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

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.

	B, number of bits of randomization of hash input		
	B large	$B = 1$	$B = 0$
Variable unstructured Rabin–Williams	tight security: '96 Bellare–Rogaway	no security: easy attack	no security: easy attack
Variable principal Rabin–Williams	tight security: this paper	loose security: this paper	loose security: this paper
Variable RSA	tight security: '96 Bellare–Rogaway	loose security: '93 Bellare–Rogaway	loose security: '93 Bellare–Rogaway
Fixed RSA	tight security: '96 Bellare–Rogaway	tight security: '03 Katz–Wang	loose security: '93 Bellare–Rogaway
Fixed principal Rabin–Williams	tight security: this paper	tight security: this paper	loose security: this paper
Fixed unstructured Rabin–Williams	tight security: '96 Bellare–Rogaway	tight security: this paper	tight security: this paper

Table 0.1. Proven lower bounds on "security in the random-oracle model" relative to roots (for RSA) or factorization (for Rabin–Williams).

[*] Permanent ID of this document: `c30057d690a8fb42af6a5172b5da9006`. Date of this document: 2008.01.30. This paper incorporates and supersedes various postings by the author between 2000 and 2007. This work was supported by the National Science Foundation under grants CCR–9983950, DMS–0140542, and ITR–0716498, and by the Alfred P. Sloan Foundation.

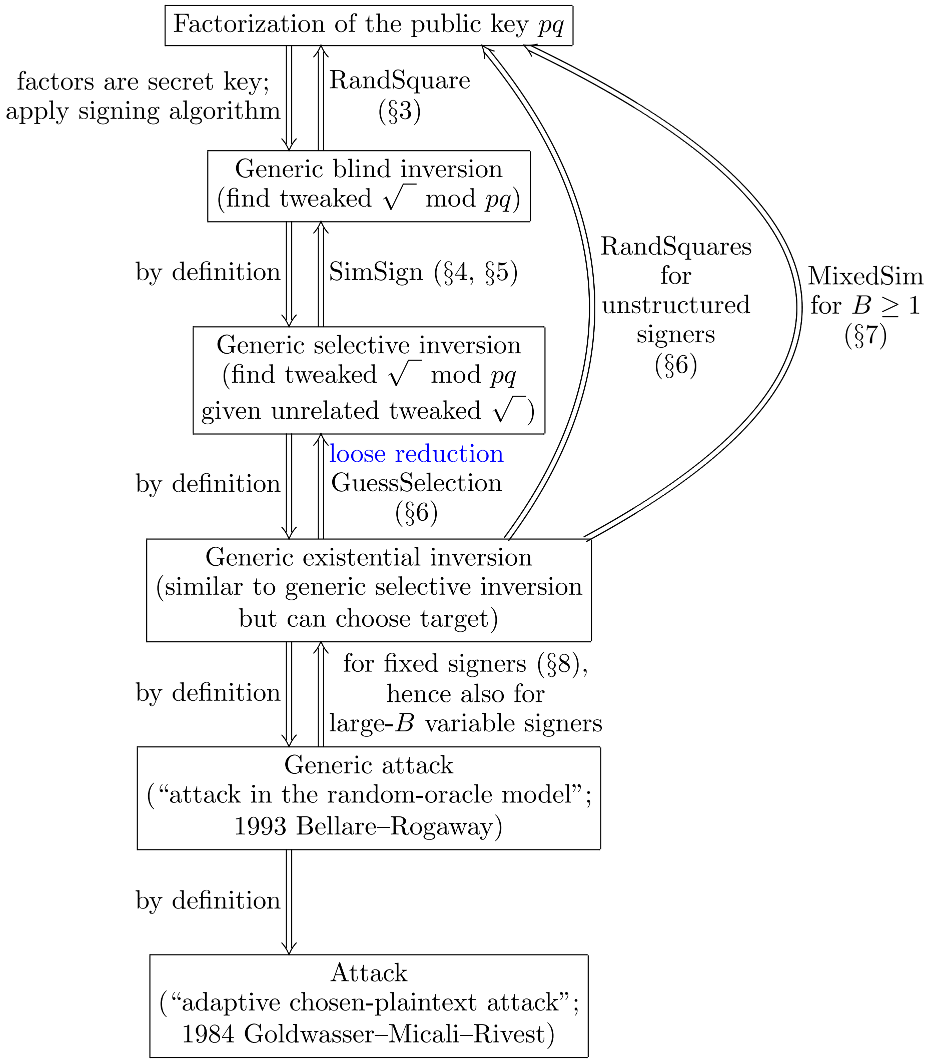
Factorization of the public key pq
factors are secret key;
apply signing algorithm
RandSquare
(§3)
Generic blind inversion
(find tweaked $\sqrt{\ }$ mod pq)
by definition
SimSign (§4, §5)
RandSquares
for
unstructured
signers
(§6)
MixedSim
for $B \geq 1$
(§7)
Generic selective inversion
(find tweaked $\sqrt{\ }$ mod pq
given unrelated tweaked $\sqrt{\ }$)
by definition
loose reduction
GuessSelection
(§6)
Generic existential inversion
(similar to generic selective inversion
but can choose target)
by definition
for fixed signers (§8),
hence also for
large-B variable signers
Generic attack
("attack in the random-oracle model";
1993 Bellare–Rogaway)
by definition
Attack
("adaptive chosen-plaintext attack";
1984 Goldwasser–Micali–Rivest)

Fig. 0.2. Proven reductions among various types of attacks against Rabin–Williams signatures. SimSign is more difficult for Rabin–Williams than for RSA; the unstructured case was outlined in 1996 Bellare–Rogaway, but the principal case was specifically prohibited in 1996 Bellare–Rogaway and requires extra work performed in this paper. GuessSelection is standard but not tight. MixedSim is tight; it combines the new simulator with the central idea of 2003 Katz–Wang. RandSquares is also new in this paper, and also tight; it can be viewed as the result of eliminating guesses from RandSquare(SimSign(GuessSelection))), or as the result of eliminating aborts from an overgeneralization of MixedSim.

“K”	number of key bits; $0 < pq - 2^K < 2^K$
“D”	distribution of secret keys (p, q)
“H”	the hash function
“B”	number of bits of randomization of hash input
“α”	“unstructured”: signer chooses uniform random tweaked $\sqrt{\ }$; “principal”: signer finds unique tweaked $\sqrt{\ }$ that is a square etc.; “|principal|”: if $\sqrt{\ }$ is between $(pq+1)/2$ and $pq-1$ then negate it
“β”	“variable”: signer generates new random bits for each signature; “fixed”: signer repeats signature if message is repeated

Table 0.3. Summary of security-relevant parameters in the Rabin–Williams signature system. See Section 2 for definitions.

1 Introduction

Variants of the Rabin–Williams public-key signature system have, since 1980, 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?[1] This is not an idle concern: some RSA-type systems have been broken by devastating attacks that (1) are much faster than known methods to compute roots modulo pq and (2) work for a large fraction of all functions H, given oracle access to H.

Some systems have been proven immune to such attacks. For example, in the 1993 paper [5] 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 devastation, 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

[1] Notes on terminology: Twenty years ago, in [14, Section 2.2], Goldwasser, Micali, and Rivest defined various types of “attacks” against signature systems—in particular, “adaptive chosen-message attacks,” the “most severe natural attack an enemy can mount.” The definition has been repeated countless times in the literature, and the reader is assumed to be familiar with it. This paper follows common practice in abbreviating “adaptive chosen-message attack” as simply “attack.”

This paper follows [5] in focusing on attacks that work for (a significant fraction of) all functions H, given access to an oracle computing H. In [5] these attacks are called attacks “in the random-oracle model.” This paper follows the more concise terminology of [9, Section 7.1], [22, Section 1.1], [23, Section 4], et al.: these attacks are “H-generic attacks,” or simply “generic attacks.”

hash values seen by the FDH attack. Coron in [10] 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 an 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 [11]. A 2003 theorem of Katz and Wang 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. 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 Rabin–Williams 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" signers and does not cover "principal" or "|principal|" signers. The specific case of "fixed unstructured $B = 0$" Rabin–Williams is nevertheless worth studying 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.

These proofs owe a heavy debt to the efforts of Koblitz and Menezes in [17] and [18] to clarify the limits of "provable security." In particular, in [17, 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 for RSA (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. Thanks to Dan Boneh for pointing out [15] to me shortly after it was posted. Thanks to Tanja Lange and the anonymous referees for many suggestions regarding exposition.

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, a tweaked-square-root distribution α, either "unstructured" or "principal" or "|principal|"; second, a signature-repetition parameter β, either "fixed" or "variable." All of these parameters are explained in detail below.

Readers wondering "Why are you analyzing these specific systems?" should read the detailed cost analysis and historical survey in [8]. The short answer is that, among all the 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.) This engineering has not produced the world's simplest family of signature systems—this section needs two pages to state all the details of what the signer and verifier do—but the loss in simplicity is justified by the reduction in cost.

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

For each algorithm A define $\mathrm{PrFactor}(A)$ as the probability that $A(pq) \in \{p, q\}$, when (p, q) is chosen randomly from the distribution D. This probability depends explicitly on A and implicitly on the parameters (K, D). No security is possible when K and D 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 very little 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. 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 $H : \{B\text{-bit strings}\} \times \{\text{messages}\} \to \left\{1, 2, \ldots, 2^K\right\}$. For example, for $B = 0$ and $K = 2048$, the function H assigns an element of $\left\{1, 2, \ldots, 2^{2048}\right\}$ 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 $H(r, m)$.

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 signers, principal signers, |principal| signers. 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 $B \geq 1$), 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 $h = H(r, m)$. 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.
- |**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 6—is affected much more dramatically by the choice.

This paper is not the first paper to point out the importance of the signature distribution for Rabin–Williams security proofs. For example, Bellare and Rogaway in [7, Section 6] wrote "SignPRab . . . returns a random square root . . . We stress that a random root is chosen; a fixed root won't do." In my terminology, Bellare and Rogaway are requiring unstructured signers and prohibiting principal signers, |principal| signers, etc. Sometimes principal signers require extra work for a security proof (work done in Section 4 of this paper); sometimes they don't seem to allow a security proof at all.

Variable signers, fixed signers. 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.

The importance of this choice for security proofs was first pointed out by Katz and Wang in [15]. The conventional wisdom before [15] was that tight security proofs required a large B; Katz and Wang proved tight security for various types of fixed signers with $B = 1$. As a more extreme illustration of the importance of this choice, consider the fact that "fixed unstructured $B = 0$" Rabin–Williams now has a tight security proof, whereas "variable unstructured $B = 0$" Rabin–Williams is easily breakable.

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

3 Generic blind inversion

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, D. For each algorithm A define PrInvBlind(A) as the probability that $A(pq, h')$ is some $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ such that $e'f'(s')^2 \equiv h' \pmod{pq}$, when

- (p, q) is a D-distributed random secret key,
- h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$,

and (p, q) is independent of h'. How large can PrInvBlind(A) be, as a function of the resources consumed by A?

Any fast probability-1 algorithm A for this generic-blind-inversion problem immediately implies a fast probability-1 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 uses A to compute 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 fast high-probability algorithm A for this problem immediately implies a fast high-probability factorization algorithm RandSquare(A). The proof is completely standard, *except* for the details of how the tweaks e, f are handled; readers are encouraged to read the proof as a warmup for the security proofs in subsequent sections.

Here is the factorization algorithm RandSquare(A):

```
0. Input $n$.
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. Go back to step 1 if $h' \notin \{1, 2, \ldots, 2^K\}$.
4. Compute $(e', f', s') = A(n, h')$.
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 PrInvBlind(A) implies a similarly large factorization chance PrFactor(RandSquare(A)). The time of RandSquare(A) is practically identical to the time of A: the difference is a few easy operations modulo n to generate h, repeated only $n/2^K < 2$ times on average, plus a few gcd operations.

Theorem 3.1. PrFactor(RandSquare(A)) $\geq (1/2)$ PrInvBlind(A).

Proof. Let (p, q) be a D-distributed random secret key. The quantity $h' = efs^2 \bmod pq$ in step 4 of (RandSquare(A))(pq) 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 $e'f'(s')^2 \equiv h' \pmod{pq}$ occurs with probability exactly PrInvBlind(A). 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 pq.

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

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

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

Case 4: $\gcd\{h', pq\} = 1$. I claim that $(s')^2 \equiv s^2 \pmod{pq}$. Notice first that $e'f'(s')^2 \equiv efs^2 \pmod{pq}$, and recall that p, q are primes 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 $f'(s')^2 \equiv fs^2 \pmod{pq}$. 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 $(s')^2 \equiv s^2 \pmod{pq}$.

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, pq divides $(s')^2 - s^2$ without dividing $s' - s$ or $s' + s$, so $\gcd\{n, s' - s\}$ is a nontrivial factor of pq. □

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 $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Also fix K and D. For each algorithm A define $\text{PrInvSelective}_1(A)$ as the probability that $A(pq, h, e, f, s, h')$ is some $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ such that $e'f'(s')^2 \equiv h' \pmod{pq}$, when

- (p, q) is a D-distributed random secret key,
- h is a uniform random element of $\{1, 2, \ldots, 2^K\}$,
- (e, f, s) is an α-distributed random tweaked square root of h mod pq,
- h' is a uniform random element of $\{1, 2, \ldots, 2^K\}$,

and all of these choices are independent. How large can $\text{PrInvSelective}_1(A)$ be, as a function of the resources consumed by A?

This generic-selective-inversion problem is a natural step for the cryptanalyst beyond the generic-blind-inversion problem in Section 3. Any fast probability-1 algorithm A to solve this problem immediately implies a fast probability-1 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 $(e', f', s') = A(pq, h, e, f, s, h')$, 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 fast high-probability algorithm A for this problem immediately implies a fast high-probability algorithm $\text{SimSign}_1(A)$ for the generic-blind-inversion problem, and therefore implies a fast high-probability factorization algorithm $\text{RandSquare}(\text{SimSign}_1(A))$.

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

There are three different constructions of the simulator, and thus three different constructions of $\text{SimSign}_1(A)$, one for each of the three choices of α. Here is $\text{SimSign}_1(A)$ for the simplest choice, $\alpha = \text{unstructured}$:

```
0. Input $n$ and $h'$.
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. Go back to step 1 if $h \notin \{1, 2, \ldots, 2^K\}$.
4. Print $A(n, h, e, f, s, h')$.
```

Here is $\text{SimSign}_1(A)$ for $\alpha \in \{\text{principal}, |\text{principal}|\}$:

```
0. Input $n$ and $h'$.
1. Generate a uniform random $(e', f', x) \in \{-1, 1\} \times \{1, 2\} \times \{0, \ldots, n-1\}$.
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. Go back to step 1 if $h \notin \{1, 2, \ldots, 2^K\}$.
8. Print $A(n, h, e, f, s, h')$ if $\alpha = \text{principal}$, else $A(n, h, e, f, \min\{s, n-s\}, h')$.
```

The following theorem states that a large success chance $\text{PrInvSelective}_1(A)$ implies a large success chance $\text{PrInvBlind}(\text{SimSign}_1(A))$. The time of $\text{SimSign}_1(A)$ is practically identical to the time of A: 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. $\text{PrInvBlind}(\text{SimSign}_1(A)) = \text{PrInvSelective}_1(A)$.

The reader may have noticed that my constructions of $\text{SimSign}_1(A)$, 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. Let (p, q) be a D-distributed random secret key. Write $n = pq$. Let h' be a uniform random element of $\{1, 2, \ldots, 2^K\}$, independent of (p, q). Consider $(\text{SimSign}_1(A))(n, h')$.

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 $\text{PrInvSelective}_1(A)$.

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 $f = 2$ then $eh \equiv 2s^2$, 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). In particular, each integer $h \in \left\{1, 2, \ldots, 2^K\right\}$ is produced by exactly four choices of (e', f', x).

|Principal|: h is uniform exactly as above, 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 $h_1, h_2, \ldots, h_Q, h' \in \left\{1, 2, \ldots, 2^K\right\}$, 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 $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Fix K and D. Fix $Q \geq 0$. For each algorithm A define $\text{PrInvSelective}_Q(A)$ as the chance that $A(pq, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$ is some $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq-1\}$ satisfying $e'f'(s')^2 \equiv h' \pmod{pq}$, when

- (p, q) is a D-distributed random secret key,
- each h_i is a uniform random element of $\left\{1, 2, \ldots, 2^K\right\}$,
- (e_i, f_i, s_i) is an α-distributed random tweaked square root of $h_i \bmod pq$,
- h' is a uniform random element of $\left\{1, 2, \ldots, 2^K\right\}$,

and all of these choices are independent. How large can $\text{PrInvSelective}_Q(A)$ be, as a function of the resources consumed by A?

The answer is that this problem is provably as difficult as factorization of public keys. The construction of SimSign_Q is an easy generalization of last section's construction of SimSign_1. For example, here is $\text{SimSign}_Q(A)$ for $\alpha =$ unstructured:

```
0. Input $n$ and $h'$.
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 $A(n, h_1, e_1, f_1, s_1, \ldots, h_Q, e_Q, f_Q, s_Q, h')$.
```

The remaining constructions work similarly.

Theorem 5.1. $\text{PrInvBlind}(\text{SimSign}_Q(A)) = \text{PrInvSelective}_Q(A)$.

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?

More formally: Fix $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Fix K and D. Fix $Q \geq 0$. For each algorithm A define $\text{PrInvExistential}_Q(A)$ as follows. A is given pq where (p, q) is a D-distributed random secret key, and uniform random elements $h_1, h_2, \ldots, h_{Q+1}$ of $\{1, 2, \ldots, 2^K\}$, all of these choices being independent. A makes Q distinct oracle queries i; in response to each i, A is given an α-distributed random tweaked square root (e_i, f_i, s_i) of h_i modulo pq, again independently of other choices. Now $\text{PrInvExistential}_Q(A)$ is the probability that A outputs some $(i, e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, \ldots, pq - 1\}$ such that $e' f' (s')^2 \equiv h_i \pmod{pq}$ and such that i was not one of the oracle queries.

The big difference between this generic-existential-inversion 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 or generic blind inversion; in fact, the proof jumps directly to the factorization problem. I don't know any way to get from a generic-existential-inversion algorithm to a generic-blind-inversion algorithm, in the case $B = 0$, except via factorization.

The new idea. Let's start by reviewing the standard proof that the gain is at most a factor $Q + 1$. Given a generic-existential-inversion algorithm A, build a generic-selective-inversion algorithm $\text{GuessSelection}(A)$ 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 $A(n, h_1, \ldots, h_{Q+1})$, using e_i, f_i, s_i to answer query i from A; abort if A selects $i = \pi$ for a query rather than for output.

The choice of π is independent of the operation of A before an abort occurs, so this algorithm GuessSelection(A) aborts with probability exactly $Q/(Q+1)$. If GuessSelection(A) does not abort then it runs A with exactly the right input distribution.

This construction is the heart of the 1993 Bellare–Rogaway loose security proof. The random choice of π in GuessSelection(A) is a guess for the index i that A 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 feed this generic-selective-inversion algorithm GuessSelection(A) to the reductions in previous sections. Section 5 produces a generic-blind-forgery algorithm SimSign(GuessSelection(A)): each input h_i is replaced by an output from the appropriate simulator. Section 3 then produces a factorization algorithm RandSquare(SimSign(GuessSelection(A))): the input h' is replaced by a random efs^2, so that a tweaked square root of h' reveals a factorization of pq.

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 exploit unstructured signatures by treating all of the inputs $h_1, \ldots, h_{Q+1}$ 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 GuessSelection(A).

Here is the new, direct, almost ludicrously simple construction of a factorization algorithm RandSquares(A) from a generic-existential-inversion algorithm A for $\alpha =$ unstructured:

```
0. Input $n$.
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 $(j, e', f', s') = A(n, h_1, \ldots, h_{Q+1})$, using $(e_i, f_i, s_i)$ to answer query
   $i$ from $A$. 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 RandSquares(A) is the time for A plus the time for the final gcd computations and, on average, the time for $(Q+1)n/2^K < 2(Q+1)$ generations of h_i.

Theorem 6.1. *PrFactor(RandSquares(A)) $\geq$ (1/2) PrInvExistential(A) if $\alpha =$ unstructured.*

Proof. Let (p, q) be a D-distributed random secret key. By construction the quantities $h_1, \ldots, h_{Q+1}$ inside $(\mathrm{RandSquares}(A))(pq)$ are independent uniform random elements of $\{1, 2, \ldots, 2^K\}$, so the event $e'f'(s')^2 \equiv h_j \pmod{pq}$ occurs with probability exactly $\mathrm{PrInvExistential}(A)$. Given this event, one of $s', s' - s_j$ has a nontrivial factor in common with pq with conditional probability at least $1/2$, 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 (via an oracle) random access to $h_1(0), \ldots, h_1(2^B - 1), h_2(0), \ldots, h_2(2^B - 1), \ldots, h_{Q+1}(0), \ldots, h_{Q+1}(2^B - 1)$. 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 the tweaked-square-root distribution $\alpha \in \{\text{unstructured}, \text{principal}, |\text{principal}|\}$. Section 6 discussed $\alpha = \text{unstructured}$, and gave a tight security proof for unstructured signers for $B = 0$; this proof generalizes immediately to a tight security proof for unstructured signers 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 $\mathrm{MixedSim}(A)$ out of a generic-existential-inversion algorithm A. This algorithm $\mathrm{MixedSim}(A)$, 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 $r \neq r_i$;
- runs A, 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 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.

How powerful are claw-free permutation pairs? 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 [15, Section 4.1, Theorem 2].

The Katz–Wang theorem is stated for all "claw-free permutation pairs," following [14] and a suggestion of Dodis and Reyzin. One could directly apply the Katz–Wang theorem to the exponent-2 claw-free permutation pair defined by Goldwasser, Micali, and Rivest in [14, Section 6.3], obtaining a tight security proof for an alternate system that at first glance appears quite similar to Rabin–Williams. Unfortunately, a closer look shows that verification in the alternate system is even slower than verification of exponent-3 RSA signatures. This alternate signature system is therefore of much less practical interest than the Rabin–Williams system.

Specifically, [14, Section 6.3] considers the permutations $x \mapsto \left|x^2 \bmod pq\right|$ and $x \mapsto \left|4x^2 \bmod pq\right|$ of the set of positive integers having Jacobi symbol 1 modulo pq. (Absolute values and "positive" here refer to integers between 1 and $(pq-1)/2$.) One can hash to this set by luck (if B is not very small), as Rabin did, or by tweaks, as Williams did. The verifier then has to check that s is a preimage of $H(m)$ under the first permutation: i.e., that s is positive, that s has Jacobi symbol 1, and that $\left|s^2 \bmod pq\right| = H(m)$. Unfortunately, the Jacobi-symbol computation takes much more time than squaring modulo pq.

Dropping the Jacobi-symbol requirement—in other words, switching back to Rabin–Williams signatures—speeds up verification but moves outside the world of permutation pairs; the wider range of accepted inputs means that the verifier's squaring map is no longer a permutation. One can 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. This is exactly where my simulators involved extra work.

Can the same idea be pushed to 0 bits of hash randomization? For $B = 0$, the MixedSim construction accomplishes nothing. It never uses the unstructured simulator; it 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 $s' - s_j$ will have a factor in common with n.

However, 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 $B = 0$.

Tight security for principal $B = 0$ Rabin–Williams remains an open question. Switching from unstructured $B = 0$ signers to principal $B = 0$ signers breaks all of my tight security reductions, and presumably breaks any tight black-box reduction. A tight black-box reduction for principal $B = 0$ Rabin–Williams was claimed in [20, Section 6, Theorem 1], but [20, Section 6, Theorem 1, Proof, equality between "Pr(F successes)" and "$\epsilon(k)$"] implicitly assumes that attackers cannot distinguish principal square roots from arbitrary integers modulo pq.

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 $B=0$ signers, this attack succeeds with probability $\geq 1/2$, for all functions H: notice that $r_1 = r_2$ since $B = 0$, and therefore that $e_1f_1s_1^2 \equiv e_2f_2s_2^2$; continue as in Theorem 3.1. Fortunately, this attack does not work for fixed signers, or for principal or |principal| signers, or for signatures with large B.
- MD5 collisions: The attacker finds distinct messages m, m' with $\mathrm{MD5}(m) = \mathrm{MD5}(m')$. 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.

Consider the class of "H-generic attacks" that work for all (or a significant fraction of all) functions H, given oracle access to H. This class includes many of the attacks in the literature, although there are also many exceptions; it does not include the MD5-collisions attack, for example, but it does include the factorization attack and the signing-leak attack.

How powerful are H-generic attacks against the Rabin–Williams system? Can they be better than factorization? Define $\mathrm{PrAttack}(A)$ as the average, over all functions H, of the success probability of A using an oracle for H. Can $\mathrm{PrAttack}(A)$ be much larger than the other probabilities considered in this paper, as a function of the resources consumed by A?

The signing-leak example shows that these attacks can be quite successful: variable unstructured $B = 0$ signers are broken by an extremely fast generic attack. But the picture is different for fixed signers. For fixed signers, generic attacks that see hash values of $Q+1$ distinct messages are as difficult as Q-query generic existential inversion. Given a generic-attack algorithm A, build a generic-existential-inversion algorithm $\mathrm{FixSignatures}(A)$ as follows: $\mathrm{FixSignatures}(A)$ runs A, 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 signatures produced by a legitimate *fixed* signer, so $\mathrm{FixSignatures}(A)$'s chance of success is the same as A's chance of success against fixed signers.

This FixSignatures construction is weaved through the Katz–Wang reduction "generic attack for fixed $B = 1$ RSA $\Longrightarrow$ blind RSA inversion," and can similarly be weaved through separate proofs of "generic attack for fixed unstructured $B \geq 0$ Rabin–Williams $\Longrightarrow$ factorization" and "generic attack for fixed principal $B \geq 1$ Rabin–Williams $\Longrightarrow$ factorization" and so on; but this means repeating the same construction as part of every reduction. I learned the general principle "generic attack for fixed $\Longrightarrow$ generic existential inversion" from the illustrative "RSA1" and "RSA2" examples given by Koblitz and Menezes in [17, Sections 3.2 and 3.4].

In particular, generic attacks against fixed signers are as difficult as factorization whenever generic existential inversion is as difficult as factorization. Note also that variable signers are indistinguishable from fixed signers *if* B is large. The bottom line is that there cannot be any generic attacks better than factorization against fixed unstructured $B \geq 0$ Rabin–Williams, or against fixed principal $B \geq 1$ Rabin–Williams, or against fixed |principal| $B \geq 1$ Rabin–Williams, or against variable large-B Rabin–Williams.

References

1. Victoria Ashby (editor), *First ACM conference on computer and communications security*, Association for Computing Machinery, New York, 1993. See [5].
2. Vijay Atluri (program chair), Trent Jaeger (program chair), *Proceedings of the 10th ACM conference on Computer and communications security*, ACM Press, 2003. ISBN 1–58113–738–9. See [15].
3. 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 [18].
4. 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 [10].
5. Mihir Bellare, Phillip Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, in [1] (1993), 62–73. Citations in this document: §1, §1, §1.
6. Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin*, in [21] (1996), 399–416; see also newer version [7].
7. Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin* (1996); see also older version [6]. URL: `http://www-cse.ucsd.edu/~mihir/papers/exactsigs.html`. Citations in this document: §2.
8. Daniel J. Bernstein, *RSA signatures and Rabin–Williams signatures: the state of the art* (2008). URL: `http://cr.yp.to/papers.html#rwsota`. Citations in this document: §2.
9. Simon Blake-Wilson, Don Johnson, Alfred Menezes, *Key agreement protocols and their security analysis*, in [12] (1997), 30–45. Citations in this document: §1.
10. Jean-Sébastien Coron, *On the exact security of Full Domain Hash*, in [4] (2000), 229–235. MR 2002e:94109. URL: `http://www.eleves.ens.fr/home/coron/publications/publications.html`. Citations in this document: §1.

11. Jean-Sébastien Coron, *Optimal security proofs for PSS and other signature schemes*, in [16] (2002), 272–287. URL: `http://www.eleves.ens.fr/home/coron/publications/publications.html`. Citations in this document: §1.
12. Michael Darnell (editor), *Cryptography and coding: proceedings of the 6th IMA International Conference held at the Royal Agricultural College, Cirencester, December 17–19, 1997*, Lecture Notes in Computer Science, 1355, Springer-Verlag, 1997. ISBN 3–540–63927–6. MR 99g:94019. See [9].
13. Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, Nan Wang, *Efficient signature schemes with tight reductions to the Diffie-Hellman problems*, Journal of Cryptology **20** (2007), 493–514. See [15].
14. 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: §1, §7, §7, §7.
15. Jonathan Katz, Nan Wang, *Efficiency improvements for signature schemes with tight security reductions*, in [2] (2003), 155–164; portions incorporated into [13]. URL: `http://www.cs.umd.edu/~jkatz/papers.html`. Citations in this document: §1, §2, §2, §7.
16. 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 [11].
17. Neal Koblitz, Alfred J. Menezes, *Another look at "provable security"*, revised 4 May 2005 (2005); see also newer version [19]. URL: `http://eprint.iacr.org/2004/152/`. Citations in this document: §1, §1, §8.
18. Neal Koblitz, Alfred J. Menezes, *Another look at "provable security". II*, in [3] (2006), 148–175. URL: `http://eprint.iacr.org/2006/229`. Citations in this document: §1.
19. Neal Koblitz, Alfred J. Menezes, *Another look at "provable security"*, Journal of Cryptology **20** (2007), 3–37; see also older version [17]. ISSN 0933–2790.
20. Kaoru Kurosawa, Wakaha Ogata, *Efficient Rabin-type digital signature scheme*, Designs, Codes and Cryptography **16** (1999), 53–64. ISSN 0925–1022. Citations in this document: §7, §7.
21. 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 [6].
22. Douglas R. Stinson, *Some observations on the theory of cryptographic hash functions* (2001). URL: `http://eprint.iacr.org/2001/020`. Citations in this document: §1.
23. Douglas R. Stinson, *A polemic on notions of cryptographic security* (2004). URL: `http://www.cacr.math.uwaterloo.ca/~dstinson/pubs.html`. Citations in this document: §1.
24. 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.