News from the Rabin-Williams front

D. J. Bernstein

Thanks to:
University of Illinois at Chicago
NSF CCR–9983950
Alfred P. Sloan Foundation
Math Sciences Research Institute
University of California at Berkeley
American Institute of Mathematics
Keys

In 30-digit Rabin-Williams, a secret key is a pair of primes $p, q \in [0.5 \cdot 10^{15}, 10^{15}]$ with $p \mod 8 = 3$, $q \mod 8 = 7$. Corresponding public key: $pq$.

(RSA: Similar.)
Normal key generation

User generates

*random* secret key \((p, q)\)

with (e.g.) uniform distribution.

Easy way to do this:
Generate uniform random 15-digit \(p\).
Generate uniform random 15-digit \(q\).
If \((p, q)\) is not a secret key, try again.
Top-first key generation

Hard way to do the same thing:
1. Generate random 15-digit $t$ with the right distribution.
2. Generate uniform random $p, q$ such that $t = \text{top 15 digits of } pq$.

Basic idea of step 2:
Generate $p$ first;
choose $q$ near $10^{15} t/p$.

(Slightly non-uniform distribution is somewhat easier, faster.)
Key compression to 1/2 size
(known for many years)
Top-first allows public keys
to be compressed to 15 digits.
All users share the same $t$.
User 1 generates $p_1, q_1$ such that
$t = \text{top 15 digits of } p_1 q_1$.
User 2 generates $p_2, q_2$ such that
$t = \text{top 15 digits of } p_2 q_2$.
Each key has 30 digits,
but top 15 digits are shared.
Key compression to 1/3 size

(Coppersmith 2003)

For appropriate distribution of \( t \), can generate random \( p, q \) such that \( t = \text{top 20 digits of } pq \).

So public keys can be compressed to 10 digits.
Say \( t = 71382956724390183111 \).

Generate \( a, b \) such that 
\( ab \) starts \( 713829567243901 \):
e.g., \( a = 840889406630442 \),
\( b = 848898275582176 \),
\( 10^{10} t - ab = 423637965798208 \).

Lattices: Find small \( x, y \)
such that \( bx + ay \approx 10^{10} t - ab \):
e.g., \( x = 78379 \), \( y = -79125 \).

See if \( p = a + x \), \( q = b + y \) are prime.
Signatures

Rabin-Williams signature of message $m$ under public key $pq$ is vector $(e, f, r, s)$ such that $e \in \{-1, 1\}$, $f \in \{1, 2\}$, $r$ is a 256-bit string, $s$ is an integer, and

$$fs^2 \equiv eH(r, m) \pmod{pq}.$$ 

$H$ is a public hash function.
Security

Usual signing strategy (Rabin 1979): Signer chooses uniform random $r$, then obvious deterministic $e, f, s$.

Strategy gives security guarantee: Any forgery algorithm that works for all functions $H$ can be converted into an algorithm to factor $pq$ at similar speed.
Reducing randomness

Alternate strategy (Barwood 1997, independently Wigley 1997): Choose $r$ deterministically as a secret hash of $m$.

Strategy gives security guarantee even if $r$ is only 1 bit instead of 256 bits.

(Katz, Wang 2003)

cr.yp.to/sigs.html#rwtight