Proving tight security for Rabin-Williams signatures

Daniel J. Bernstein *
Department of Mathematics, Statistics, and Computer Science (M/C 249)
The University of Illinois at Chicago, Chicago, IL 60607–7045
djb@cr.yp.to

February 20, 2007

Abstract

This paper proves "tight security in the random-oracle model relative to factorization" for the lowest-cost signature systems available today: every hash-generic signature-forging attack can be converted, with negligible loss of efficiency and effectiveness, into an algorithm to factor the public key. The most surprising system is the "fixed unstructured $B = 0$ Rabin-Williams" system, which has a tight security proof despite hashing unrandomized messages. At a lower level, the three main accomplishments of the paper are (1) a "$B \geq 1$" proof that handles some of the lowest-cost signature systems by pushing an idea of Katz and Wang beyond the "claw-free permutation pair" context; (2) a new expository structure, elaborating upon an idea of Koblitz and Menezes; and (3) a proof that uses a new idea and that breaks through the "$B \geq 1$" barrier.

1 Introduction

Variants of the Rabin-Williams public-key signature system have, for twenty-five years, held the speed records for signature verification. Are these systems secure?

There are many other signature systems of RSA/Rabin type. One can break each system by computing roots modulo the signer's public key pq or by breaking the system's hash function H. Are there other attacks? This is not an idle concern: some RSA-type systems have been broken by embarrassing attacks that (1) are much faster than known methods to compute roots modulo pq and (2) work for every function H (or a large fraction of choices of H), given oracle access to H.

Some systems have been proven immune to embarrassing attacks. For example, in the 1993 paper [3] that popularized this line of work (along with the terminology "secure in the random-oracle model"), Bellare and Rogaway proved the following security property for the traditional "FDH" form of exponent-e RSA: every H-generic attack on RSA-FDH can be converted (without serious loss of efficiency) into an algorithm to compute eth roots modulo pq.

Unfortunately, a closer look reveals that most of these proofs merely limit the embarrassment, without actually ruling it out. For example, the Bellare-Rogaway root-finding algorithm has only a $1/Q$ chance of success, where Q is the number of hash values seen by the FDH attack. Coron in [7]

*Date of this document: 2007.02.20. Permanent ID of this document: c30057d690a8fb42af6a5172b5da9006.

introduced a better algorithm having a $1/S$ chance of success, where S is the number of signatures seen by the FDH attack; but S can still be quite large.

Randomized signatures, in which B-bit random strings are prepended to messages before the messages are signed, allow much tighter proofs if B is large. For example, every H-generic attack on randomized exponent-e RSA (or Rabin's 1979 signature system) can be converted into a algorithm to compute eth roots modulo pq (or to factor pq) with a good chance of success. But generating random strings takes time, and transmitting the strings consumes bandwidth. Can we do better?

A 2002 theorem of Coron is widely interpreted as saying that FDH is stuck at $1/S$, i.e., that tight proofs require randomization of hash inputs; see [9]. A tight security proof by Katz and Wang in [12] allows much shorter random strings for some RSA variants but breaks down for Rabin-Williams. There are other systems with tight security proofs, but none of them offer state-of-the-art efficiency.

Contributions. This paper proves tight security for several state-of-the-art variants of the Rabin-Williams public-key signature system. What's most surprising is the "fixed unstructured $B = 0$" variant, a specific type of FDH that has a tight security proof despite hashing unrandomized messages. In the Rabin-Williams context, a minor technical assumption in Coron's theorem—the assumption of "unique" signatures—turns out to be a major loophole, producing a tight security proof from a random choice *later* in the signing process, after all hashing is done.

There are actually two security proofs in this paper. The "$B \geq 1$" proof uses a more general approach, pushing the Katz-Wang idea beyond the well-known "claw-free permutation pair" setting and carefully handling the "tweaked square roots" that appear in the Rabin-Williams system. The "unstructured $B = 0$" proof relies on a new proof idea that is more specific but also responsible for the aforementioned surprise. As far as I can tell, the new proof idea is tied to Rabin-Williams and cannot say anything useful about RSA; within the Rabin-Williams context, the new proof idea is tied to "unstructured" signatures and does not cover "principal" or "|principal|" signatures. The specific case of "fixed unstructured $B = 0$" Rabin-Williams signatures is nevertheless worth study because it is a state-of-the-art signature system of particular interest to implementors; among all high-speed systems with tight security proofs it is the only one that does not need to randomize hash inputs.

My proofs follow a new expository structure that explicitly separates five levels of hard problems (generic blind inversion, generic selective inversion using one signature, generic selective inversion using many signatures, generic existential inversion, and generic attacks) and that closely tracks the intuition of a cryptanalyst studying potential attacks. My impression is that previous proofs can be, and would gain in clarity from being, rewritten within the same structure; for example, the structure formalizes the standard idea of a simulator as a very easy step from the first-level hard problem to the second-level hard problem, with all other difficulties stripped away. I can easily imagine the structure being useful for future proofs in the area. I hope that one proof suffices to make the general structure clear.

These proofs owe a heavy debt to the efforts of Koblitz and Menezes to clarify the limits of "provable security"; see [18] and [19]. In particular, in [18, Section 3.2], in the case of RSA with $B = 0$, Koblitz and Menezes explicitly stated an apparently new "RSA1" hard problem (which I call "generic existential inversion") and conjectured that it had the same difficulty as the usual hard problem (which I call "generic blind inversion"). The simplicity and clarity of the new hard problem inspired me to consider the analogous problem for Rabin-Williams. Koblitz and Menezes had commented that Coron's $1/S$ reduction could be translated to a $1/S$ reduction between these two hard problems, and that it was unreasonable to hope for a better reduction in light of Coron's

2002 theorem. I was quite surprised to discover that the "unstructured" case of the analogous Rabin-Williams conjecture could in fact be *proven*.

Acknowledgments and priority dates: Thanks to Dan Boneh for pointing out [12] to me. I posted a "$B \geq 1$" Katz-Wang-style proof in 2003. I posted the new expository structure in November 2006. I posted the "unstructured $B \geq 0$" proof, with the new proof idea, in November 2006.

2 Parameters; keys; verification; signing

This section defines the family of signature systems whose security is analyzed later in the paper. Standardizing a particular signature system in the family means standardizing various parameters: K, the number of key bits; D, the distribution of secret keys; H, the hash function; and B, the number of bits of randomization of the hash input. The signer's behavior is further controlled by two parameters relevant to security: first, whether signatures are "unstructured" or "principal" or "|principal|"; second, whether signatures are "fixed" or "variable." All of these parameters are explained in detail below.

Readers wondering "Why are you looking at these particular systems?" should consult Appendix A for a cost analysis and a historical survey. The short answer is that, among all the signature systems that are conjectured to provide a reasonable security level, these systems were engineered to minimize cost. Exception: in applications where signature length is much more important than verification time, lower costs are achieved by systems of ElGamal/Schnorr/ECDSA type.

Secret keys and public keys. All users of the system know an integer $K \geq 10$. Typical choices of K include 1024 (not recommended), 1536, and 2048. All users of the system also know a distribution D (for example, the uniform distribution) of pairs of prime numbers (p, q) such that $p \in 3 + 8\mathbf{Z}$, $q \in 7 + 8\mathbf{Z}$, and $2^K < pq < 2^{K+1}$.

Each user of the system chooses a random secret key (p, q) from the distribution D, and computes a corresponding public key pq.

The difficulty of factoring pq depends on the parameters (K, D); no security is possible when these parameters are chosen poorly. If $K = 512$, for example, then the attacker can use the number-field sieve to factor arbitrary integers between 2^K and 2^{K+1} with a moderate amount of effort, and can then freely forge signatures. As another example, if D has highly limited randomness and is concentrated on 2^{32} pairs (p, q), the attacker can factor pq by simply trying each of those 2^{32} pairs.

Theoreticians often simplify this picture by assuming that D is the uniform distribution. In the real world, however, implementors often choose non-uniform distributions to save time in key generation. This paper considers arbitrary distributions of pairs (p, q), and thus arbitrary distributions of public keys pq. For each distribution D, this paper proves that various hard problems involving public keys from distribution D are equivalent to factoring public keys from distribution D.

Hashing and verification. All users of the system know an integer $B \geq 0$. Three interesting choices of B are 0, 1, and 128. All users of the system also know a function $H : \{B\text{-bit strings}\} \times \{\text{messages}\} \to \{1, 2, \ldots, 2^K\}$. For example, for $B = 0$ and $K = 2048$, the function H assigns an element of $\{1, 2, \ldots, 2^{2048}\}$ to each message. There are many popular choices of H, usually built from components such as MD5, SHA-1, and SHA-256.

A vector (e, f, s) is a **tweaked square root** of an integer h modulo a public key pq if $e \in \{1, -1\}$; $f \in \{1, 2\}$; $s \in \{0, 1, \ldots, pq - 1\}$; and $efs^2 \equiv h \pmod{pq}$. A vector (e, f, r, s) is a **signature** of

a message m under a public key pq if r is a B-bit string and (e, f, s) is a tweaked square root of $H(r, m)$.

For example, the algorithm displayed in Appendix A computes a tweaked square root of h, specifically the **principal tweaked square root** of h. This is the unique tweaked square root (e, f, s) such that e is 1 if h is a square modulo q, otherwise -1; f is 1 if eh is a square modulo p, otherwise 2; and s is a square modulo pq.

The difficulty of forging signatures depends on H. No security is possible when the hash function is chosen poorly. For example, if $H(r, m)$ is determined by MD5(m), then an attacker can find collisions in H by finding collisions in MD5.

Reader beware: Many authors allow the output range of H to be a function of the public key, but there cannot actually be any such dependence when H is a system parameter shared by all users, as it always is in practice. Putting a shared limit on the output range of H also means slightly changing the notion of a generic attack, and slightly changing the security proofs. My proofs include these minor changes.

Unstructured signatures, principal signatures, |principal| signatures. Each message m has exactly 2^{B+2} signatures under pq: there are 2^B choices of r, and then 4 choices of tweaked square root (e, f, s) of $H(r, m)$ modulo pq. Which signature does the signer choose?

A stupid signer could easily expose his secret key to the attacker through this choice. For example, the signer could leak the ith bit of p in the ith signature as the bottom bit of r (if $B \geq 0$), as the Jacobi symbol of s modulo pq, etc. This example demonstrates that there is no hope of security if the signing function is chosen poorly. How do we know that a smarter-sounding signing algorithm does not have a similar leak?

There are three signature distributions proposed in the literature:

- **Unstructured**: The signer chooses a uniform random string r, and then a uniform random tweaked square root of $H(r, m)$, independently of all previous choices.
- **Principal**: The signer chooses a uniform random string r independently of all previous choices, and then chooses the principal tweaked square root of $H(r, m)$.
- **|Principal|**: The signer chooses a uniform random string r independently of all previous choices, and then chooses the "|principal|" tweaked square root of $H(r, m)$. If the principal tweaked square root is (e, f, s) then the |principal| tweaked square root is $(e, f, \min\{s, pq - s\})$; the point is that $\min\{s, pq - s\}$ takes a bit less space than s.

One step in this paper's security proofs—see Section 4—is split into three cases accordingly. A later step—see Section 7—is affected much more dramatically by the choice.

Variable signatures, fixed signatures. What happens if the signer is given the same message to sign once again? There are two choices in the literature:

- **Fixed**: Given the same message again, the signer chooses the same signature again.
- **Variable**: Given the same message again, the signer generates a fresh signature, making random choices independently of the previous choices.

To understand the importance of this choice for security, consider the fact that "fixed unstructured $B = 0$" signatures now have a tight security proof, whereas "variable unstructured $B = 0$" signatures are easily breakable. The importance of this choice for tight security *proofs* was pointed

out by Katz and Wang in [12]; the conventional wisdom before [12] was that tight security proofs required a large B.

The above description of fixed signatures might sound as if each signer needs to remember all previous signed messages. However, the signer can produce indistinguishable results without memory, assuming standard conjectures in secret-key cryptography. The trick is simple: the signer replaces the random bits with secret functions of m. This trick was posted to `sci.crypt` in 1997 by Barwood and independently by Wigley; it was reinvented several years later by Katz and Wang in [12]. The usual construction is for the signer to first compress m with a very fast "almost universal hash function," and then scramble the result, for example with a block cipher; the relevant conjecture is that the scrambling function is indistinguishable from uniform.

For principal and |principal| signatures with $B = 0$, no randomness is required, and variable signatures are the same as fixed signatures.

3 Generic blind inversion

The security proof in this section is well known, but readers are nevertheless encouraged to read it as a warmup for the security proofs in subsequent sections.

Suppose we are given a public key pq and an integer $h' \in \{1, 2, \ldots, 2^K\}$. How quickly can we compute a tweaked square root of h' modulo pq? One approach is to factor pq; are there better approaches?

More formally: Fix K and D. Consider any algorithm A_1 that, given an integer n with $2^K < n < 2^{K+1}$ and an integer $h' \in \{1, 2, \ldots, 2^K\}$, computes a vector $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, n-1\}$. Define A_1 as **successful** if $e'f'(s')^2 \equiv h' \pmod{n}$. Define $X_1(A_1)$ as the probability that A_1 is successful when n is a random public key from the distribution D, h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$, and n, h' are independent. How large can $X_1(A_1)$ be, as a function of the resources consumed by A_1?

An always-successful fast algorithm A_1 for this generic-blind-inversion problem immediately implies an always-successful fast algorithm to forge Rabin-Williams signatures, given oracle access to the hash function H. The attacker simply chooses the message m' that he wants to sign, chooses any B-bit string r', computes $h' = H(r', m')$, and computes a tweaked square root (e', f', s') of h'. Then (e', f', r', s') is a signature of m'. Conversely, cryptanalysts trying to forge Rabin-Williams signatures will naturally consider this simple attack strategy as a first possibility.

Tight security proof. Unfortunately for the cryptanalyst, this problem is provably as difficult as factorization of public keys. Any successful fast algorithm A_1 for this problem immediately implies a successful fast factorization algorithm A_0. The proof is completely standard, except for the details of how the tweaks e, f are handled.

Write $X_0(A_0)$ for the chance that an algorithm A_0, given a random public key n with distribution D, computes a nontrivial factor of n. Starting from A_1, consider the following factorization algorithm A_0:

1. Generate a uniform random vector $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
2. Compute $h' = efs^2 \bmod n$.
3. Start over if $h' \notin \{1, 2, \ldots, 2^K\}$.
4. Compute $(e', f', s') = A_1(n, h')$.

5. If $\gcd\{n, s' - s\} \notin \{1, n\}$, print it and stop.
6. If $\gcd\{n, s'\} \notin \{1, n\}$, print it and stop.

The following theorem states that a large success chance $X_1(A_1)$ implies a large factorization chance $X_0(A_0)$. The time of A_0 is practically identical to the time of A_1: the only difference is a few easy operations modulo n to generate h', repeated only $n/2^K < 2$ times on average.

Theorem 3.1. $X_0(A_0) \geq (1/2)X_1(A_1)$.

Proof. Let n be a random public key with distribution D. At step 4 of $A_0(n)$, the quantity $h' = efs^2 \bmod n$ is a uniform random element of $\{1, 2, \ldots, 2^K\}$; recall that each choice of h' is produced by exactly four choices of e, f, s. Thus the event $e'f'(s')^2 \equiv h' \pmod n$ occurs with probability exactly $X_1(A_1)$. I claim that, given this event, there is conditional probability at least $1/2$ that one of $s', s' - s$ has a nontrivial factor in common with n.

Case 1: $\gcd\{h', n\} = n$. This is impossible, since $1 \leq h' \leq 2^K < n$.

Case 2: $\gcd\{h', n\} = p$. In this case $\gcd\{s', n\} = p$ as desired.

Case 3: $\gcd\{h', n\} = q$. In this case $\gcd\{s', n\} = q$ as desired.

Case 4: $\gcd\{h', n\} = 1$. I claim that $(s')^2 \equiv s^2 \pmod n$. Notice first that $e'f'(s')^2 \equiv efs^2 \pmod n$, and recall that $n = pq$ for primes p, q with $p \in 3 + 8\mathbf{Z}$ and $q \in 7 + 8\mathbf{Z}$. Both possibilities for f, namely 1 and 2, are squares modulo q, so $f'(s')^2$ and fs^2 are squares modulo q, and both are nonzero since $\gcd\{h', q\} = 1$; the ratio e'/e is therefore a square modulo q and hence cannot be -1. Consequently $e' = e$ and $f'(s')^2 \equiv fs^2 \pmod n$. Both $(s')^2$ and s^2 are squares modulo p, and both are nonzero since $\gcd\{h', p\} = 1$; the ratio f'/f is therefore a square modulo p and hence cannot be 2. Hence $f' = f$ and $(s')^2 \equiv s^2 \pmod n$.

Recall that there are exactly four choices of e, f, s consistent with h', and observe that e', f', s' is independent of this choice. All four choices have the same e, f as I just showed, so only two of them have $s \equiv s'$ or $s \equiv -s'$. The other two choices occur with conditional probability $1/2$; for those choices, n divides $(s')^2 - s^2$ without dividing $s' - s$ or $s' + s$, so $\gcd\{n, s' - s\}$ is a nontrivial factor of n. □

4 Generic selective inversion using one signature

Suppose we're given a public key pq, two integers $h, h' \in \{1, 2, \ldots, 2^K\}$, and a tweaked square root (e, f, s) of h modulo pq. How quickly can we compute a tweaked square root of h' modulo pq? One approach is to factor pq; are there better approaches?

More formally: Fix K and D. Fix $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Consider any algorithm A_2 that, given an integer n with $2^K < n < 2^{K+1}$, an integer $h \in \{1, 2, \ldots, 2^K\}$, a vector $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, n-1\}$, and an integer $h' \in \{1, 2, \ldots, 2^K\}$, computes a vector $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, n-1\}$. Define A_2 as **successful** if $e'f'(s')^2 \equiv h' \pmod n$. Define $X_2(A_2)$ as the probability that A_2 is successful when n is a random public key from the distribution D, h is a uniform random element of $\{1, 2, \ldots, 2^K\}$, (e, f, s) is a random tweaked square root of h modulo n with distribution α, h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$, and all of these choices are independent. How large can $X_2(A_2)$ be, as a function of the resources consumed by A_2?

This problem is a natural step for the cryptanalyst beyond the generic-blind-inversion problem in Section 3. Any always-successful algorithm A_2 for this problem immediately implies an always-successful fast algorithm to forge Rabin-Williams signatures, given oracle access to the hash function H. The forgery algorithm takes h and (e, f, s) from a legitimately signed message m, chooses a message $m' \neq m$, chooses a B-bit string r', computes $h' = H(r', m')$, computes $(e', f', s') = A_2(pq, h, e, f, s, h')$, and outputs (e', f', r', s') as a successful forgery of m'.

Similar comments apply to the problems articulated in subsequent sections. Each problem is a natural problem for the cryptanalyst to consider, providing more flexibility than the previous problem and potentially making attacks easier.

Tight security proof. Unfortunately for the cryptanalyst, this problem is provably as difficult as factorization of public keys. Any successful fast algorithm A_2 for this problem immediately implies a successful fast algorithm A_1 for the generic-blind-inversion problem, and therefore implies a successful fast factorization algorithm A_0.

The intuition here is that A_2 learns nothing from seeing h, e, f, s. It is well known how to formalize this intuition: namely, build a **simulator** that, given n, generates (h, e, f, s) with exactly the same distribution as a signer who first generates h and then uses the factorization of n to generate (e, f, s).

There are three different constructions of the simulator, and therefore three different constructions of A_1, one for each of the three choices of α. Here is a construction of A_1 for the simplest choice, $\alpha =$ unstructured:

1. Generate a uniform random vector $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
2. Compute $h = efs^2 \bmod n$.
3. Start over if $h \notin \{1, 2, \ldots, 2^K\}$.
4. Print $A_2(n, h, e, f, s, h')$.

Here is a construction of A_1 for $\alpha \in \{\text{principal}, |\text{principal}|\}$:

1. Generate a uniform random vector $(e', f', x) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
2. Compute $g = \gcd\{x, n\}$.
3. If $g = n$ or $g \bmod 8 = 7$, set $e = 1$; otherwise set $e = e'$.
4. If $g = n$ or $g \bmod 8 = 3$, set $f = 1$; otherwise set $f = f'$.
5. Compute $s = x^2 \bmod n$.
6. Compute $h = efs^2 \bmod n$.
7. Start over if $h \notin \{1, 2, \ldots, 2^K\}$.
8. Print $A_2(n, h, e, f, s, h')$ if $\alpha =$ principal, else $A_2(n, h, e, f, \min\{s, n-s\}, h')$.

The following theorem states that a large success chance $X_2(A_2)$ implies a large factorization chance $X_1(A_1)$. The time of A_1 is practically identical to the time of A_2: the only difference is a few easy operations modulo n to generate h', repeated only $n/2^K < 2$ times on average.

Theorem 4.1. $X_1(A_1) = X_2(A_2)$.

The reader may have noticed that the constructions of A_1, in the principal and |principal| cases, go to some extra work to handle extremely rare events such as $g = n$. The reward for this work is a particularly clean theorem. The simulators produce *exactly* the right output distribution, rather than producing *almost exactly* the right output distribution and forcing the user to worry about the difference.

Proof. Consider $A_1(n, h')$, where n is a random public key with distribution D, h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$, and n, h' are independent.

Unstructured: There are exactly four choices of (e, f, s) for each possible h; so the distribution of h is uniform, and (e, f, s) is a uniform random tweaked square root of h. Thus $e'f'(s')^2 \equiv h'$ with probability exactly $X_1(A_1)$.

Principal: If $e = 1$ then $h \equiv efs^2 = fs^2$ is a square modulo q since 2 is a square modulo q. If $e = -1$ then $h \equiv efs^2 = -fs^2$, which I claim is a non-square modulo q; otherwise q divides s, so q divides x, so $g = \gcd\{x, n\} \in \{n, q\}$, so $g = n$ or $g \bmod 8 = 7$, so $e = 1$, contradiction. Similarly, if $f = 1$ then $eh \equiv s^2$ is a square modulo p, and if $f = -1$ then $eh \equiv -s^2$, which I claim is a non-square modulo p; otherwise p divides s, so p divides x, so $g = \gcd\{x, n\} \in \{n, p\}$, so $g = n$ or $g \bmod 8 = 3$, so $f = 1$, contradiction. Furthermore, by construction s is a square modulo n. Therefore (e, f, s) is the principal tweaked square root of h. The only remaining task is to show that the distribution of h is uniform.

Which choices of (e', f', x) lead to h? Write (e, f, s) for the principal tweaked square root of h. If $\gcd\{h, n\} = 1$ then $\gcd\{s, n\} = 1$ so $g = \gcd\{x, n\} = 1$; thus $e' = e$, $f' = f$, and x is one of the four square roots of s modulo n. If $\gcd\{h, n\} = p$ then $\gcd\{s, n\} = p$ so $g = \gcd\{x, n\} = p$; thus $e' = e$, $f' \in \{1, 2\}$, and x is one of the two square roots of s modulo n. If $\gcd\{h, n\} = q$ then $\gcd\{s, n\} = q$ so $g = \gcd\{x, n\} = q$; thus $e' \in \{-1, 1\}$, $f' = f$, and x is one of the two square roots of s modulo n. If $\gcd\{h, n\} = n$ then $\gcd\{s, n\} = n$ so $g = \gcd\{x, n\} = n$; thus $e' \in \{-1, 1\}$, $f \in \{1, 2\}$, and $x = 0$. To summarize, each integer $h \in \{0, 1, \ldots, n-1\}$ is produced by *at most* four choices of (e', f', x). There are n possibilities for h and $4n$ possibilities for (e', f', x), so each integer $h \in \{0, 1, \ldots, n-1\}$ is produced by *exactly* four choices of (e', f', x).

|Principal|: As above h is uniform, and (e, f, s) is the principal tweaked square root of h, so $(e, f, \min\{s, n - s\})$ is the |principal| tweaked square root of h. □

5 Generic selective inversion using many signatures

Suppose we're given a public key pq, integers $h_1, h_2, \ldots, h_Q, h' \in \{1, 2, \ldots, 2^K\}$, and a tweaked square root of each h_i modulo pq. How quickly can we compute a tweaked square root of h' modulo pq? One approach is to factor pq; are there better approaches?

More formally: Fix K and D. Fix $Q \geq 0$. Fix $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Consider any algorithm A_3 that, given an integer n with $2^K < n < 2^{K+1}$, a vector $(h_i, e_i, f_i, s_i) \in \{1, 2, \ldots, 2^K\} \times \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, n-1\}$ for each $i \in \{1, 2, \ldots, Q\}$, and an integer $h' \in \{1, 2, \ldots, 2^K\}$, computes a vector $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, n-1\}$. Define A_3 as **successful** if $e'f'(s')^2 \equiv h' \pmod{n}$. Define $X_3(A_3)$ as the probability that A_3 is successful when n is a random public key from the distribution D, each h_i is a uniform random element of $\{1, 2, \ldots, 2^K\}$, (e_i, f_i, s_i) is a random tweaked square root of h_i modulo n with distribution α, h' is

a uniform random element of $\{1, 2, \ldots, 2^K\}$, and all of these choices are independent. How large can $X_3(A_3)$ be, as a function of the resources consumed by A_3?

The answer is that this problem is provably as difficult as factorization of public keys. The construction of A_1 from A_3 is an easy generalization of last section's construction of A_1 from A_2. For example, here is A_1 for $\alpha =$ unstructured:

1. For each $i \in \{1, 2, \ldots, Q\}$:
2. Generate a uniform random vector (e_i, f_i, s_i) in the usual range.
3. Compute $h_i = e_i f_i s_i^2 \bmod n$.
4. Go back to step 2 if $h_i \notin \{1, 2, \ldots, 2^K\}$.
5. Print $A_3(n, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$.

The remaining constructions work similarly.

Theorem 5.1. $X_1(A_1) = X_3(A_3)$.

Proof. Exactly as in Section 4. □

6 Generic existential inversion: the unstructured $B = 0$ case

Suppose we're given a public key pq and integers $h_1, \ldots, h_{Q+1} \in \{1, 2, \ldots, 2^K\}$. We're allowed to (adaptively) select Q distinct i's and see tweaked square roots of the corresponding h_i's. Our goal is to compute a tweaked square root of the *other* h_i. How quickly can we do this?

The big difference between this problem and the generic-selective-inversion problem in Section 5 is that we're now allowed to decide which of the h_i's will be easiest to attack. Does this make the problem easier? Perhaps we gain from the extra flexibility.

This section uses a new idea to show that there is no gain in the case of unstructured signatures. The reader might guess, after previous sections, that the proof constructs an algorithm for generic selective inversion (A_2, A_3) or generic blind inversion (A_1); in fact, the proof jumps directly to A_0, the factorization problem. I don't know any way to get from a generic-existential-inversion algorithm A_4 to A_1, in the case $B = 0$, except via A_0.

The new idea. Let's review the standard proof that the gain is at most a factor $Q + 1$. Given a generic-existential-inversion algorithm A_4, build a generic-selective-inversion algorithm A_3 that handles inputs $(n, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$ as follows:

- Choose a uniform random integer $\pi \in \{1, \ldots, Q + 1\}$.
- Insert h' at position π in the list $h_1, \ldots, h_Q$, and relabel the resulting list as $h_1, \ldots, h_{Q+1}$. Also relabel e_i, f_i, s_i accordingly.
- Run $A_4(n, h_1, \ldots, h_{Q+1})$, using e_i, f_i, s_i to answer query i from A_4; abort if A_4 selects $i = \pi$ for a query rather than for output.

The choice of π is independent of the operation of A_4 before an abort occurs, so A_3 aborts with probability $Q/(Q + 1)$. If A_3 does not abort then it runs A_4 with the correct input distribution.

The random choice of π is a guess for the index i that A_4 will use for its output. When a correct guess does occur, it makes the generic-existential-inversion problem equivalent to the generic-selective-inversion problem, eliminating the extra flexibility of the generic-existential-inversion problem.

Now let's review how this algorithm A_3 is converted into a factorization algorithm. First A_3 is converted into a generic-blind-forgery algorithm A_1: the input h_i is replaced by an output from the simulator. Then A_1 is converted into a factorization algorithm A_0: the input h' is replaced by a random efs^2, so that a tweaked square root of h' reveals a factorization of n.

Wait a minute! What's happening to h_i is almost the same as what's happening to h'. In fact, with the unstructured simulator, what's happening to h_i is *exactly* the same as what's happening to h'! Why did we bother to distinguish h_i from h' in the first place? The new idea is to treat all of $h_1, \ldots, h_{Q+1}$ the same way, directly producing a factorization algorithm; there is no need to guess which one is h', and there is no need for a detour through A_3 and A_1.

Here is the new, direct, almost ludicrously simple construction of a factorization algorithm A_0 from a generic-existential-inversion algorithm A_4:

1. For each $i \in \{1, 2, \ldots, Q+1\}$:
2. Generate a uniform random vector (e_i, f_i, s_i) in the usual range.
3. Compute $h_i = e_i f_i s_i^2 \bmod n$.
4. Go back to step 2 if $h_i \notin \{1, 2, \ldots, 2^K\}$.
5. Compute $(j, e', f', s') = A_4(n, h_1, \ldots, h_{Q+1})$, using (e_i, f_i, s_i) to answer query i from A_4. There is no possibility of aborting here; we have an answer for every i!
6. If $\gcd\{n, s' - s_j\} \notin \{1, n\}$, print it and stop.
7. If $\gcd\{n, s'\} \notin \{1, n\}$, print it and stop.

The time for A_0 is the time for A_4, on average the time for $(Q+1)n/2^K < 2(Q+1)$ generations of h_i, and the time for the final gcd computations.

Theorem 6.1. $X_0(A_0) \geq (1/2)X_4(A_4)$.

Proof. Let n be a random public key with distribution D. Inside $A_0(n)$, the quantities $h_1, \ldots, h_{Q+1}$ are independent uniform random elements of $\{1, 2, \ldots, 2^K\}$. Thus the event $e'f'(s')^2 \equiv h_j \pmod{n}$ occurs with probability exactly $X_4(A_4)$. Given this event, there is conditional probability at least $1/2$ that one of $s', s' - s_j$ has a nontrivial factor in common with n, exactly as in Theorem 3.1. □

7 Generic existential inversion: the $B \geq 1$ case

Fix $B \geq 0$. Suppose we're given a public key pq and random access to integers $h_1(0), \ldots, h_1(2^B - 1), h_2(0), \ldots, h_2(2^B - 1), \ldots, h_{Q+1}(0), \ldots, h_{Q+1}(2^B - 1)$. We're allowed to (adaptively) select Q distinct i's; for each selected i we see a uniform random $r_i \in \{0, 1, \ldots, 2^B - 1\}$ and a tweaked square root of $h_i(r_i)$. Our goal is to compute some r and some tweaked square root of $h_i(r)$ for the remaining i. How quickly can we do this?

As usual, the answer depends on $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$, the signature-choice parameter. The case $\alpha =$ unstructured was discussed in Section 6. The tight security proof for unstructured signatures for $B = 0$ generalizes immediately to a tight security proof for unstructured signatures for all B. The initial computations of $h_i(r)$ might sound overly time-consuming when B is large, because there are $2^B(Q+1)$ pairs (i, r); but these computations can be deferred until they are actually needed.

What about $\alpha \in \{\text{principal}, |\text{principal}|\}$? There is a tight security proof for all $B \geq 1$, coming from a different way to build a factorization algorithm out of a generic-existential-inversion algorithm A_5. The construction, given n,

- chooses a uniform random r_i for each $i \in \{1, 2, \ldots, Q+1\}$;
- uses the α simulator to build $e_i(r_i), f_i(r_i), s_i(r_i), h_i(r_i)$;
- uses the *unstructured* simulator to build $e_i(r), f_i(r), s_i(r), h_i(r)$ for each $r \neq r_i$;
- runs A_5, answering query i with $r_i, e_i(r_i), f_i(r_i), s_i(r_i)$;
- aborts if the output j, r', e', f', s' has $r' = r_j$; and
- tries $\gcd\{s', n\}$ and $\gcd\{s' - s_j(r'), n\}$ as factors of n.

This algorithm aborts with probability exactly $1/2^B$: r_j is independent of everything seen by A_5 and therefore independent of r'. If the algorithm does not abort then it has conditional probability at least $1/2$ of factoring n, exactly as in Theorem 3.1.

Readers should recognize the central idea of this construction—choosing a random r_i, building $h_i(r_i)$ according to the target simulator, and building $h_i(r)$ for $r \neq r_i$ to solve the underlying hard problem—as exactly the Katz-Wang idea used to prove [12, Section 4.1, Theorem 2]. Readers should also recognize, however, that [12, Section 4.1, Theorem 2] is stated at the level of generality of "claw-free permutation pairs" (following [11] and a suggestion of Dodis and Reyzin), which might sound quite general but are incapable of handling Rabin-Williams signatures. Several exponent-2 "claw-free permutation pairs" have been stated in the literature, but all of them have considerably slower signature verification than the Rabin-Williams system. One can easily recognize claws in the Rabin-Williams context, but they are claws between a 4-to-1 map and a 1-to-1 map, with two different algorithms for generating the inputs to the two maps. I'm considering adding an appendix to this paper to state a "claw-free map pairs" generalization of the Katz-Wang theorem; if you're interested in this, please let me know.

For $B = 0$, the above construction accomplishes nothing: it never uses the unstructured simulator and always aborts. The construction needs at least one bit of hash-input randomization to separate the target simulator from the unstructured simulator. Eliminating the abort does not produce a security proof: if s_j was produced by (e.g.) the principal simulator then it is *not* a uniform random square root of its square and there is no reason to believe that $s' - s_j(r')$ will have a factor in common with n.

For $\alpha =$ unstructured, eliminating the abort *does* produce a security proof, and further eliminating the selection of r_i produces exactly the new construction of Section 6. This is another way to see both the limitations and the power of the new idea in Section 6: the construction refuses to distinguish the α simulator from the unstructured simulator, and therefore requires $\alpha =$ unstructured, but the construction also skips the selection of r_i, and therefore can handle $B = 0$.

8 Generic attacks

Let's review three typical examples of attacks on the Rabin-Williams system:

- NFS factorization: The attacker uses the number-field sieve to factor pq into (p, q). The attacker then chooses a message m', chooses a B-bit string r', computes $h' = H(r', m')$ using an oracle for H, uses (p, q) to compute a tweaked square root (e', f', s') of h', and forges the signature (e', f', r', s') of m'. This attack always succeeds, for all functions H. Fortunately, this attack is very slow when K is large.

- Signing leaks: The attacker chooses a message m and asks the signer for two signatures of m. The signer responds with (e_1, f_1, r_1, s_1) and (e_2, f_2, r_2, s_2). The attacker computes $\gcd\{s_2, n\}$ and $\gcd\{s_1 - s_2, n\}$, hoping to factor n and proceed as in the previous attack. In the case of variable unstructured $B = 0$ signatures, this attack succeeds with probability $\geq 1/2$, for all functions H: notice that $r_1 = r_2$ since $B = 0$, and therefore that $e_1 f_1 s_1^2 \equiv e_2 f_2 s_2^2$; continue as in Theorem 3.1. Fortunately, this attack does not work for fixed signatures, or for principal or |principal| signatures, or for signatures with large B.
- MD5 collisions: The attacker finds distinct messages m, m' such that $\text{MD5}(m) = \text{MD5}(m')$. The attacker asks the signer for a signature of m and then forges the same signature of m'. This attack works if $B = 0$ and H is determined by MD5, a surprisingly common situation in practice. Fortunately, one can easily change H to stop the attack.

The first two attacks work for all functions H. The third does not. Let's ignore the third attack for the moment.

Consider the class of "H-generic attacks" that work for all functions H—or that at least work for a *large fraction* of all functions H. How powerful are H-generic attacks against the Rabin-Williams system? Can they be better than factorization? Given an attack A_6 that uses H as an oracle, define $X_6(A_6)$ as the success probability of A_6 against a D-distributed random public key n and a uniformly distributed random function H. Can $X_6(A_6)$ be much larger than the other probabilities considered in this paper, as a function of the resources consumed by A_6?

The signing-leak example shows that these attacks can be quite successful: variable unstructured $B = 0$ signatures are broken by an extremely fast generic attack. But the picture is different for fixed signatures. For fixed signatures, generic attacks that see hash values of $Q + 1$ distinct messages are as difficult as Q-query generic existential inversion. A generic-attack algorithm A_6 is straightforwardly converted into a generic-existential-inversion algorithm A_5 as follows: A_5 runs A_6, keeps track of the distinct messages $m_1, m_2, \ldots, m_{Q+1}$ that are hashed, answers a hash query for (r, m_i) as $h_i(r)$, and answers a signature query for m_i by feeding i to its tweaked-square-root oracle. The distribution of signatures in this algorithm is identical to the distribution of *fixed* signatures produced by a legitimate signer, so A_5's chance of success is the same as A_6's chance of success against fixed signatures.

In particular, generic attacks on fixed signatures are as difficult as factorization whenever generic existential inversion is as difficult as factorization. The bottom line is that there cannot be any embarrassing generic attacks—attacks better than factorization—against fixed unstructured $B \geq 0$ Rabin-Williams signatures, or against fixed principal $B \geq 1$ Rabin-Williams signatures, or against fixed |principal| $B \geq 1$ Rabin-Williams signatures.

Of course, there are severe limits on the comfort that a user can draw from these tight security proofs. Maybe the attacker doesn't need anything better than factorization: perhaps an implementor chose $K = 512$; perhaps the attacker knows a fast factorization algorithm; perhaps the attacker owns a quantum computer. Even if pq really is difficult to factor, there is no guarantee of security against *non-generic* attacks.

Proofs of this type are nevertheless valuable time-savers for cryptographers. The proofs help focus cryptanalytic effort: cryptanalysts who aren't interested in factorization are guided by the knowledge that they'll have to come up with non-generic attacks. Furthermore, the literature is littered with examples of systems (usually without security proofs, but occasionally with non-tight security proofs) subsequently broken by embarrassing generic attacks. Experience shows that

proof construction and cryptanalysis are linked—a link I have tried to emphasize in my expository structure!—and that authors who attempt to build a proof for each system frequently end up discovering embarrassing attacks that would otherwise have been missed. Weeding out these bad systems saves far more time for the community than could possibly be saved by skipping security proofs.

I would not advocate a general rule of ignoring systems without tight security proofs. Sometimes insisting on tight security proofs is in conflict with choosing the most efficient system. In particular, Schnorr's short-signature system does not have a tight security proof but remains unbroken and is roughly twice as efficient as the closest competitors with tight security proofs—see, e.g., [12, Section 3]. On the other hand, the conflict for systems of RSA/Rabin type seems to have been mostly, if not entirely, eliminated by the proofs in this paper.

References

[1] Victoria Ashby (editor), *First ACM conference on computer and communications security*, Association for Computing Machinery, New York, 1993. See [3].

[2] Rana Barua, Tanja Lange (editors), *Progress in Cryptology—INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11–13, 2006, Proceedings*, Lecture Notes in Computer Science, 4329, Springer, 2006. ISBN 3–540–49767–6. See [19].

[3] Mihir Bellare, Phillip Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, in [1] (1993), 62–73. Citations in this document: §1.

[4] Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin*, in [20] (1996), 399–416; see also newer version [5].

[5] Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin* (1996); see also older version [4]. URL: `http://www-cse.ucsd.edu/~mihir/papers/exactsigs.html`. Citations in this document: §A, §A, §A, §A, §A.

[6] Mihir Bellare (editor), *Advances in cryptology—CRYPTO 2000: proceedings of the 20th Annual International Cryptology Conference held in Santa Barbara, CA, August 20–24, 2000*, Lecture Notes in Computer Science, 1880, Springer-Verlag, Berlin, 2000. ISBN 3–540–67907–3. MR 2002c:94002. See [7].

[7] Jean-Sébastien Coron, *On the exact security of Full Domain Hash*, in [6] (2000), 229–235. MR 2002e:94109. URL: `http://www.eleves.ens.fr/home/coron/publications/publications.html`. Citations in this document: §1, §A, §A.

[8] Jean-Sébastien Coron, *Security proof for partial-domain hash signature schemes*, in [25] (2002), 613–626. URL: `http://www.gemplus.com/smart/r_d/publications/`. Citations in this document: §A.

[9] Jean-Sébastien Coron, *Optimal security proofs for PSS and other signature schemes*, in [13] (2002), 272–287. URL: `http://www.eleves.ens.fr/home/coron/publications/publications.html`. Citations in this document: §1, §A, §A.

[10] Martin Gardner, *A new kind of cipher that would take millions of years to break*, Scientific American (1977), 120–124. Citations in this document: §A, §A.

[11] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal on Computing **17** (1988), 281–308. ISSN 0097–5397. MR 89e:94009. URL: `http://theory.lcs.mit.edu/~rivest/publications.html`. Citations in this document: §7, §A.

[12] Jonathan Katz, Nan Wang, *Efficiency improvements for signature schemes with tight security reductions* (2003). URL: `http://www.cs.umd.edu/~jkatz/papers.html`. Citations in this document: §1, §1, §2, §2, §2, §7, §7, §8, §A, §A, §A, §A.

[13] Lars Knudsen (editor), *Advances in cryptology—EUROCRYPT 2002: proceedings of the 21st International Annual Conference on the Theory and Applications of Cryptographic Techniques held in Amsterdam, April 28–May 2, 2002*, Lecture Notes in Computer Science, 2332, Springer-Verlag, Berlin, 2002. ISBN 3–540–43553–0. See [9].

[14] Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 1st edition, 1st printing, Addison-Wesley, Reading, 1969; see also newer version [15]. MR 44:3531.

[15] Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 1st edition, 2nd printing, Addison-Wesley, Reading, 1971; see also older version [14]; see also newer version [16]. MR 44:3531.

[16] Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 2nd edition, Addison-Wesley, Reading, 1981; see also older version [15]; see also newer version [17]. ISBN 0–201–03822–6. MR 83i:68003. Citations in this document: §A.

[17] Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 3rd edition, Addison-Wesley, Reading, 1997; see also older version [16]. ISBN 0–201–89684–2. Citations in this document: §A.

[18] Neal Koblitz, Alfred J. Menezes, *Another look at "provable security"*, revised 4 May 2005 (2005). URL: `http://eprint.iacr.org/2004/152/`. Citations in this document: §1, §1.

[19] Neal Koblitz, Alfred J. Menezes, *Another look at "provable security". II*, in [2] (2006). URL: `http://eprint.iacr.org/2006/229`. Citations in this document: §1.

[20] Ueli M. Maurer (editor), *Advances in cryptology—EUROCRYPT '96: Proceedings of the Fifteenth International Conference on the Theory and Application of Cryptographic Techniques held in Saragossa, May 12–16, 1996*, Lecture Notes in Computer Science, 1070, Springer-Verlag, Berlin, 1996. ISBN 3–540–61186–X. MR 97g:94002. See [4].

[21] Michael O. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, Technical Report 212, MIT Laboratory for Computer Science, 1979. URL: `http://ncstrl.mit.edu/Dienst/UI/2.0/Describe/ncstrl.mit_lcs/MIT/LCS/TR-212`. Citations in this document: §A, §A, §A, §A, §A.

[22] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), 120–126. ISSN 0001–0782. URL: `http://cr.yp.to/bib/entries.html#1978/rivest`. Citations in this document: §A, §A, §A, §A, §A.

[23] Douglas R. Stinson, *Cryptography: theory and practice*, CRC Press, Boca Raton, Florida, 1995. ISBN 0–8493–8521–0. MR 96k:94015. Citations in this document: §A.

[24] Hugh C. Williams, *A modification of the RSA public-key encryption procedure*, IEEE Transactions on Information Theory **26** (1980), 726–729. ISSN 0018–9448. URL: `http://cr.yp.to/bib/entries.html#1980/williams`. Citations in this document: §A.

[25] Moti Yung (editor), *Advances in cryptology—CRYPTO 2002: 22nd annual international cryptology conference, Santa Barbara, California, USA, August 2002, proceedings*, Lecture Notes in Computer Science, 2442, Springer-Verlag, Berlin, 2002. ISBN 3–540–44050–X. See [8].

A Appendix: the Rabin-Williams public-key signature system

This appendix surveys, and traces the history of, several useful features of the Rabin-Williams public-key signature system. These features motivate practical use of the Rabin-Williams signature system. These features also motivate theoretical study of the security of the system.

The first few features are widely understood, and are discussed here purely to set the history straight. Some features later in the list are much less widely known; readers who think of Rabin-Williams as a minor variant of RSA are specifically encouraged to read the discussions of compression and expansion.

Warning: The Rabin-Williams "signature system" is actually a family of signature systems parametrized by key size, hash function, randomizer length, signature choice, etc. Some of the features depend on particular parameter choices. These dependencies are stated explicitly.

Readers are expected to be familiar with the general idea of modular-root signature systems: a signature is an eth root of something modulo a public key n whose prime factorization is known to the signer. This idea was introduced by Rivest, Shamir, and Adleman in [10] and [22]. This appendix can be viewed as a comparison chart surveying the features of various modular-root signature systems.

Hashing. In the Rabin-Williams system, messages are scrambled by a public hash function H. A signature of m is an eth root of $H(m)$, not an eth root of m. This is essential for security.

History: The signature system proposed by Rivest, Shamir, and Adleman did not hash messages; it was trivially breakable. Specifically, [22, page 122, first display] defines a signature S of a message M under a public key B by "$S = D_B(M)$," never mentioning hashing; [22, page 122, third display] defines D as a power of its input; one trivial attack is to forge the message 1 with signature 1. Rabin's system in [21] did hash messages; it remains unbroken today. The apparent security benefit of hashing was mentioned in [21, page 10, last sentence]: "Actually, this [attack idea] does not seem a serious threat because of the hashing effected by $C(M)$."

Reader beware: Many authors unjustifiably refer to an oversimplified, trivially breakable, non-hashing system as "Rabin's system"; consider, for example, the claim by Goldwasser, Micali, and Rivest in [11, Section 3] that "Rabin's signature scheme is totally breakable if the enemy uses a

directed chosen-message attack." Furthermore, many authors—see, e.g., [23, Section 7.1]—describe hashing as merely a way to handle long messages, rather than as an essential component of the system no matter what the message length might be. Modern treatments of "RSA" usually include hashing but usually fail to give Rabin any credit for the idea.

Small exponent. In the Rabin-Williams system, the exponent of a signature s is a small fixed integer, rather than a large integer chosen randomly by the signer. This makes verification *much* faster. It also saves space in public keys.

History: The system proposed by Rivest, Shamir, and Adleman used large exponents. See [22, page 123, fifth line]: "You then pick the integer d to be a large, random integer which is relatively prime to $(p-1) \star (q-1)$ ... The integer e is ... the 'multiplicative inverse' of d." Rabin in [21, page 5] pointed out the huge speed advantage of small exponents.

Most modern descriptions of "RSA," and the vast majority of "RSA" implementations, include small fixed exponents, typically 3 or 17 or 65537. Usually the descriptions fail to give Rabin any credit for the idea. For example, Knuth in [16, pages 386–389] (and again in [17, pages 403–406]) explained an exponent-3 system and unjustifiably called it "RSA." Many treatments actively miscredit small exponents to [10] and [22]. For example, here's a quote from a well-known algorithms book by Cormen, Leiserson, Rivest, and Stein: "The RSA cryptosystem ... Select a *small* odd integer e ... The RSA cryptosystem was proposed in 1977 by Rivest, Shamir, and Adleman." (Emphasis added.)

Large exponents have inexplicably attracted somewhat more attention than small fixed exponents as the topic of security proofs, even though small exponents are just as easy for theoreticians to handle and much more interesting for practitioners. Bellare and Rogaway in [5] analyzed a traditional system that they called "FDH," and a system of their own design called "PSS," in both cases using large exponents. See [5, Section 2.1]; see also Coron's [7, Section 2.3] and [9, Definition 4]. Katz and Wang were exponent-agnostic in [12]: they stated their results for more general "claw-free permutation pairs." The "PRab" claim in [5, Theorem 6.1] used a small exponent, but the proof was merely outlined; [8, Theorem 4] used a small exponent, but no proof was given.

"Strong RSA" proofs require large exponents, but those systems do not provide fast verification and do not seem to have attracted any practical interest.

Exponent 2. The Rabin-Williams system uses exponent 2 rather than exponent 3. This speeds up verification, and improves the compression and expansion features discussed below. The signer's secret primes p and q are chosen from $3 + 4\mathbf{Z}$ to simplify signing.

I've noticed that some programmers fear exponent 2: there appears to be a widespread belief that fast exponent-2 signing requires Euclid-type computations of Jacobi symbols. For example, the "IEEE Standard Specifications for Public-Key Cryptography" claim that, compared to exponent 2, exponent 3 has a "code-size and speed advantage because there is no need to compute the Jacobi symbol." However, the simple fact is that Euclid-type computations of Jacobi symbols are *not* required. See below for a state-of-the-art algorithm to compute principal Rabin-Williams signatures with the same "code-size and speed advantage"; the information that would be extracted from a Euclid-type Jacobi-symbol computation is trivially extracted from a few multiplications.

Exponent 2 does require some extra effort in security proofs, because a uniform distribution of $s \bmod pq$ does not correspond to a uniform distribution of $s^2 \bmod pq$. The proof strategy here depends on whether signers choose uniform random square roots, or square roots distinguished by being squares, or square roots distinguished by being absolute values of squares.

History: Exponent 2 was introduced by Rabin in [21]. Most writers fail to give credit to Rabin for hashing and small exponents but do give credit to Rabin for exponent 2. I see no reason to use any other exponent; perhaps 2 will eventually become the most popular exponent, and, as a side effect, Rabin will receive more of the recognition that he deserves.

Message randomization: r. The Rabin-Williams system actually computes a square root of $H(r, m)$, not a square root of $H(m)$. Here r is a string selected randomly by the signer. The number of bits of r is a system parameter B. This randomization was introduced in [21]: Rabin suggested $B = 60$, with a random choice of r.

One can see, in the literature, five different strategies to choose the parameter B:

- Choose $B = 0$. This means that r is empty and that the H input is not actually randomized. The main argument for this choice is that any larger B means extra effort to generate r, extra effort to include r in the H input, and extra effort to transmit r along with s.
- Choose $B = 1$. The main argument for this choice is that a nonzero B is required for the type of tight security proof introduced by Katz and Wang in [12]. The conventional wisdom is that $B = 0$ does not allow a tight security proof; see the "FDH" analyses in [5] and [7]. On the other hand, this paper challenges the conventional wisdom; this paper's new proof idea shows tight security for "fixed unstructured" signatures even in the case $B = 0$.
- Choose $B = 8$. As a historical matter, Rabin's system was able to produce signatures for only about 1/4 of all choices of r (since only a small fraction of all integers mod pq are squares), and Rabin suggested trying again if the first r failed; having 256 choices of r means that all choices will fail with probability about 2^{-106}. However, the Rabin-Williams system eliminates these failures, as discussed below. The only remaining argument for $B = 8$ is that it marginally improves the tightness of the Katz-Wang approach.
- Choose B large enough to prevent the attacker from guessing r in advance; for example, $B = 128$. This choice allows a different type of tight security proof that is not considered in this paper and that seems to have been rendered obsolete by the idea of "fixed" signatures.
- Choose B large enough to prevent all collisions in r: for example, $B = 256$. This choice allows an older type of tight security proof that seems to have been obsolete for many years.

My impression is that, in practice, the choice $B = 0$ is by far the most popular choice, although I haven't done a scientific study.

One might wonder, from the above description, why large choices of B would attract any interest, and in particular why Rabin chose $B = 60$ rather than $B = 8$. The answer is that large choices of B are often conjectured to make non-generic attacks, attacks that pay attention to the hash function H, more difficult. For example, MD5-based signatures have been broken for $B = 0$ but not for $B = 128$. Does a larger B allow us to choose a smaller, faster hash function H? Could this compensate for the direct costs of a longer r? Resolving these questions means understanding the cost-security tradeoff for hash functions, obviously not an easy task. For the moment I recommend that theoreticians remain agnostic and continue to investigate all possible B's.

Reader beware: Many authors have failed to give Rabin proper credit for his randomized signatures (r, s). For example, Goldwasser and Bellare have posted lecture notes (1) claiming that Rabin introduced a signature system with neither H nor r; (2) assigning credit to a 1983 paper of Goldwasser, Micali, and Yao for "pioneering" randomized signatures; and then (3) describing

a "PSS0" system—randomized (r, s)—as if it were new. Similar comments apply to the "PFDH" system in [9] and [12].

The tweaks e and f. Recall that Rabin's system needed to try several values of r, on average about 4 values, before finding a square $H(r, m)$ modulo pq. The Rabin-Williams system eliminates this problem by using *tweaked* square roots in place of square roots. A tweaked square root of h modulo pq is a vector (e, f, s) such that $e \in \{-1, 1\}$, $f \in \{1, 2\}$, and $efs^2 - h \in pq\mathbf{Z}$; the signer's secret primes p and q are chosen from $3 + 8\mathbf{Z}$ and $7 + 8\mathbf{Z}$ respectively. Each h has exactly four tweaked square roots, so each choice of r works, speeding up signatures.

Here is a straightforward high-speed fault-tolerant algorithm to compute the "principal" tweaked square root (e, f, s) of h modulo pq:

1. Compute $U \leftarrow h^{(q+1)/8} \bmod q$.
2. If $U^4 - h \bmod q = 0$, set $e = 1$; otherwise set $e = -1$.
3. Compute $V \leftarrow (eh)^{(p-3)/8} \bmod p$.
4. If $(V^4(eh)^2 - eh) \bmod p = 0$, set $f = 1$; otherwise set $f = 2$.
5. Compute $W \leftarrow f^{(3q-5)/8}U \bmod q$. (The power $2^{(3q-5)/8} \bmod q$ can be precomputed.)
6. Compute $X \leftarrow f^{(9p-11)/8}V^3eh \bmod p$. (The power $2^{(9p-11)/8} \bmod p$ can be precomputed.)
7. Compute $Y \leftarrow W + q(q^{p-2}(X - W) \bmod p)$. (The power $q^{p-2} \bmod p$ can be precomputed.)
8. Compute $s \leftarrow Y^2 \bmod pq$.
9. At this point $efs^2 \bmod pq = h$; check this explicitly and restart in case of a fault.

The bottlenecks in this algorithm are one exponentiation modulo p in the first step and one exponentiation modulo q in the third step.

History: The tweaks e and f were introduced by Williams in [24]. I posted an example of a no-explicit-Jacobi-symbol signing algorithm in October 2000.

Message recovery. Hash functions H can be, and often are, designed to allow "message recovery." This means that a fixed-length prefix of (r, m) can be computed from $H(r, m)$. See [5, Section 5] and [12, Section 4.2] for generically safe methods to construct H from another hash function.

The advantage of "message recovery" is that it allows compression of signed messages: one simply omits the fixed-length prefix from (r, m, s). The verifier sees s, computes the alleged $H(r, m)$ by computing $s^2 \bmod pq$, recovers the prefix of (r, m), and then checks that $H(r, m)$ matches.

Message recovery is often viewed as an argument for choosing a large B, such as $B = 128$. The argument is as follows: "Message recovery eliminates the space required by r. The space required by r was the only disadvantage of a large B. Maybe a large B stops attacks." There are several problems with this argument. First, bandwidth is only part of the picture: for example, a larger B means a larger cost to generate r. Second, signed messages are usually uncompressed; one important reason is that uncompressed signatures (and expanded signatures, as discussed below) allow faster verification. Third, except when (r, m) is extremely short, the alleged savings is a myth. Adding 128 bits to r means pushing 128 bits of m out of the compressed prefix of (r, m), and therefore expanding signed messages by 128 bits.

Signature compression. Another way to save space—more effective than message recovery when (r, m) is short—is to transmit all of (r, m) but transmit only the top half of the bits of s. The receiver can use Coppersmith's algorithm to find a square root of $H(r, m)$ modulo pq given the top

half of the bits of the square root. Bleichenbacher proposed a better approach, allowing the same amount of compression with much faster decompression and verification: s is transmitted as the largest under-half-size denominator in the continued fraction for fs/pq.

A similar compression method applies to exponent-3 "RSA" but saves only 1/3 of the signature bits rather than 1/2.

Key compression. Yet another way to save space is to compress public keys pq. It is widely known that RSA/Rabin keys can be compressed to 1/2 size. It is not so widely known that Coppersmith found a method to compress keys to 1/3 size. It is also not widely known that this compression—done properly!—can be proven to preserve security, not just against generic attacks but against all attacks. The critical point is that generating one key p_1q_1 in the conventional way, and then generating another key pq that shares the top 1/2 (or 2/3) of the bits of p_1q_1, produces exactly the same distribution of pq that would have been produced by generating pq in the conventional way.

Key compression has the same benefits for higher-exponent "RSA," so it is orthogonal to a comparison of Rabin-Williams with "RSA." It is, however, relevant to a a comparison of Rabin-Williams with signature systems of ElGamal/Schnorr/ECDSA type. For example, a 1024-bit Rabin-Williams signature can be compressed to a 512-bit signature, and a 1024-bit key can be compressed to a 352-bit key. A typical ECDSA variant at the same conjectured security level has a smaller signature (320 bits) and a smaller key (160 bits) but has much slower verification; for many applications, the slowdown in verification outweighs the 192-bit savings in signature length.

Bleichenbacher pointed out a way to further compress keys pq—all the way down to 0 bits!—inside vectors (pq, e, f, s, r, m). The idea is to recover pq as a divisor of $efs^2 - H(r, m)$. The standard compression method (or, as an alternative, Coppersmith's compression method) already reveals the top 1/2 (or 2/3) of the bits of the divisor, and the remaining bits are easily (or very easily) found by lattice-basis-reduction techniques.

Signature expansion. The Rabin-Williams system offers another attractive option for applications where verification speed is much more important than signature length: signatures can be expanded for faster verification. Specifically, the signature (e, f, r, s) satisfying $efs^2 \equiv H(r, m) \pmod{pq}$ can be converted into an expanded signature (e, f, r, s, t) satisfying $efs^2 - pqt = H(r, m)$. The verifier can efficiently check the latter equation modulo a random 128-bit prime (or several smaller primes) with negligible chance of error. The verifier can amortize the prime-generation cost across any number of signatures by keeping the prime secret and reusing it.

A similar idea applies to exponent-3 "RSA" but requires a double-length t, considerably slowing down verification.

History: I posted this expansion idea to `sci.crypt` in March 1997.