Hash Functions With Proofs of Security

Lars R. Knudsen

April 21, 2008

Generic attacks

For $H : \{0,1\}^* \rightarrow \{0,1\}^n$ and $h : \{0,1\}^m \rightarrow \{0,1\}^n$, $m > n$

attack	rough complexity
collisions	$\sqrt{2^n} = 2^{n/2}$
2nd preimages	2^n
preimage	2^n

Goal: generic attacks are best (known) attacks

Number-theoretic, difficult problems

- Factoring:

  given $N = pq$, find p and q,

  where p, q big, (odd) prime numbers, $p \neq q$

- Recommended that $N \geq 2^{1024}$ for high level of security

- A 1024-bit N:
  135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

Number-theoretic, difficult problems (2)

- Discrete logarithm:

$$\text{given } \beta = \alpha^a \bmod p, \text{ find } a,$$

  where p prime, a chosen random from Z_{p-1}, $\alpha \in Z_p^*$ primitive
- Recommended that $p > 2^{1024}$ for high level of security
- But not all instances of these problems are hard, e.g.,
  - if $p \simeq q$, then factoring $N = pq$ is easy
  - if $p-1$ has only small prime factores then finding discrete logs modulo p is easy

Hash based on factoring (Shamir)

- $N = pq$, $p \neq q$, large odd primes, α fixed, large order mod N.
- Public: N, α

$$H : \{0,1\}^* \to Z_N^*$$

$$H(x) = \alpha^x \bmod N$$

- Collision: $H(x) = H(x') \;\Rightarrow\; x - x' = k\phi(N)$.
- With $N = pq$ and $\phi(N) = (p-1)(q-1)$ easy to find p and q

Hash based on discrete log (Pfitzmann, Van Heijst)

- Public primes: $p, q = \frac{p-1}{2}$, s.t. DLP(p) is hard
- Public primitive elements of Z_p: α, β (randomly chosen)

$$h : Z_q \times Z_q \to Z_p^*$$

$$h(x, y) = \alpha^x \beta^y \bmod p$$

- Find a collision for $h \Rightarrow$ compute $\log_\alpha(\beta)$

Number-theoretic hash functions

- most schemes slow, e.g., no real speed-up for use in digital signature schemes
- some schemes have unfortunate algebraic properties (may interact badly with other public-key algorithms)
- open problem to devise efficient "provably" secure hash function

Newer constructions

- VSH - Very Smooth Hash
  - Contini, Lenstra, Steinfeld, 2005
  - collision $\Rightarrow$ nontrivial modular square roots of very smooth numbers modulo N (composite)
  - efficient collision finder implies fast factoring algorithm
- LASH - A Lattice Based Hash Function
  - Bentahar, Page, Saarinen, Silverman, Smart 2006
  - based on the problem of finding small vectors in lattices

Factoring - Equal squares

- Let $n = pq$, $p \neq q$, odd primes. Let $x \in Z_n^*$
- 4 square roots of x^2 mod n are $x, -x, y, -y$ mod n where $x \neq \pm y$ mod n

mod n	mod p	mod q
x	z	w
$-x$	$-z$	$-w$
y	z	$-w$
$-y$	$-z$	w

- $\gcd(x+y, n) = q$, $\gcd(x-y, n) = p$,
- Find (random) a, b s.t. $a^2 = b^2$ mod n, factor n with prob. $\frac{1}{2}$

Example. Dixon's algorithm with $n = 4189$

- $\sqrt{4189} \simeq 64.7$
- Use the factor base $\mathcal{B} = \{-1, 2, 3, 5, 7, 11, 13\}$

x_j^2 mod n	factorisation of x_j^2	a_j
58^2 mod n	$-1 \cdot 3 \cdot 5^2 \cdot 11$	$(1,0,1,0,0,1,0)$
61^2 mod n	$-1 \cdot 2^2 \cdot 3^2 \cdot 13$	$(1,0,0,0,0,0,1)$
67^2 mod n	$2^2 \cdot 3 \cdot 5^2$	$(0,0,1,0,0,0,0)$
69^2 mod n	$2^2 \cdot 11 \cdot 13$	$(0,0,0,0,0,1,1)$
74^2 mod n	$3^2 \cdot 11 \cdot 13$	$(0,0,0,0,0,1,1)$

- $\gcd(58 \cdot 61 \cdot 67 \cdot 69 + ((2^3) \cdot (3^2) \cdot (5^2) \cdot 11 \cdot 13), n) = n$;
- $\gcd(58 \cdot 61 \cdot 67 \cdot 74 + ((2^2) \cdot (3^3) \cdot (5^2) \cdot 11 \cdot 13), n) = 59$;

Quadratic and number field sieves

- Quadratic sieve, advanced variant of Dixon's algorithm
- Number field sieve(NFS), advanced variant of quadratic sieve
- NFS currently best known algorithm for factoring large RSA moduli
- Size of factor base: $e^{(0.96+\mathcal{O}(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$
- Running time: $e^{(1.923+\mathcal{O}(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$
- Notation: $L[n,\alpha] = e^{(\alpha+\mathcal{O}(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$

VSH - iterated hash function

- Let $N = pq$ be a public RSA modulus ($p \neq q$, both secret)
- Let $p_1, \ldots, p_k$ be public primes such that $\prod_{i=1}^{k} p_i < N$
  - Let $m = m_1, m_2, \ldots, m_{\ell k}$ be message, $m_i \in \{0, 1\}$
  - $x_0 = 1$
  - $x_1 = x_0^2 \, (p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}) \bmod N$
  - $x_{j+1} = x_j^2 \, \prod_{i=1}^{k} p_i^{m_{jk+i}} \bmod N$
  - $\text{Hash}(m) = x_\ell$

Security of VSH

VSSR Problem. Let $n = pq$ be a public RSA modulus (p, q secret). Let $k \leq (\log n)^c$. Find $x \in Z_n^*$ such

$$x^2 \equiv \prod_{i=0}^{k} p_i^{e_i} \bmod n,$$

where at least one e_i is odd.

VSSR Assumption: The VSSR Problem is hard.

Computational VSSR Assumption:
Solving VSSR for n is as hard as to factor an S-bit modulus, where S is least positive integer satisfying

$$L[2^S, 1.923] \geq \frac{L[n, 1.923]}{k}.$$

$L[n, \alpha] = e^{(\alpha + \mathcal{O}(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}}$

VSH - security, speed

- Consider VSH with 1024-bit n, which allows for $k = 131$
- Then assumptions imply
  - a collision for VSH $\Rightarrow$ a solution to VSSR
  - a solution to VSSR as hard as factoring an 840-bit modulus
- Implementation:
  - 3 modular mult's per k bits
  - speed-up from precomputation

VSH - problems (?)

- Algebraic properties, e.g., easy to find messages with hash values h and $2h$
- Easy to invert hash for messages of small length
- VSH has multiplicative property (Saarinen 2006):

$$H(z)H(a \vee b) = H(a)H(b) \bmod n,$$

  for z the all-zero bit string, $a \wedge b = z$, and $|z| = |a| = |b|$.
- Someone must choose n such that p, q remain secret

VSH - defense

- Designers of VSH only aim for collision-resistance.
- VSH not to be used as replacement for random oracle nor SHA-1
  "random oracles do not exist in the real world, and therefore relying on them too much is not recommended"
- Potential use in schemes which require only collision-resistance, example, Cramer-Shoup signatures, which relies on strong RSA assumption and collision-resistant hash function
- One of the best attempts to build hash function on number-theoretic problem so far..

DaKoTa

- DaKoTa, a hash function co-designed by
  - Damgård, Ivan B.
  - Knudsen, Lars R.
  - Thomsen, Søren S.
- Uses combination of modular arithmetic and symmetric crypto

Based on factoring (Goldwasser, Micali, Rivest)

- $N = pq$, $p \neq q$, large primes, a_0, a_1 random squares modulo N
- Public: N, a_0, a_1

$$h : \{0,1\} \times Z_N^* \rightarrow Z_N^*$$

$$h(b, y) = a_b \; y^2 \bmod N$$

- Collision gives x, x' such that $x^2 = x'^2 \bmod N \rightarrow$ factoring
- More efficient variants with more squares $a_0, \ldots, a_k$, Damgård

Getting to DAKOTA

- $n = pq$, $p \neq q$, large primes, $p \equiv q \equiv 3 \bmod 4$
- Public: n, f
- $h : \{0,1\} \times \mathrm{SQ}(n) \rightarrow \mathrm{SQ}(n)$
- $f : \{0,1\} \rightarrow \mathrm{SQ}(n)$

$$h(b, y) \quad = \quad f(b) \; y^2 \bmod n$$

Getting closer to DAKOTA

- $n = pq$, $p \neq q$, large primes, $p \equiv q \equiv 3 \bmod 4$
- Public: n, f
- $h : \{0,1\}^k \times \mathrm{SQ}(n) \to \mathrm{SQ}(n)$
- $f : \{0,1\}^k \to \mathrm{SQ}(n)$

$$h(x,y) \quad = \quad f(x)\ y^2 \bmod n$$

Getting even closer to DAKOTA

- $n = pq$, $p \neq q$, large primes, $p \equiv q \equiv 3 \bmod 4$
- Public: n, f
- $h : \{0,1\}^k \times \mathrm{SQ}(n) \to \mathrm{SQ}(n)$
- $f : \{0,1\}^k \to \mathbf{Z}_n$

$$h(x,y) \quad = \quad f(x)^2\ y^2 \bmod n$$

Arriving at DAKOTA

- $n = pq$, $p \neq q$, large primes, $p \equiv q \equiv 3 \bmod 4$
- Public: n, f
- $h : \{0,1\}^k \times \mathrm{SQ}(n) \to \mathrm{SQ}(n)$
- $f : \{0,1\}^k \to \mathbf{Z}_n$

$$h(x,y) \quad = \quad (f(x)\ y)^2 \bmod n$$

DAKOTA- an iterated hash function

- $n = pq$, $p \equiv q \equiv 3 \bmod 4$, public: n, f
- $h : \{0,1\}^k \times \mathrm{SQ}(n) \to \mathrm{SQ}(n)$ $\qquad f : \{0,1\}^k \to \mathbf{Z}_n$

$$h(x,y) \quad = \quad (f(x)\ y)^2 \bmod n$$

- Choose $r \in \mathbf{Z}_n^*$, let $s = r^2 \bmod n$
- Split padded message x into k-bit words, $x_1, \ldots, x_t$
- Set $y_0 = s$, then compute

$$y_i = h(x_i, y_{i-1}) = (f(x_i)y)^2 \bmod n$$

- Hash of x is then y_t.

DAKOTA- an iterated hash function $h(x, y) = (f(x)y)^2 \bmod n$

Assumption

Consider probabilistic polynomial time algorithm with input f, n, and output $x, \tilde{x}, z$. Probability is negligible that

$$x \neq \tilde{x} \quad \text{and} \quad f(x)/f(\tilde{x}) = \pm z^2 \bmod n$$

Theorem

Hash function H is collision intractable under Assumption

find collision with prob $\epsilon \longrightarrow$ break Assumption with prob $\epsilon/2$.

DAKOTA- Assumption $h(x, y) = (f(x)y)^2 \bmod n$

Assumption

Consider probabilistic polynomial time algorithm with input f, n, and output $x, \tilde{x}, z$. Probability is negligible that

$$x \neq \tilde{x} \quad \text{and} \quad f(x)/f(\tilde{x}) = \pm z^2 \bmod n$$

- f must be one-way: choose $z, \tilde{x}$, compute x
- f must be coll. resistant: find collision for f, let $z = 1$
- no circular argument?, since f does (need to) not compress

DAKOTA- Proposal 1 for f $h(x, y) = (f(x)y)^2 \bmod n$

$$f : \{0, 1\}^k \longrightarrow \mathbf{Z}_n$$

- Let n and n' be 1025-bit resp. 1024-bit RSA moduli
- Let $k = 1022$
- Let $u = x^2 \bmod n'$, where $x < n'/2$
- Let $v = E_{\kappa_1}(u) = v_1 \mid \cdots \mid v_8$, where E is CBC encryption using AES
- Let $f(x) = E_{\kappa_2}(v_8 \mid \cdots \mid v_1)$
- f is one-way, collision-resistant

DAKOTA- Proposal 2 for f $h(x, y) = (f(x)y)^2 \bmod n$

$$f : \{0, 1\}^k \longrightarrow \{0, 1\}^k$$

- Let n be 1025-bit RSA-modulus, let $k = 1024$
- $f(x) = g(x) \oplus x$, where g is permutation of 1024 bits
- proposal for g:
  - tranform x into 8×8 matrix A with 16-bit values
  - Do 4 times
    $$A \leftarrow E(A)^T,$$
    where E is AES encryption (fixed key) of every column

Dakota- Performance

Hash function	Approximate speed (cycles/byte)	
	32-bit	64-bit
SHA-256	20	20
VSH	840	?
Dakota (Proposal 1)	385	170
Dakota (Proposal 2)	330	170

Concluding remarks

- 1980s: Hash functions based on block ciphers
- 1990s:
  - Dedicated, faster hash functions (Rivest-kickoff)
  - Many broken block cipher based hash function proposals
- 2000s:
  - Many dedicated schemes have been broken in later years
  - Many new constructions
- Renaissance of block cipher based proposals
- Renaissance of constructions with proofs of security
- SHA-3, likely to become “big SHA-1”, speed issue