Randomized Hashing for Signatures

Shai Halevi and Hugo Krawczyk

IBM Research

http://www.ietf.org/internet-drafts/draft-irtf-cfrg-rhash-00.txt

Our Proposal: Executive Summary

- Hash functions should have a randomized "mode of operation"
  - This mode needs weaker security properties from the underlying hash function.
- Signature standards should use this mode
  - So that signatures will remain secure ***even if*** the hash function in use only has the weaker security.

Note:

- This is a general and well-known methodology that's advisable regardless of the specific hash function in use.
- This methodology is especially advisable today, when we're not sure about the security of the hash functions we're using.

Hash Functions and Signatures

- To sign a message x:
  - Set h = H(x)
  - Compute, e.g., s = RSA^{-1}(encode(h))
  - s is the signature on x
- If an attacker can find y≠x s.t. H(x)=H(y) then s is also a signature on y
- ...you were using MD5 for THAT???

How to fix this?

- Use more secure hash functions
  - Do you know of any?
- Use schemes that require less security of the underlying hash function
  - That's our focus in this I-D
  - In particular, using randomized/salted hashing

Salted Hashing and Signatures

- Use $H_r(x)$ instead of $H(x)$
  - r is a random "salt value"
  - Later we'll talk about how to salt H
- To sign a message x:
  - Choose a new random salt r, set $h = H_r(x)$
  - Compute, e.g., $s = RSA^{-1}(encode(h,r))$
  - The signature is the pair (r,s)

Why is this better?

- Finding plain ("off-line") collisions in H are useless to attacker.
- To attack the signatures via finding collisions in H, an attacker needs to:
  - Obtain signatures (r_i,s_i) on messages x_i
  - For some i, find some $y \neq x_i$ s.t. $H_{ri}(x_i)=H_{ri}(y)$.
- This seems considerably harder than finding collisions off-line.

Standard levels of security of hash Functions

- Strong: full collision resistance
- Weaker: target collision resistance
  - We'll mostly focus on this
- There are even weaker notions
  - 2^{nd} pre-image resistance
  - One wayness

Full Collision Resistance (CR)

- Attacker cannot find any $x \neq y$ s.t. $H(x)=H(y)$
- That's a very strong requirement
  - We should design hash functions to meet this level of security
  - But also design signature schemes that do not depend on the hash functions meeting such a strong notion of security

Target Collision-Resistance (TCR)

- Security against the following attack:
  - Attacker chooses x
  - r is chosen at random and given to attacker
  - Attacker tries to find $y \neq x$ s.t. $H_r(x)=H_r(y)$
- Sounds familiar?
  - Theorem: Using TCR hashing in the mode from four slides ago is sufficient for secure signatures
  - See [Bellare-Rogaway97,Naor-Yung89]

TCR is weaker than CR

- No “birthday paradox”, brute-force attack takes 2^n time rather than $2^{n/2}$
- The attacker needs to interact with the “hasher”, not an off-line attack

Modifying signature standards to use randomized hashing

- The main issue is likely to be where to fit the salt component r in existing signature fields
  - Maybe as part of an `AlgorithmIdentifier`? (suggestion due to Burt Kalisky)
- In most settings, generating the randomness is unlikely to be an issue

RSA Signatures

- It may be possible to use the “message recovery” property of RSA
  - r can be deduced from encode(h,r)
  - So the signature is only s = RSA^{-1}(encode(h,r))
  - To verify you must first compute RSA(s), then recover r and hash
- More discussion in the draft

DSA Signatures

- DSA signatures already have a format (r,s) with a random r
- Hopefully we can use the same r also for hashing
- More discussion in the draft

How to Salt a Hash Function?

- More Research is Needed on That
- Some plausible proposals:
  - $H_r(x) = H(r \oplus x)$
    - if r is shorter than x, just repeat it
  - Or also interleave r after every block of x
  - See discussion in the Internet-Draft
- Aside: $H_r(x) = \text{HMAC-}H_r(x)$ does not seem to be the right answer

Repeating Executive Summary

- Hash functions should have a randomized "mode of operation"
  - This mode makes weaker security requirements from the hash function in use
- Signature standards should use this mode
  - So that these weaker security requirements will suffice for secure signatures

Two more comments

On “provable security”:

- “Provable Security” of signatures is often in the Random-Oracle model
- It seems a stretch to use this model when talking about “broken hash functions”
- Not clear what model is reasonable for proving security in this context

On “on-line” vs. “off-line” attacks:

On-line vs. Off-line: Scenario #1

Engineer: "We're using MD5 for certificates, LWW can forge a certificate with about 2^{35} off-line computations (takes maybe a few hours on a PC)."

Boss: "I want this fixed yesterday, cancel all vacations until it is fixed!"

(… and later I'll fire you for letting this happen)

LWW: Lenstra, Wang and Weger

On-line vs. Off-line: Scenario #2

Engineer: "We're using randomized-MD5 for certificates, LWW can forge a new certificate after we give them about 2^{35} valid certificates (2^{35} ~ 30 billion)."

Boss: "I'm going on vacation now, we'll discuss this when I'm back."

(… hopefully by then somebody else will fix it)

Is TCR Really the Right Notion?

- Actually, an attacker can also:
  - Request signatures on many messages $x_1 \ldots x_n$
  - Get $(r_1,s_1) \ldots (r_n,s_n)$
  - Tries to find $y \neq x_i$ s.t. $H_{r_i}(x_i)=H_{r_i}(y)$ (for some i)
- Note: this is an on-line attack (vs. off-line attacks if the hashing is deterministic)