Truncation in Cryptographic Hash Functions

Zulfikar Ramzan

DoCoMo USA Labs

Hash Truncation Dilemma

- Some standards organizations are urgently trying to find a suitable replacement for SHA-1 because the confidence in it has been shaken due to the recent collision results.
- One suggestion that has been made in standards proposals is to truncate the output of other hash functions:
  - e.g., Truncate-160(SHA-256(M)).
- However, this might be a bad idea if the first 160 bits of SHA-256 have any weaknesses or bias.
  - Of course, a hash function that is collision resistant should "in theory" not have weak bits.
  - In particular, weak bits imply a bias in the output distribution which would in-turn reduce the complexity of a birthday attack.
- One possible avenue for a solution: randomness extractors…

Randomness Extractors

- Complexity-theorists have done much work on **randomness extractors** which can "extract" a **short uniformly random string** from a longer string that has **high-min entropy** (but which is not uniformly random).
- Such extractors can be **implemented efficiently**
  - Two-universal hash functions are provably good extractors [IZ89].
  - Definition: For all x != y, $Pr_k[H_k(x) = H_k(y)] < \varepsilon$.
  - Example: over a finite field like GF(2n) : $H_k(x) = kx$
- Potential approach: $(Extractor_k(SHA\text{-}256(m)), k)$
- Issue: randomness extractors require some random seed to start with, but it can be public (no secret key required)!
  - Might be OK for randomized hash functions [HK05].
  - Perhaps there is a good possible source of randomness elsewhere.
  - Randomness can be specific to a protocol implementation.
  - Perhaps can use cryptographic hash function to generate randomness (though you may sacrifice provable security: chicken and egg problem).