1/\varepsilon)$)
   - Require large gap between entropy and output size:
     entropy > |output|+ 2*security (128-bit key + 128 security → ent ≥ 384)
2. $|seed| \geq \min(|X| - n, \log |X| + 2\log(1/\varepsilon) - O(1))$
   - E.g., |X|=1024 ; $\varepsilon = 2^{-128}$; practical construction: |seed| ~1000

- **Inherent?** Can't we do anything about it?
  - Had you asked me this 2 years ago I'd have said: No. Sorry.

16

Ask me again…

- Can we do anything about these *inherent* limitations?
- Yes, we can!

- But wait, are you disproving well-established lower bounds in complexity theory?
- No. Just asking a more specific question. One that is relevant to our cryptographic applications.

17

Leftover Hash Lemma (LHL)

- Universal Hash Function: Family $H=\{h_k\}$, $h_k:\{0,1\}^N \to \{0,1\}^n$ such that for all $x \neq y$ $\text{Prob}_k(h_k(x)=h_k(y)) \leq 2^{-n}$
- LHL: For any source X over $\{0,1\}^N$ of min-entropy m: $(h_k(X), k) \approx_\varepsilon (U_n, k)$ with $\varepsilon = 2^{-L/2}$ where L=m-n
- Application to key derivation:
  - 160-bit EC group → 128-bit AES key: L=160–128=32 ➔ $\varepsilon = 2^{-16}$
  - 256-bit EC group → 160-bit SHA key: L=96 ➔ $\varepsilon = 2^{-48}$
- Security degradation due to "entropy gap" is serious: e.g., to get security 2^{-80} for SHA1 we'd need EC-320

18

Can we do better?

- Not in general: statistical distance $2^{-L/2}$ is optimal [RT]
- It means that the best guarantee we have is:
  - If an algorithm succeeds with prob ε on uniform inputs, it succeeds with prob $\varepsilon+2^{-L/2}$ on inputs generated by the extractor
    - $\forall A$: $\text{Prob}(A(z)=1: z\sim h_k(X)) \le \text{Prob}(A(z)=1: z\sim U_n) + 2^{-L/2}$
  - In crypto applications this means a $2^{-L/2}$ security degradation
- But here is a trick: If we only care about *rare events* (e.g., inputs where a crypto inverter succeeds w/high prob) then we can improve significantly on the LHL bound

19

Generalized LHL (GLHL)

- Assume A outputs 1 with prob ε on uniform n-bit inputs
- Run A on inputs produced by an LHL-extractor on a source of min-entropy m.
- Then the probability that A outputs 1 is at most

$$\varepsilon + \varepsilon^{1/2} 2^{-L/2}$$

- In crypto applications this means security degradation $\varepsilon^{1/2}\, 2^{-L/2}$ rather than $2^{-L/2}$ (borrowing from primitive security)
- "Leftover Hash Lemma, Revisited": B Barak, Y Dodis, H Krawczyk, O Pereira, K Pietrzak, F Standaert, Yu Yu – CRYPTO'11

20

Examples (key search and MAC)

- GLHL: *If key is chosen via LHL-extractor on m-source, wining probability is at most* $\varepsilon + \varepsilon^{1/2} 2^{-L/2}$
- For small ε this is a significant improvement!
  - EC-160 → 128-bit AES key, $\varepsilon=2^{-96}$: LHL: 2^{-16} to GLHL: 2^{-64}
  - EC-256 → 160 HMAC-SHA1 key, $\varepsilon=2^{-80}$: LHL: 2^{-48} to GLHL: 2^{-88}
- It gets even better: We can even GAIN entropy (L<0)
  - EC-160 → 256-bit HMAC-SHA2 key, Assume $\varepsilon=2^{-200}$

    Then, security with extracted keys is 2^{-52} $(=2^{-200/2} \cdot 2^{96/2})$

21

How about indistinguishability

- Above examples use the fact that attacker's success probability ε is tiny
- But how about indistinguishability applications (e.g. sem. security)? In this case the success prob is ~ ½.
- Can we still say something? E.g., what's the security of an encryption scheme that uses an extracted key?
- A (t,ε)-secure PK ENC (i.e., attacker's success prob $½+\varepsilon$) is $(t/2, \varepsilon + \varepsilon^{1/2} 2^{-L/2})$-secure when using extracted keys
  - same for symmetric-key ENC if resistant to q>1 ENC queries

22

Bad news for PRFs

- ☹ When extracting one-time pads, PRG seeds, or PRF keys the GLHL does not improve over LHL.
- ☹ Bad for a generic KDF application where what we need is to derive a PRF key!
- ☺ But not all is lost: GLHL does improve for wPRF keys
  - Weak-PRF: A PRF that is secure when applied to random inputs
- This suffices to build a KDF with the improved GLHL security!

23

KDF From Statistical Extractors and Small Entropy Loss

- Generic KDF scheme: Univ. hash family $\{UH_h\}$; a weak PRF wPRF
  - Salt (random but public): A hash key h and a random string s
  - Compute $K=UH_h(\text{source})$, set $\text{DerivedKey}=wPRF_K(s)$
- Gets all the benefits of GLHL: reduces entropy requir't and decreases LHL security degradation.
- Yet, it still requires long salt: especially long UH key
- Good news: Can use short salt and amplify with PRG
  - Under suitable assumptions this works *even though PRG seed is public!!* **SEE CRYPTO'11 PAPER...**

24

New Work on Comp. Extractors

- "Computational Extraction and Pseudorandomness", TCC'12. D. Dachman Soled, R. Gennaro, H. Krawczyk, T. Malkin.
- New construction of KDF without using statistical extractors based on *any* exponentially hard wPRF for which |skm| ≤ |wPRF key|
  - E.g. PRF is HMAC-SHA-256, skm is a ECC-160 group element
  - Provides significant security even though we output more bits than the input entropy!
- Some interesting theoretical results on the question: Do non-trivial computational extractors require OWFs?

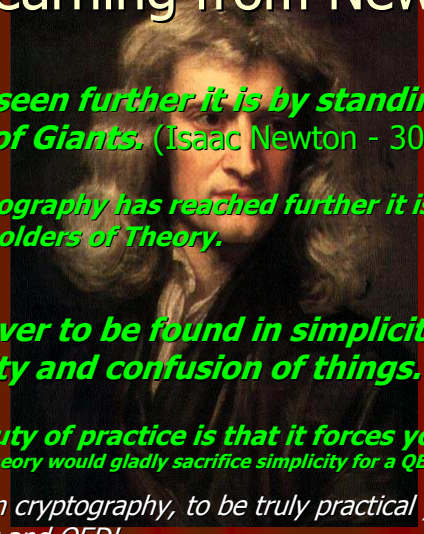
25

More on the TCC'12 paper

- Computational extractors exist unconditionally: any statistical extractor is also computational.
- Define "proper computational extractors" as those extractors whose output is statist. far from uniform
- We can build proper extractors using PRGs: Use a stat. extractor to derive a seed; expand the seed using PRG.
- Do proper computational extractors require OWFs?
- Want to know the answer? Read the paper…

26

Learning from Newton



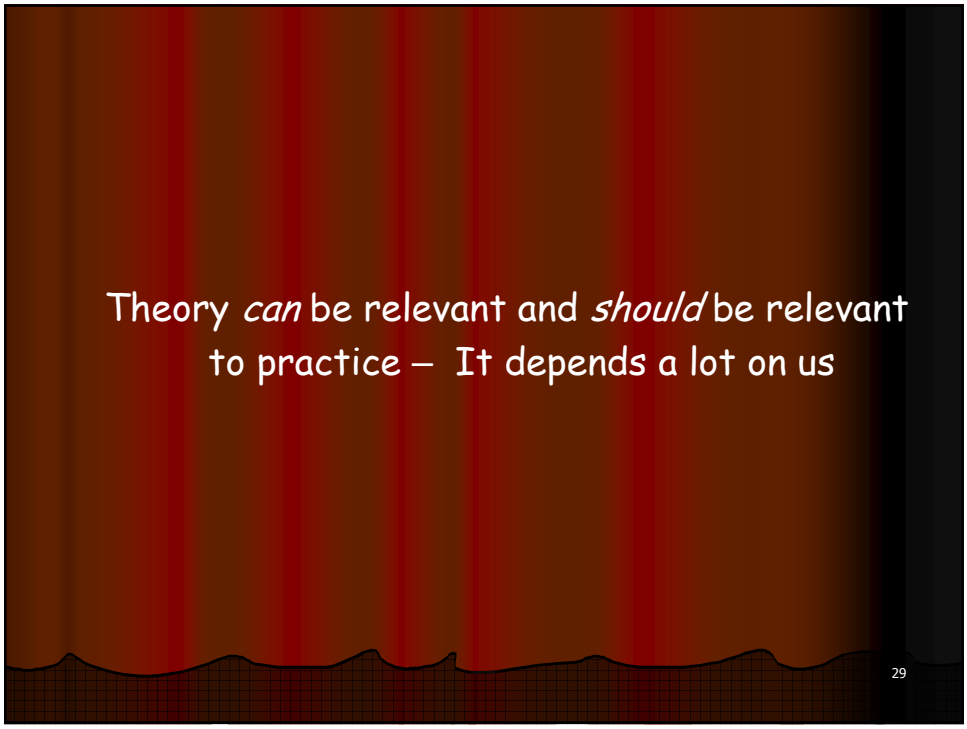
- ***If I have seen further it is by standing on ye sholders of Giants.*** (Isaac Newton - 300 B.DH.)
  - ➔ ***If Cryptography has reached further it is by standing on ye sholders of Theory.***
- ***Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things.*** *(Isaac Newton)*
  - ➔ ***The beauty of practice is that it forces you to do it SIMPLE*** ***(whereas theory would gladly sacrifice simplicity for a QED)***
  - * *Actually, in cryptography, to be truly practical you need BOTH: Simplicity and QED!*

27

Q.E.D.

- It is not about theory vs. practice but the dynamic interaction between the two (a *"balancing act"*)
- Theory is not a panacea; it is model-based and model-limited (as any applied math discipline)
- But the ONLY way to reason about "for all adversaries": no experimentation or simulations can establish security
  - Theoretical concepts and methods prove amazingly well suited to practical design and analysis (I call it *"proof-driven design"*)
- Need more involvement of researchers with standards ("do not complain later" – it is tough but rewarding)

28


Theory *can* be relevant and *should* be relevant to practice – It depends a lot on us
29