Des. Codes Cryptogr. (2015) 77:587–610
DOI 10.1007/s10623-015-0094-2

The random oracle model: a twenty-year retrospective

Neal Koblitz[1] · **Alfred J. Menezes**[2]

Received: 8 January 2015 / Revised: 29 April 2015 / Accepted: 2 May 2015 /
Published online: 13 May 2015

Abstract It has been roughly two decades since the random oracle model for reductionist security arguments was introduced and one decade since we first discussed the controversy that had arisen concerning its use. In this retrospective we argue that there is no evidence that the need for the random oracle assumption in a proof indicates the presence of a real-world security weakness in the corresponding protocol. We give several examples of attempts to avoid random oracles that have led to protocols that have security weaknesses that were not present in the original ones whose proofs required random oracles. We also argue that the willingness to use random oracles gives one the flexibility to modify certain protocols so as to reduce dependence on potentially vulnerable pseudorandom bit generators. Finally, we discuss a modified version of ECDSA, which we call ECDSA$^+$, that may have better real-world security than standard ECDSA, and compare it with a modified Schnorr signature. If one is willing to use the random oracle model (and the analogous generic group model), then various security arguments are known for these two schemes. If one shuns these models, then no provable security result is known for them.

Keywords Cryptography · Public key · Random oracle

Mathematics Subject Classification 94A60

This is one of several papers published in *Designs, Codes and Cryptography* comprising the "Special Issue on Cryptography, Codes, Designs and Finite Fields: In Memory of Scott A. Vanstone".

✉ Alfred J. Menezes
ajmeneze@uwaterloo.ca

Neal Koblitz
koblitz@uw.edu

[1] Department of Mathematics, University of Washington, Box 354350, Seattle, WA 98195, USA

[2] Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON N2L 3G1, Canada

1 Introduction

The random oracle model is a powerful tool introduced by Bellare and Rogaway in [6] in order to make it possible to give rigorous "proofs of security" for certain basic cryptographic protocols, such as Full Domain Hash (FDH) signatures [6] and OAEP encryption [7]. Typically it is a hash function that is modeled by a random oracle. Informally speaking, this means that one regards the hash function H as a black box that responds to a query for the hash value of a bitstring M by giving a random value. For each query the oracle makes an independent random choice, except that it keeps a record of its responses $H(M)$ and repeats the same response if M is queried again.

To say that $H(\cdot)$ can be modeled by a random oracle is a much stronger assumption than collision-resistance, preimage resistance, the pseudorandom function property, and other properties that are commonly assumed to hold for hash functions in various applications. It is natural to wonder whether maybe the random oracle assumption is *too* strong—whether the use of such a powerful model might cause some "provably secure" protocols to be insecure when implemented with a concrete hash function such as SHA-256. We will have more to say about this controversy in later sections.

To give background for this controversy, let us start by recalling the definition of the FDH signature scheme [6], which is one of the simplest and most elegant constructions in cryptography. We then use the random oracle assumption to give a very informal proof that a successful chosen-message attack is equivalent to inverting the trapdoor one-way function, e.g., in the case of RSA-FDH this means inverting the function $f(x) = x^e \bmod N$, where (N, e) is the public key.

Let $f : S \to S$ be a permutation of a finite set S. We assume f to be a *trapdoor one-way* function. In other words, using public information any randomized algorithm has negligible probability (taken over the elements of S and the sets of coin tosses in executing the algorithm) of finding $f^{-1}(y)$ in a reasonable amount of time. Using secret information, f can easily be inverted.

Let H be a hash function—a function from message strings of arbitrary length to the set S—whose values range uniformly over the entire set S. (This is what "full domain" means.)

The basic signature scheme works as follows. The signer Alice possesses a secret key that allows her to invert f. To sign a message M, she finds $H(M)$ and then computes $s = f^{-1}(H(M))$, which is her signature. After receiving M and s, Bob verifies the signature by checking that $f(s) = H(M)$. That's all there is to it.

Here is an informal proof of security of FDH. Suppose that Chris is an (existential) chosen-message forger. This means that he can ask for the signatures of a set of messages M_i of his choice, and then is able to produce the signature of some message M' that's not in the set. Chris queries the random oracle to get the hash values $h_i = H(M_i)$ and $h' = H(M')$, and for each i he queries Alice to get her signature s_i for the message M_i. Because of the assumption regarding randomness of the hash function, the choice of messages M_i and M' is irrelevant. What the forger has to work with is a random sequence of values h_i along with the corresponding $s_i = f^{-1}(h_i)$, and the forger is required to produce f^{-1} of a random h'. The security claim is that this is no easier than producing f^{-1} of a random $h' \in S$ without having the sequence of pairs (h_i, s_i). The informal proof amounts to the rather trivial observation that, since both the h_i and the s_i are randomly distributed over S, you can obtain an equally good sequence of pairs (h_i, s_i) by starting with the random s_i and finding $h_i = f(s_i)$; and the latter process requires only publicly known information. (Note that this uses the "full domain" assumption, i.e., H does not map to a proper subset of S.) In other words, a sequence of random $(h_i, f^{-1}(h_i))$ is indistinguishable from a sequence of random $(f(s_i), s_i)$. It makes no difference whether you look at your sequence of pairs left-to-right or right-to-left. Thus, the proof boils down to the following tautology: *the problem of solving an equation is equivalent to the problem of solving the equation in the presence of some additional data* (h_i, s_i) *that are irrelevant to the problem and that anyone can easily generate*.

This informal argument might be convincing, but it is not a formal proof. A formal proof constructs an actual reduction from the problem of inverting the one-way function to the forger's task. In Sect. 3 of [47] we discuss the formal proof, especially the tightness issue that arises in the reduction.

Could this argument be replaced by one that does not use random oracles? In [33] Dodis et al. gave a negative answer to this question "generically." That is, they proved that security of FDH cannot be proved without random oracles if the trapdoor permutation f is treated as a black box, i.e., if no special properties of its construction are used in the proof.

We shall use the term "ROM-protocol" to mean that the protocol has a security reduction (that is, a reductionist security argument) in the random oracle model but no known security reduction without random oracles. The tradition in the provable security literature is to employ the term *standard model* to refer to any set of properties and hardness assumptions that do not include random oracles. This leads to some questionable uses of the word "standard" (see [49]). We prefer the more neutral term "non-ROM protocol" to refer to a protocol that was constructed so as to have a security reduction without random oracles.

2 The bronze serpent controversy

The first major assault on the validity of the random oracle model was the widely-cited paper [24] by Canetti, Goldreich, and Halevi, who constructed a ROM-protocol that's insecure with any concrete hash function. By their own admission, their construction was contrived and bizarre from the standpoint of real-world cryptography. Nevertheless, two of the three authors arrived at some extremely strong conclusions based on the result. First, according to Canetti (Sect. 6.1 of [24]), "This leaves us no choice but concluding that, in spite of its apparent successes, the Random Oracle model is a bad abstraction of protocols for the purpose of analyzing security." Goldreich (Sect. 6.2 of [24]) was equally blunt:

> Bottom-line: It should be clear that the Random Oracle Methodology is not sound; that is, the mere fact that a scheme is secure in the Random Oracle Model cannot be taken as evidence (or indication) to the security of (possible) implementations of this scheme. Does this mean that the Random Oracle Model is useless? Not necessarily: it may be useful as a test-bed (or as a sanity check). Indeed, if the scheme does not perform well on the test-bed (resp., fails the sanity check) then it should be dumped. But one should not draw wrong conclusions from the mere fact that a scheme performs well on the test-bed (resp., passes the sanity check). In summary, the Random Oracle Methodology is actually a method for ruling out some insecure designs, but this method is not "complete" (i.e., it may fail to rule out insecure designs).

(Halevi's conclusions in Sect. 6.3 of [24] were more moderate and balanced.)

In our first paper in the "Another look" series [47] (posted in 2004), we discussed the random oracle model as part of our attempt to evaluate the strengths and weaknesses of the proofs of security of well-known protocols. In addition to the construction in [24], we looked at the two other most frequently cited examples of failure of the random oracle model (see [8,41]). In all three cases we found that the constructions relied in essential ways on violations of basic principles of sound cryptographic practice. For instance, a certain message triggers the release of the secret key, or a verification step from one part of a hybrid protocol is inserted into another part. Although the examples might be clever and of theoretical interest, they were no cause of concern for practitioners.

Remark 1 The same comment applies to recent efforts to undermine the random oracle model. For example, in [43] the authors show that if indistinguishability obfuscation exists, then there exists a bit-encryption protocol that is secure in the random oracle model but is insecure when the random oracle is instantiated by any concrete function. They do this by modifying a scheme so that, when presented with certain pairs $(x_i, H(x_i))$, it outputs the secret key.

In [47] we went further, arguing that the inability of some of the top researchers in cryptography—the authors of [8,24,41,57]—to come up with a convincing example of any real danger in using ROM-protocols should in and of itself serve as an argument in their favor:

> ...if one of the world's leading specialists in provable security (and coauthor of the first systematic study of the random oracle model [6]) puts forth his best effort to undermine the validity for practical cryptography of the random oracle assumption, and if the flawed construction in [8] is the best he can do, then perhaps there is more reason than ever to have confidence in the random oracle model.

We concluded our discussion by saying that "Our confidence in the random oracle assumption is unshaken."[1]

In response to [47], Goldreich wrote an opinion piece [40] in which he accused us of post-modernism and had especially harsh words for our defense of the random oracle model, which he likened to worship of the Bronze Serpent:

> Indeed, what happened with the Random Oracle Model reminds us of the biblical story of the Bronze Serpent, reproduced next. (See *Numbers* (21:4–8) and *2 Kings* (18:4).) During the journey of the People of Israel in the desert, the prophet-leader Moses was instructed by the Lord to make a "fiery serpent" as a symbolic means for curing people that have been bitten by snakes (which were previously sent by the Lord as a punishment for some prior sin). Several hundred years later, the bronze serpent made by Moses has become an object of idol worship. This led the righteous King Hezekiah (son of Ahaz) to issue an order for breaking this bronze serpent to pieces. Let us stress that the king's order was to *destroy an object that was constructed by direct instruction of the Lord*, because this object has become a fetish. Furthermore, this object no longer served the purpose for which it was constructed. This story illustrates the process by which a good thing may become a fetish, and what to do in such a case... [G]iven

[1] It has long been known that one has to use the random oracle assumption carefully if the protocol uses an iterated hash function, because of the extension attack (see Example 9.64 in [55]). That is, the random oracle assumption essentially says that a deterministic function $H(K, M)$ behaves like a random function to someone who does not know the key K. However, if a message is obtained by adding a suffix to a queried message, then the hash of the whole message is the hash of the suffix with known key, and so the random oracle assumption does not apply. It is because of this extension attack that prefix-MAC, defined by $H(K, M)$, is insecure. Despite the need for caution, in fact this potential pitfall has never, to the best of our knowledge, led anyone to a fallacious proof. In particular, no one ever claimed a security result for prefix-MAC under the random oracle model.

the sour state of affairs, it seems good to us to abolish the Random Oracle Model. [emphasis in original]

Goldreich is not the only researcher who uses strong words to disparage any protocol whose proof relies on the random oracle model. The choice of title of the paper [33], for example, would suggest to the casual observer that the authors had found an insecurity in FDH. However, that is not the case—showing the impossibility of a generic non-ROM proof is very different from actually finding a security weakness. In fact, in the twenty years since the security analysis of FDH in [6] no one has found any actual insecurity, generic or otherwise, related to the use of the random oracle model in the proof.

* * *

The purpose of this paper is to reflect upon how things stand roughly a decade after the controversy described above. First, it should be noted that no real-world protocol failures have been found that result from the use of random oracles; people are still citing the same four theoretical papers [8,24,41,57] to explain why they feel the need to replace ROM-protocols. With this in mind one can only marvel at the extraordinary amount of work that's been devoted to constructing non-ROM replacements for these protocols. In [9] Bellare, Hoang, and Keelveedhi comment:

There is a large body of work on cryptography without random oracles. (A Google Scholar search shows 286 papers with the phrase "without random oracles" in the title, and 3640 with this phrase somewhere in the paper, as of June 6, 2013.)

Our purpose is not to comment on 286 papers. Rather, we examine three of the most important efforts to construct replacements for ROM-protocols (see [13,37,44]) and find that all of the non-ROM constructions have potential security weaknesses that were not present in the original ROM-versions. Following [25], we also describe a remarkable fact about pairing-based protocols: essentially the only ones that are known to be directly vulnerable to induced-fault side-channel attacks on pairing computations are those that were constructed specifically as non-ROM alternatives to earlier ROM-protocols. This is not accidental—the very same feature of the protocol accounts for both the non-ROM proof and the susceptibility to induced fault attacks. This is a case where a tremendous effort devoted to developing a way around what appears to be only a theoretical problem has resulted in greatly increased vulnerability to what in some settings is a significant real-world threat.

On occasion it is possible for protocol development that is motivated by a desire to avoid random oracles to lead to a non-ROM protocol that is superior to the earlier ROM-protocol from the standpoint of both security and efficiency. However, in many cases the opposite occurs, and this is what we describe in this paper.

Finally, we argue that if one is willing to use random oracles, then some important protocols can be modified so as to avoid or reduce dependence on pseudorandom bit generators without losing reductionist security guarantees. In particular, we discuss provable security of ECDSA$^+$, which is arguably an improvement over standard ECDSA from a real-world security standpoint, and of a modified version of Schnorr signatures. If one is willing to use random oracles (and generic groups), then these signature schemes have several reductionist security properties; if not, then, as far as we know, they have none.

3 Gennaro–Halevi–Rabin signatures

The Gennaro–Halevi–Rabin (GHR) signature scheme [37] is an interesting variant on RSA signatures that allowed the authors to prove existential unforgeability against chosen-message attack without using random oracles. It works as follows. Suppose that Alice wants to sign a message M. Her public key consists of an RSA modulus N and a random integer t; here $N = pq$ is a product of two primes such that $(p-1)/2$ and $(q-1)/2$ are also prime. Let $h = H(M)$ be the hash value, where we assume that the hash function H takes odd values (so that there is negligible probability that h has a common factor with $p-1$ or $q-1$). Alice now computes $\widetilde{h}$ such that $\widetilde{h}h \equiv 1 \pmod{p-1}$ and $\pmod{q-1}$. Her signature s is $t^{\widetilde{h}} \bmod N$. Bob verifies Alice's signature by computing h and then $s^h \bmod N$, which should equal t.

Unfortunately, while the GHR scheme is provably secure in the usual Goldwasser–Micali–Rivest [42] security model for signatures, it easily succumbs to a certain type of attack that is outside that model.

3.1 The duplicate signature key selection (DSKS) attack

In a DSKS attack [12] we suppose that Alice, whose public key is accompanied by a certificate, has sent Bob her signature s on a message M. A successful DSKS attacker Chris is able to produce a certified public key of his own under which the same signature s verifies as his signature on the same message M. We are not interested in the trivial attack where Chris simply claims Alice's public key as his own, and so we shall suppose that the certification authority demands a proof of knowledge of the corresponding private key before granting a certificate for the public key.

As discussed in Sect. 2 of [50], although a DSKS attack falls outside the standard Goldwasser–Micali–Rivest security model (which asks only for security against adaptive chosen-message forgers), it can have serious consequences in certain applications of signatures, such as lotteries, coupon redemption systems, and so on.

Following [12], we note that it is easy to mount a DSKS attack on the Gennaro–Halevi–Rabin signature scheme. Suppose that an adversary Chris wants to carry out such an attack. That is, he wants to find N' and t' such that $s^h \equiv t' \pmod{N'}$. But this is simple. He can take an arbitrary RSA modulus N' with $N' > N$ and then just set $t' = s^h \bmod N'$. Notice how little computation is needed—Chris expends no more computational effort than a legitimate user.

In [37] the main motive for introducing Gennaro–Halevi–Rabin signatures was to avoid the use of random oracles. The ease of carrying out a DSKS attack on GHR illustrates a danger in redesigning protocols so as not to need random oracles in a proof—doing so might open up new vulnerabilities to attacks that are outside the security model used in the proof.

4 Boneh–Boyen signatures

In [15] Boneh, Lynn, and Shacham constructed pairing-based short signatures that they showed to be secure in the random oracle model assuming intractability of the Computational Diffie–Hellman (CDH) problem. Three years later Boneh and Boyen [13] proposed a new variant of the signature scheme that they designed with the objective of obtaining a security reduction without using random oracles. There was a price to be paid for avoiding the random oracle assumption. First of all, a Boneh–Boyen signature is about twice as long as a Boneh–Lynn–Shacham signature. In the second place, a Boneh–Boyen signature is more complicated. Namely, suppose that G is a group of prime order q that is used in pairing-based cryptography. The Boneh–Lynn–Shacham signature is simply $s = h^x$, where $h \in G$ is the hash value of the message and x is the secret key. The Boneh–Boyen signature is a pair (r, s), where r is a random integer mod q, and $s = g^{1/(x+h+yr)}$, where g is a (publicly known) generator of the group, the hash h of the message is an integer mod q, and the pair (x, y) of integers mod q is the secret key.

In the third place, the assumption about CDH is replaced by what Boneh and Boyen call the Strong Diffie–Hellman (SDH) assumption, which has been much less extensively studied than the CDH and is presumably a stronger assumption. In the third place, in order to gain confidence in the intractability of this possibly easier SDH problem, in Sect. 5 of [13] they derive a lower bound on the computational complexity of SDH in a generic group—that is, they prove security under the generic group assumption.

The m-SDH problem in a group $\mathbb{G}$ of prime order q is the problem, given group elements $g, g^x, g^{x^2}, \ldots, g^{x^m}$ (where x is an unknown integer mod q), of constructing a pair (c, h) such that $h^{x+c} = g$ (where c is a nonzero integer mod q and h is a group element). The difficulty of this problem can be shown to be less than or equal to that of the CDH problem (which requires the construction of g^{xy} given g, g^x, and g^y). In Sect. 5 of [13] the authors prove that m-SDH in a generic group with a pairing cannot be solved in fewer than (roughly) $\sqrt{q/m}$ operations.

The group $\mathbb{G}$ that is used in the type of cryptosystem in [15] and [13] is called a Gap Diffie–Hellman group [15]. It must have an efficiently computable bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. (More generally, one might have two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$; for simplicity, we are supposing that $\mathbb{G}_1 = \mathbb{G}_2$.) The existence of such a pairing implies that the Decision Diffie–Hellman (DDH) problem (this is the problem, given g, g^x, g^y, and g^z, of determining whether or not $z \equiv xy \pmod{q}$) is efficiently solvable. Informally speaking, the Gap Diffie–Hellman property means that computational problems such as the Discrete Log Problem and the Computational Diffie–Hellman problem are much harder than the DDH—that is, in the inequalities DDH $\leq$ CDH $\leq$ DLP the first is a strict inequality with a large "gap" in difficulty.

The security of a pairing-based protocol rests on the hope that there's a big gap between the DDH and the problem underlying the protocol (and that, in practice, there is no faster way to solve this underlying problem than to solve the DLP in $\mathbb{G}$ or in $\mathbb{G}_T$). For the signature scheme in [13] the underlying problem is m-SDH, where m is a bound on the number of signature queries allowed in a chosen-message attack.

The conjectured gap between Decision Diffie–Hellman and m-SDH is hard to analyze. Strictly speaking, it is not even accurate to speak of a "gap", since in a general group we do not know that DDH $\leq$ m-SDH—that is, we do not know how to use an oracle for m-SDH in order to efficiently solve DDH. While it is reasonable to conjecture that m-SDH is hard in the groups $\mathbb{G}$ that are used in pairing-based cryptography, the fact that we cannot even say for sure that m-SDH $\geq$ DDH might be a cause for concern.

It was because of the difficulty of analyzing the m-SDH assumption that the authors of [13] felt the need to resort to the generic group model. Thus, in order to avoid using random oracles, they used generic groups—even though, as pointed out in [48], the generic model for groups is arguably a much weaker reflection of reality than is the random oracle model for hash functions.

Moreover, a more serious difficulty with the provable security result for the signature scheme in [13] soon came to light. The Boneh–Boyen lower bound $\sqrt{q/m}$ for the difficulty of m-SDH is weaker by a factor of $\sqrt{m}$ than the lower bound $\sqrt{q}$ for the difficulty of CDH in the generic group model. At first it seemed that the factor $\sqrt{m}$ was not a cause for concern, and that the true difficulty of the m-SDH problem was probably $\sqrt{q}$ as in the case of CDH. However, at Eurocrypt 2006 Cheon [27], using the same attack that had been described earlier in a different setting by Brown and Gallant [21,36], showed that m-SDH can be solved—and in fact the discrete logarithm x can be found—in $\sqrt{q/m_0}$ operations if m_0 divides $q-1$ and $m_0 \le m$, $m_0 < q^{1/3}$.

A little later Jao and Yoshida [45] showed that the m-SDH problem is actually equivalent to the forging problem for Boneh–Boyen signatures, and hence the Cheon–Brown–Gallant attack on the m-SDH problem leads to an actual attack on Boneh–Boyen signatures. Jao and Yoshida described an algorithm that for most pairing-friendly curves is able to recover private keys in roughly $q^{2/5}$ time using roughly $q^{1/5}$ signature queries. Thus, given current knowledge, the non-ROM Boneh–Boyen signature scheme in fact has significantly lower security than the ROM-protocol of Boneh–Lynn–Shacham that it replaced. The price of avoiding random oracles was quite steep.

5 Fault attacks on pairing-based protocols

In a fault attack [14] the adversary causes an error in a cryptographic device that's performing an operation with a secret key. The adversary uses the incorrect output, perhaps with other available data, to obtain some information about the secret key. Starting with [59], a number of authors have developed fault attacks on pairing-based protocols. The purpose of the recent paper [25] is for the first time to consider which of the many pairing-based protocols in the literature would actually succumb to any of these fault attacks. It turned out that only a few would be vulnerable.

As in the previous section, let $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a bilinear pairing, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of prime order q. Suppose that $P \in \mathbb{G}_1$ is a publicly known point, and Alice has a secret point $Q \in \mathbb{G}_2$. Suppose that the pairing value $e(P, Q)$ is transmitted during the protocol or is easy to determine from the transmitted values. During a later execution of the same protocol with the same computation of $e(P, Q)$ the adversary induces a fault that causes Alice to compute $e'(P, Q)$. In some circumstances the adversary can use the correct and erroneous values together to compute Q.

Typically $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve groups, and the pairing is computed by a series of iterations of "Miller operations" involving a linear function $L(Q)$. If the adversary is able to cause a certain kind of sign error [68] or a premature termination [59], then a comparison between the correct and incorrect pairing values leads to equations that can be solved for the coordinates of Q.

A necessary condition for this type of fault attack to provide useful information to the adversary is that the protocol must transmit pairing values (or ratios of pairing values) for a pair of points one of which is public and the other secret. And it must transmit the values themselves rather than hashes of the pairing values. The authors of [25] found three protocols that succumb to these attacks—a public-key encryption scheme [17], an identity-based encryption scheme [38], and an oblivious transfer protocol [23].

All three of those protocols had been designed specifically so as to have a security reduction that did not require the random oracle assumption. In all cases a crucial feature that made it possible to avoid random oracles was that actual pairing values rather than their hashes were used; it was this feature that made the protocol vulnerable to the fault attacks in [59,68].

6 Full domain hash

In defiance of the impossibility result in [33], Hohenberger, Sahai, and Waters [44] were able to achieve chosen-message security of FDH signatures without using the random oracle model. Of course, the hash function had to be constructed in a special way in order to accomplish this. Nevertheless, it was an impressive and unexpected result.

The authors of [44] describe their hash function as simply a replacement that preserves the essential nature of the FDH protocol:

> However, [earlier] schemes proven secure without random oracles required *changing the underlying cryptographic scheme* in addition to instantiating the random oracle with a concrete hash function... In other words, can we achieve security *without changing the underlying cryptographic scheme* at all, but only by replacing the random oracle with a specific family of hash functions? In this work, we give the first positive answer to this question... Our first result is creating a *replacement* hash function for the oracle $H(\cdot)$ and developing a security proof without relying on the random oracle heuristic. To keep with our original goals, our only modifications will be to $H(\cdot)$ and we *will use the signature scheme construction as is*, with no changes to the underlying trapdoor permutation family. [emphasis in original]

It turns out, however, that the hash function construction in [44] has certain features which, although in no sense invalidating the results of the paper, nevertheless are a cause of concern. After describing the construction, we'll discuss the properties that under certain circumstances may lead to security weaknesses.

In the version of RSA-FDH in [44], the construction of the hash function depends on the particular user Alice who needs to sign messages. Suppose that Alice has an RSA public key (N, e) and knows the factorization $N = pq$ and decryption exponent d. One also supposes that there is a (publicly known) encoding that maps the message space[2] to a code that has a certain required minimum Hamming distance between the encodings of distinct messages (see Sect. 3.4 of [35] and Sect. 5 of [53]); this is needed for the proof of adaptive security in [44]. We let M' denote the encoding of M.

Alice randomly chooses a constant v prime to N and a sequence of exponents $a_{i,b}$, $b = 0, 1$, $i = 1, \ldots, n$, where n is the bitlength of M'. Given an input M whose encoding M' has bits M'_i, the hash value is defined to be $v^{\pi(M')} \bmod N$, where $\pi(M') = \prod a_{i,M'_i}$. The hash function $H(\cdot)$ is then an obfuscation of the program that computes this. Roughly speaking, that means that $H(\cdot)$ is a program that computes the same values, but examining the obfuscated program reveals no information about the process that computes the values. In other words, $H(\cdot)$ acts like a black box, although the entire program is publicly available for inspection. More precisely, the obfuscation process is assumed to have the *indistinguishability* property, which is weaker than the black-box property [2]: Given two programs of the same size that produce the same outputs, suppose that one of them was the input to the obfuscation process; in other words, it was used to produce the obfuscated program. Then an outsider cannot determine with non-negligible advantage which of the two programs was the one that was obfuscated.

6.1 Some concerns

We now describe some issues that arise with this hash function.

[2] This assumes that the message space is bounded; in [44] it is suggested that if one wants to allow messages of arbitrary length, one should first hash the message using a collision-resistant hash function.

6.1.1 Image of the hash function has skewed distribution

In RSA FDH as described in Sect. 1, a basic requirement is that the hash function maps to all of $\mathbb{Z}/N\mathbb{Z}$ and does so uniformly. In contrast, the image of the hash function in [44] is contained in the cyclic subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ that is generated by v; this is always a proper subgroup $S \subset (\mathbb{Z}/N\mathbb{Z})^*$, since the maximum possible order of v is $\varphi(N)/\gcd(p-1, q-1) \le \varphi(N)/2$. Thus, the non-ROM version of RSA-FDH that is constructed in [44] is not really "Full Domain" Hash.

Moreover, the proper subgroups $S' \subset S$ will generally be hit by the hash function with much greater probability than $1/[S : S']$ (which is the expected probability when the image is uniformly distributed). The exact distribution depends on the random n-tuple of pairs of exponents $(a_{i,0}, a_{i,1})$, as well as on the factorization of $p-1$ and $q-1$.

For example, suppose that we are using 3072-bit RSA with a modulus N such that $\varphi(N)$ is divisible by t distinct primes $p_j \approx 2048$; let us also suppose that all of these primes divide the order of v (which is likely for randomly chosen v), and we are hashing 256-bit messages M with 1024-bit encodings M'. Let S' be the subgroup of S of index $p_1 \cdots p_t$. We would expect that for each j exactly one of the 2^{11} randomly chosen $a_{i,b}$ is divisible by p_j. Suppose that $p_j | a_{i_j, b_j}$ and p_j does not divide any of the other 2047 exponents, $j = 1, \ldots, t$. (Also suppose that $i_j \ne i_{j'}$ for $j \ne j'$.) In this case the hash value of M will be in S' if and only if the i_j-th bit of M' coincides with b_j, $j = 1, \ldots, t$. Thus, the hash values will be spread more thickly on S' than in the case of a uniform map roughly by a factor of $2^{-t} p_1 \cdots p_t \approx 2^{10t}$.

6.1.2 The hash function is partially invertible if the RSA secret key is revealed

A consequence of the skewed distribution is that if the RSA secret key is known, then the hash function can be partially inverted, in the sense that in the above setting the exact order of $H(M)^k$ in the group $(\mathbb{Z}/N\mathbb{Z})^*$, where $k = \varphi(N)/(p_1 \cdots p_t)$, reveals the i_j-th bit of M' for all $j = 1, \ldots, t$. (Note that after finding $p_1, \ldots, p_t$ the adversary can readily determine the set of pairs (i_j, b_j) by computing the exact order of $H(M)^k$ for test-messages M.[3]) Although only the encoding $M \mapsto M'$ is assumed to be publicly efficiently computable, for a linear error-correcting code as in [53] the decoding $M' \mapsto M$ is also likely to be. In that case the partial recovery of M' may lead to significant information about the message M, although how much information depends on the details of the encoding.

As far as we can tell, this partial invertibility does not compromise the actual security of the application to FDH-type signatures. That is, there is no problem as long as Alice uses the hash function only in the way stipulated in [44].

However, in general it is not very good cryptographic practice to use features from one part of a protocol in another part, particularly if knowledge of the private key for the former causes a weakening of the latter. For example, suppose that Alice's RSA private key is stolen. At that point she gets a new key and has the certificate for the stolen key revoked. Her earlier signatures should still, however, be valid (and non-repudiable). And if she injudiciously used her hash function for other purposes unrelated to signing, ideally one would hope that no harm would come. But in the case of the hash function in [44], if she used it to store passwords, then the theft of her RSA signing key could have dire consequences because of the partial invertibility property. This is not, of course, the fault of the authors of [44]. Rather,

[3] Suppose you find the exact order of $H(M)^k$ for each of 50 random messages. The subset of messages M for which this order is divisible by p_j will all have the same bit b_j occurring as the i_j-th bit of the corresponding encodings. Any other bit of those encodings will have both 0's and 1's (except with negligible probability). Thus, you can easily spot (i_j, b_j).

the conclusion is that if a hash function of the type described in [44] is ever deployed, it must be accompanied by a sternly worded warning never to use it as a general-purpose hash function. The security proof for the hash function in combination with FDH-type signatures says nothing at all about security of the hash function in other uses.

6.1.3 A user's hash function must be certified

It is a bit odd for the set-up of a hash function to vary from one user to another, as it does in [44]. Because of the unusual nature of the set-up, users have a special burden if they want to avoid some obvious problems. In particular, the entire obfuscated program is part of Alice's public key and must be certified along with (N, e). Otherwise, an adversary can easily forge the signatures s of arbitrary messages M; he merely chooses an arbitrary function that for each such M takes M' to s^e mod N and then constructs a plausible "obfuscation" of the program that evaluates this function.

6.1.4 The scheme is highly vulnerable to DSKS attacks

Although the authors of [44] say that they have modified FDH "*without changing the underlying cryptographic scheme* at all," it should be noted that in the usual FDH everyone uses the same hash function, whereas in the scheme in [44] each user has her own hash function. This leaves the door open for Duplicate Signature Key Selection (DSKS) attacks.

The obfuscated program for the hash function is part of Alice's key and must be certified; however, the steps Alice went through in order to construct it are known only to her. For everyone else it functions as a black box. This makes the task of a DSKS attacker very easy.

Suppose, for example, that Alice obtains a list of several million lottery numbers ℓ along with the signatures s_ℓ of the rightful holders of the lottery numbers. Before the drawing, she constructs an arbitrary function H such that $H(\ell) = f(s_\ell)$ for all ℓ, where f is her own trapdoor function (i.e., $f(s_\ell) = s_\ell^e$ mod N if (N, e) is her public RSA-key; here N must be chosen larger than all the s_ℓ). Alice certifies a plausible "obfuscation" of the program for H along with her keys. As soon as the winning lottery number is announced, she can immediately claim the winnings, since the winner's signature verifies under her certified key. Notice how general this attack is—it doesn't depend on any particular properties of the trapdoor function. In contrast, in the ROM-version of RSA-FDH with a fixed encryption exponent (i.e., $e = 3$ or $e = 2^{16} + 1$) to the best of our knowledge no DSKS attack is possible (see Sect. 2.2.2 of [50]).

6.1.5 The hash function is prohibitively inefficient

The construction in [44] that allows a non-ROM proof of adaptive security of RSA-FDH requires the signer to perform n multiplications modulo the order of v, where n is the bit-length of the encoding M' of a message M. To hash a 256-bit message suppose we take $n = 1024$. The verifier must use the obfuscated hash program, which must be part of Alice's certified public key. As we shall see, this means that the storage space for the public key, the task of the certification authority, and the running time for verification are all prohibitively large.

The most efficient way known to compute obfuscations uses the method of multilinear pairings in [29]. The complexity of the method grows rapidly with the "level" of the pairing. The construction uses a modulus x_0, which is a product of secret primes. In Sect. 3.1.3 of

[1] it is shown that the bitlength of x_0 is at least $4k^2\lambda^2 \log_2(\lambda)$, where k is the level of the pairing and λ is the security parameter.

Suppose that we want a 128-bit security level. Then the RSA modulus N should have bitlength 3072. Let ℓ be the order of v mod N; since it is likely that $\ell \geq \varphi(N)/4$ (this is virtually certain if N is the product of two primes of the form $2r+1$ with r prime), we shall suppose that ℓ has bitlength 3070. The exponents $a_{i,b}$ should also be assumed to have this bitlength.

The hash program that's obfuscated first computes the 1024-bit encoding M' of the message M, then finds the product $\pi(M')$ mod ℓ of the 1024 exponents a_{i,M'_i}, and finally computes $v^{\pi(M')}$ mod N. The most efficient construction that satisfies the required *indistinguishability obfuscation* property is due to Zimmerman (Appendix to [70]). In that construction the multilinearity level k is of order 2^d, and the number of ring operations modulo x_0 is of order s, where d denotes the depth and s the size of the circuit that's obfuscated. Neglecting the reduction steps mod ℓ and mod N and also the exponentiation,[4] let's compute the depth and size of a circuit that multiplies together the 3070-bit integers. For simplicity we shall assume that the circuit has a single output bit—this is the type of circuit to which the obfuscation constructions apply—and ignore the extension that's necessary to handle a circuit with 3072 output bits.

First let's consider an algorithm that is fairly efficient in terms of circuit size s but not depth d. We note that a circuit with a single output bit satisfies $s \leq 2^d$.

Suppose that each multiplication of 3070-bit integers requires a circuit of depth $\lceil \log_2(3070) \rceil = 12$. Since the 1023 multiplications can be performed in a binary-tree structure with 10 levels, the resulting circuit has depth $12 \cdot 10 = 120$. This leads to the estimate $k > 2^d = 2^{120}$ for the level of multilinearity. Since $4k^2\lambda^2 \log_2(\lambda) \approx 2^{259}$, we find that $x_0 \approx 2^{2^{259}}$ in this case.

In an effort to get a more reasonable bound on k (and on the bitlength of x_0 and the running time), let's consider a more complicated algorithm whose circuit has greater size but much lower depth. In [4] Beame et al. constructed an algorithm with logarithmic-depth circuit for the iterated product problem. If m denotes the bitlength of the integers being multiplied, their circuit has depth $O(\log_2(m))$ and size $O(m^5 \log_2^2(m))$. In our case let's set $m = 2^{12}$ and set the constant in the big-O size estimate equal to 1. Then $s \approx 2^{67}$. Since $s \leq 2^d$, we also have $d \geq 67$. This gives a better bound for k—namely, 2^{67} rather than 2^{120}—but (because of the much greater circuit size) a higher bound $s \approx 2^{67}$ for the number of ring operations. Note that the bitlength of x_0 is at least $4k^2\lambda^2 \log_2(\lambda) \approx 2^{153}$. Thus, each of the 2^{67} ring operations is modulo an integer $x_0 \gg 2^{2^{153}}$.

Some caveats are in order. Some of the estimates we have used could perhaps be substantially improved. For example, the estimates in [1] for the bitlength of x_0 were obtained by considering known attacks on multilinear pairings when used to implement the Coron–Lepoint–Tibouchi protocol for one-round multi-party key exchange [29]; it is possible that smaller x_0 can be employed when multilinear pairings are used for code obfuscation. In addition, in our case the iterated product problem has a special type of input: each integer is taken from a fixed pair $(a_{i,0}, a_{i,1})$ according to the i-th bit of M'. Hence it might be possible to design simpler circuits for this subproblem of the iterated product problem. On the other hand, we have made optimistic choices for big-O constants, and we have neglected the mod-

[4] Remark 1 of [44] suggests that in the exponentiation, rather than first computing $\pi(M')$, "it might be more efficient to incrementally raise an accumulated value to each a_{i,M'_i}." However, such a highly sequential algorithm would have circuit depth in the thousands, and hence its obfuscation would have a multilinearity level whose bitlength is also in the thousands.

ular exponentiations. We have also neglected to account for the extension one needs to use obfuscation for a circuit that has 3072 output bits rather than just 1. In any case, even after substantial improvement in the estimates, the bitlength of x_0 and the running time are likely to remain way above the practical range.

It should also be noted that Zimmerman's results [70] are proved in a certain generic model of the multilinear maps—under an assumption that is similar to the generic group assumption that's used in some security reductions for Diffie–Hellman and elliptic curve protocols (see [48] and Sect. 7 below). It is open to question whether one should use such a construction to implement the obfuscation for the FDH in [44], since the whole purpose of that paper was to avoid the random oracle model, and one cannot plausibly claim that the generic model for multilinear maps rests on more solid ground than the random oracle model. That is, it does not seem prudent to rely on the generic model for multilinear maps in order to avoid the random oracle model for hash functions. But if one uses the constructions that predate [70], all of which use matrix branching programs, one gets much worse estimates than in [70].

Recall that the obfuscation—including the value x_0—is part of the public key and must be certified by the CA, and signature verification involves arithmetic modulo x_0. Clearly none of this is remotely possible in practice. In particular, factoring the RSA modulus N (by the number field sieve), thereby completely breaking the system, is far easier than verifying a signature.

The word "practical" in the title of [29] is justified by the application to one-round ℓ-party Diffie–Hellman key exchange, for which timings are given in Sect. 6.4 for $\ell = 7$. In that case one needs one application of a 6-linear pairing, and the implementers found that a key exchange at the 80-bit security level requires roughly 20 s. However, if we extrapolate to enormous levels of multilinearity—as in the case of RSA-FDH with 128-bit security—we obtain humongous running times.

Many commentators have noted that cryptographic research is characterized by a particularly sharp divide between theory and practice; for an interesting perspective on this issue, see Bellare's invited talk at Crypto 2014 [5]. Reading the obfuscation literature, even those of us who are accustomed to this gap are startled when we encounter the vast chasm that separates obfuscation theory from cryptographic practice.

Remark 2 An obfuscated program for evaluating a punctured PRF is used in [65] to get a signature scheme with short signatures and fast signature generation that can be proved secure without random oracles. However, signature verification is prohibitively slow, as the authors acknowledge. In addition, the public key is huge, since it contains the obfuscated program. This is a very high price to pay for slightly shorter and faster signatures and no random oracles.

Remark 3 For evidence of the impracticality of indistinguishability obfuscation, see [1]. The authors implemented an obfuscation of a 16-bit point function at the 52-bit security level. Using a 32-core machine, the time for evaluating the obfuscated circuit was 3 h, and the size of the obfuscated program was 31 GB. More recently, Bernstein et al. [11] presented several techniques for evaluating obfuscated programs, which yield a more than 50-fold speedup in the evaluation of the obfuscation circuit for the 16-bit point function.[5] In practice one would need a far more elaborate function to evaluate $H(\cdot)$ with a 3072-bit modulus N, for instance; and one would want at least a 128-bit security level.

Remark 4 In November 2014, several researchers [16,28,39] devised polynomial-time attacks on the multilinear pairing construction in [29] that recover all of the secret quantities.

[5] Bernstein D.: Personal communication, 23 Feb 2015.

The authors of [16] and [39] proposed a modification to the multilinear pairing construction in [29] that appears to resist the new attacks. However, shortly afterwards these countermeasures were shown to be ineffective [30]. In February 2015, Coron, Lepoint, and Tibouchi [31] proposed a variant of their multilinear pairing construction that they claim resists the new attacks. While the impact of the attacks on the security of obfuscation remains to be determined, the attacks serve as a reminder that a protocol that is excessively complicated is likely to have subtle vulnerabilities.

6.1.6 All of the obfuscation constructions are very complicated

Constructing a version of FDH using obfuscation is an elaborate process. The ROM-version of FDH, in contrast, is one of the simplest constructions in cryptography—just hash and apply the trapdoor function. One cannot use a raw off-the-shelf NIST-standardized hash function, because its output does not have length equal to that of the full domain of the trapdoor function, as required. However, it is not hard to make the necessary adjustments, obtaining a protocol all of whose ingredients have been studied and cryptanalyzed by many people over many years.

6.2 KISS

The famous keep-it-simple (KISS) principle of engineering applies with special force to cybersecurity. In the first place, formal analysis is, in general, much more difficult for a complicated system than for a simple one. When a suite of protocols relies on a whole menagerie of little-studied complexity assumptions, one is truly entering a "brave new world" that rests on an untested foundation, as we argued in [49].

Security through obscurity—a once-popular notion that was the antithesis of KISS—was, of course, discredited long ago. But beyond that, security experts have become increasingly aware of how hard it is to analyze and take countermeasures that protect a system that is far more complicated than necessary. In particular, the possible forms that a side-channel attack could take become multiplied as the protocol construction acquires layer upon layer of complexity. (We discuss this in Sect. 4 of [50].)

Most of the time when protocols are constructed for the purpose of duplicating the functionality of ROM-protocols while avoiding random oracles, the resulting system is far more complicated than the one it replaces. If there were any convincing evidence that ROM-protocols have real-world security weaknesses, then we would have to bite the bullet and do our best to analyze these new protocols. However, we have not seen any such evidence. Thus, it does not make much sense to sacrifice either efficiency or true security for the sole purpose of getting security proofs without random oracles. As the popular aphorism says, "If it ain't broke, don't fix it!"

7 The value of random oracles: the example of ECDSA

An underlying premise of this paper is that reductionist security proofs are of value. If they were not, then the whole question of whether or not random oracles should be used in proofs would be moot. That is, practitioners who believe that security proofs are worthless—and who adhere to the viewpoint (attributed to Lars Knudsen) that "if it's provably secure, then it probably isn't"—have no reason to read this paper.

So let's agree to accept the premise that, as we stated in [51], "security proofs are useful as a minimal type of assurance." Once we accept this premise, it follows that whenever we replace a well-studied cryptographic protocol by an alternative version that we believe improves on its real-world security, we would want the reductionist security results to carry over to the modified version.

Lately the cybersecurity world has become aware of startling security lapses in commonly used pseudorandom bit generators. At Crypto 2012 Lenstra reported that many users' RSA private keys can be easily revealed because of a faulty implementation of random selection of prime factors of the modulus (see [52]). In August 2013 it was discovered that a bug in the Android operating system had been causing a defective implementation of pseudorandom number generation, resulting in a large-scale theft of Bitcoins [22], among other things. One month later the public learned from the Edward Snowden leaks [61] that the NSA had put a backdoor in the NIST-standardized Dual Elliptic Curve Deterministic Random Bit Generator (EC_DRBG), which was included as the default in RSA's BSAFE toolkit. These scandals have made practitioners more aware than ever of the pitfalls of heavily relying on random number generation.

One of the responses has been to find ways to reduce the dependence on PRBGs. For instance, the usual version of the Elliptic Curve Digital Signature Algorithm (ECDSA) not only relies on a random number for key generation, but also needs a new random value k for each message that is signed. The security of ECDSA is particularly sensitive to poor implementation of pseudorandom generation of k. For example, in [58] it was shown that if just three bits of k are known for several hundred signatures, then it is possible to recover the static secret key K. Starting with Barwood [3] and Wigley [69], a number of people (see [64]) have proposed replacing pseudorandom generation of k by a deterministic value $k = H'(K, M)$ obtained by hashing the message and the static secret key. See also Sect. 2 of [10] for a discussion of the rationale for this modification of ECDSA.

Remark 5 An advantage of stipulating $k = H'(K, M)$ rather than random k is that it forces the user to change k for each message. In 2011 Sony's implementation of ECDSA for PlayStation 3 was completely broken because they used a constant k; see [34]. An adversary who has two different messages signed with the same k can easily compute the private key.

7.1 ECDSA and ECDSA*

Before discussing security of the modified version of ECDSA, we first recall how Alice signs a message in ECDSA. Let $\mathbb{G}$ be a subgroup of prime order q of the group of points on an elliptic curve defined over a finite field, where we suppose that it is computationally infeasible to find discrete logarithms in $\mathbb{G}$. Let $P \in \mathbb{G}$ be a fixed basepoint. Alice's private key is a random integer K mod q, and her public key is the point $Q = KP$. Let H be a hash function whose values are integers mod q. To sign a message M, Alice first randomly selects an ephemeral secret integer k mod q that must be different for each message she signs. She computes kP and lets r denote the x-coordinate of kP, regarded mod q.[6] She computes $s = k^{-1}(H(M) + Kr) \bmod q$; her signature is (r, s). To verify Alice's signature, Bob checks that the x-coordinate of $s^{-1}H(M)P + s^{-1}rQ$ is congruent to r mod q.

We let ECDSA* denote the modified version with $k = H'(K, M)$, where H' is a hash function. (It could be—but doesn't have to be—the same hash function that is used in the signing equation. We shall later make different assumptions about the two hash functions, so

[6] This definition of r applies to ECDSA over a prime field; a different method of determining r from kP is needed for ECDSA over a binary field.

if the same hash function is used in both positions, then $H = H'$ must be a hash function for which both sets of assumptions are believed to hold.) Before replacing ECDSA by ECDSA* one would want to investigate whether or not the reductionist security results for the former protocol carry over to the modified version.

7.2 Security reductions

Recall that we say that a signature scheme is *existentially unforgeable against adaptive chosen-message attack* if an adversary that is permitted to interact with a signer, getting valid signatures for a bounded number of messages that the adversary selects during the interaction, cannot feasibly produce a valid forgery of any message at all that was not queried.

The definitive work on provable security for ECDSA was done by Dan Brown in the early 2000s (see [19]). He proved that in the generic group model ECDSA is existentially unforgeable against adaptive chosen-message attack provided that certain conditions are satisfied by the hash function (collision resistance, a uniformity property, and intractability of finding a message M with $H(M) = 0$).[7] A natural question to ask about replacing the random k in ECDSA by deterministic $k = H'(K, M)$ is whether Theorem 3 of [19] carries over.

The answer is that it does, provided that one adopts the random oracle assumption. In other words, one can get a security theorem if one models the hash function H' as a random oracle. In that case the values of k appear random to an attacker, since no information at all about k can be predicted by someone who does not know the full secret key K as well as the message.[8]

Can one dispense with random oracles in going from ECDSA to ECDSA*? The result also carries over under the weaker assumption that H' is a pseudorandom function (PRF), provided that one uses a separate independent secret key K' to generate k. (We give an outline of a proof below.) This modification is obviously not desirable from a practical standpoint, since it would mean that each user has to have two static keys (or, equivalently, a key of twice the length).[9] Apparently the only advantage of using different keys in the signing equation and in the generation of k is that doing so makes the proof of security go through under the PRF assumption rather than the random oracle assumption.

Let ECDSA** denote the two-key version of ECDSA with $k = H'(K', M)$. We sketch an informal proof that ECDSA** has the same reductionist security as ECDSA (against restricted adversaries, which do not query the same message more than once) provided that H' is a PRF. Recall the (informal) definition of the PRF property: Suppose an attacker queries an oracle that has equal probability of always giving a random value $O(M)$ or always giving $H'(K', M)$ with hidden key K'. In reasonable time the attacker is unable to guess which it is doing with a non-negligible advantage (that is, with probability non-negligibly greater than 1/2 of being correct).

[7] In [20] Brown also proved unforgeability in the random oracle model for the hash function but without the generic group assumption. However, he needed to make the so-called semi-logarithm assumption, which has been little studied and is presumably stronger than intractability of discrete logs.

[8] More precisely, in Table 1 of [19] the description of the oracle's response to a "hint command" (that is, a signature query) is modified as follows. The input is a message M rather than a hash value h, and instead of random generation of z_{m+1} one queries $(K, M) = (z_2/z_1, M)$ to the hash oracle. A few minor modifications are then needed in the proof of Lemma 1 of [19].

[9] Two-key versions of signature schemes are just as impractical for deployment as are two-key message authentication codes. See [51] for an analysis of the security theorems that under suitable conditions enable one to carry over security results from two-key nested MACs to one-key variants.

To prove that ECDSA** has the security properties of ECDSA provided that H' is a PRF, we show that an algorithm A that successfully attacks ECDSA** but not ECDSA could be used to construct an algorithm B that breaks the PRF property of H'. Namely, we construct B as follows. Its input is the oracle in the definition of the PRF property. The algorithm B can generate a random secret key K. As it interacts with the algorithm A, it uses the oracle to produce the ephemeral secret k when it answers A's queries by giving either ECDSA signatures (if the oracle is $O(M)$) or ECDSA** signatures (if the oracle is $H'(K', M)$ for some hidden K'). If A produces a forgery, then B guesses that the oracle is $H'(K', M)$, and if not, then B guesses that the oracle is $O(M)$. To put it another way, the fact that A is successful against ECDSA** and not against ECDSA can be converted into a test of whether one is being given a random oracle or $H'(K', M)$.

This proof breaks down for ECDSA* because the party giving ECDSA* signatures has to know the key for H' in order to produce the signing equation. In other words, the PRF property of H' only gives randomness from the perspective of someone who has no information at all about the key. The signer has complete knowledge of the key, and the attacker, although he doesn't know the key, has some indirect information about it (since it was used twice in the signing equation, including its appearance in the formula for k). Most likely this distinction between ECDSA* and ECDSA** has no significance for security in practice, but in theory (that is, in carrying out a formal proof) it makes a big difference.

Remark 6 One could get a proof without random oracles that ECDSA* is as secure as ECDSA** by making the assumption that the hash function and the elliptic curve group satisfy the following 1-key/2-key condition: Someone who can ask queries of the signer cannot determine with non-negligible advantage whether $k = H'(K, M)$ or $k = H'(K', M)$, where K is the static key used in the signing equation and K' is another static key that is independent of K. In other words, one could simply assume that ECDSA* and ECDSA** are indistinguishable from one another. However, as we argued in [49], a security proof is of questionable value if it relies upon an unnatural and unstudied assumption that is tailor-made for a given protocol and causes the security reduction to become an exercise in circular reasoning. Such an approach is not, in our view, preferable to the use of the random oracle model, which leads to more substantive results.

7.3 A further modification of ECDSA

In Sect. 5 of [47] we discussed security reductions for the Schnorr signature scheme [66] and compared that scheme to DSA.[10] Informally speaking, the security of both schemes is based on intractability of the discrete log problem in a finite field. However, the security reductions for Schnorr signatures in the random oracle model [62,63] were lost in going from Schnorr to DSA. We commented that one could retain some reductionist security in DSA if instead of simply hashing the message M in the signing equation one hashes both M and a value that depends on the ephemeral key k. Since ECDSA was modeled on DSA and not on Schnorr signatures, the same remark applies to ECDSA.

Intuitively, the reason why hashing only M in the signing equation gives more power to the forger than hashing both M and a value that depends on k is that in the latter case the hash

[10] A somewhat similar modification of ECDSA was proposed by NIST in [32]. The NIST modification randomizes the message by combining it with an ECDSA signature component; however, the procedure for generating ephemeral secret keys is the same as in ECDSA. The NIST modification appears not to be sufficient for obtaining the reductionist security result we want. We also note that Brickell et al. [18] obtained several security results for modifications of DSA, that is, for a broad class of signature schemes based on the discrete log problem in a finite field.

computation requires prior knowledge of k, whereas in the former case the forger can wait to choose the ephemeral key k until after seeing the hash value $H(M)$. Note, by the way, that even though legitimate ECDSA* signers determine k deterministically as $k = H'(K, M)$, any choice of k will lead to a signature that satisfies the verifier, who has no way of knowing whether k was computed deterministically, randomly, or in some other way. So of course one cannot assume that the forger's k is $H'(K, M)$.[11]

Thus, suppose we further modify ECDSA* by replacing $H(M)$ by $H(kP, M)$ in the signing equation and define the signature to be (kP, s); let ECDSA$^+$ denote the resulting scheme.[12]

We note that for greater efficiency one doesn't have to hash both coordinates of kP, and one doesn't have to include both coordinates in the signature. One can take just the x-coordinate along with a single bit that indicates which of two possible y-coordinates is the y-coordinate of kP.

Hence, in ECDSA$^+$ Alice signs a message as follows. As before, $P \in \mathbb{G}$ is a fixed basepoint in the subgroup of prime order q of the group of points on an elliptic curve. Alice's private key is a random integer K mod q, and her public key is the point $Q = KP$. Let H and H' be hash functions whose values are integers mod q. To sign a message M, Alice sets $k = H'(K, M)$ and computes $R = kP$. Let r denote the x-coordinate of R, regarded as an integer mod q; and let $\widetilde{R}$ denote a bitstring that includes the x-coordinate of R along with an additional bit that indicates which of two possibilities is the y-coordinate of R. Alice computes $s = k^{-1}(H(\widetilde{R}, M) + Kr) \bmod q$; her signature is $(\widetilde{R}, s)$. To verify Alice's signature, Bob extracts r from $\widetilde{R}$ and then checks that $s^{-1}H(\widetilde{R}, M)P + s^{-1}rQ$ is the point R.

It is not hard to see that Theorem 3 of [19] carries over to ECDSA$^+$. That is, our first claim is that *ECDSA$^+$ is existentially unforgeable against adaptive chosen-message attack under the generic group assumption on* $\mathbb{G}$, *the random oracle assumption on H', and the conditions on H (collision-resistance, a uniformity property, and intractability of finding a preimage of 0) that are given in* [19].

In addition, a version of the security result proved for Schnorr signatures can also be proved for ECDSA$^+$. Unlike Brown's results, these proofs need only intractability of discrete logs rather than the much stronger generic group assumption. Thus, our second claim is that *ECDSA$^+$ is existentially unforgeable against adaptive chosen-message attack if the discrete log problem in $\mathbb{G}$ is intractable and the hash functions H and H' are modeled by random oracles.* Following [62,63], we now give an informal proof of this claim.

Informal proof Suppose we wish to find the discrete logarithm K of Q to the base P: $Q = KP$. We use the forger to forge ECDSA$^+$-signatures with public key Q. Whenever the forger needs an H-value or H'-value, we simulate an oracle that gives random values in response to the forger's hash queries (except that the same value is returned if the forger makes the same query a second time). Notice that k is determined before the forger asks for $H(\widetilde{R}, M)$, where $R = kP$; this is a crucial distinction between ECDSA$^+$ and ECDSA*.

We also need to simulate a signing oracle that responds to the forger's signature queries. Whenever the forger requests a signature on a message M, we randomly choose two integers

[11] Strictly speaking, one cannot even assume that the forger has selected a k at all, since all that is required of a forger is a message and signature that satisfy the verification algorithm. The preceding discussion is intended to be informal and intuitive; it is not a formal argument.

[12] A very similar version was considered in [54], where it was denoted ECDSA III. The version of Malone-Lee and Smart differs from ECDSA$^+$ in just two respects: the ephemeral key k is random rather than given by $H'(K, M)$, and in the signing equation instead of kP (that is, the x-coordinate along with an additional bit to indicate y) they hash the sum of the x- and y-coordinates along with M.

u and v mod q and set $R = uP + vQ$ and $s = v^{-1}r$ mod q (where r is the x-coordinate of R; if $r = 0$ we change our choice of u and v). We return the signature $(\widetilde{R}, s)$. In order to verify the signature the forger has to ask for $h = H(\widetilde{R}, M)$, and in response to that hash query we send $h = us$ mod q. Since $s^{-1}(hP + rQ) = uP + vQ = R$, the signature verifies. It should be noted that in order for this to be a valid simulation, one has to check that h is independent of R and random, but this is an easy consequence of the fact that the pairs (u, v) for which $uP + vQ = R$ form a line in the uv-plane mod q, and h ranges through the integers mod q along that line.

Let q_H be a bound on the number of queries for an H-value. We choose a random index $j \le q_H$ and hope that the j-th queried value $H(\widetilde{R}, M)$ happens to be the one the forger uses to produce a signature $(\widetilde{R}, s)$ on the message M. Let ϵ be the forger's success probability, and let ϵ_j be the probability that the j-th H-value leads to the forged signature; then $\sum_j \epsilon_j = \epsilon$.

We use two copies of the forger. We give both forgers the public key Q, the same sequence of random bits, and the same random answers to their hash queries until they both make the same j-th H-query. At that point we give two independent random answers h_1 and h_2,[13] and from then on we give the two forgers different sequences of random bits and different random function values. That is, our interaction with the forgers "forks" after the j-th H-query.

We hope that both forgers produce signatures corresponding to the same M and $\widetilde{R}$. If they do not (or if we're unable to proceed because no j-th H-query is made), then we start over again. If the two forgers do output signatures $(\widetilde{R}, s_1)$ and $(\widetilde{R}, s_2)$, then we have the following two congruences mod q:

$$ks_1 \equiv h_1 + rK \qquad \text{and} \qquad ks_2 \equiv h_2 + rK.$$

Since k (which is unknown to us) is the same in both equations, we can solve for K in terms of known values: $K = -r^{-1}(s_2h_1 - s_1h_2)/(s_2 - s_1)$ mod q.

We need to know that there's a non-negligible probability that this will all work out as we hope. The bounding of this probability from below is called the "forking lemma" [62,63]. Let S denote the set of all possible sequences of random bits and random hash responses during the course of the above procedure. Let A be the set of possible sequences of random bits and random hash values until the forgers' j-th H-query, and let B be the set of sequences of random bits and random hash values after that. Suppose that there are a elements in A, b elements in B, and ab elements in $S = A \times B$. For $\epsilon_j ab$ values in S the forger produces a valid signature with $\widetilde{R}$ and M from the j-th H-query. Applying the "splitting lemma,"[14] we can say that there are at least $\epsilon_j a/2$ elements of A that have the following property: the remaining part of the forgery algorithm (starting with the j-th H-query) has probability at least $\epsilon_j/2$ of leading to the desired signature. For each such element of A the probability that both copies of the forger lead to such signatures is at least $(\epsilon_j/2)^2$. Thus, the probability that an element of $A \times (B \times B)$ will lead to two different signatures with the same $\widetilde{R}$ and M is at least $(\epsilon_j/2)^3$.

Since j is chosen uniformly at random from $\{1, 2, \ldots, q_H\}$, the probability that the above procedure leads to two signatures of the same message is at least $\frac{1}{8q_H}\sum \epsilon_j^3$. The minimum of this sum subject to the condition that $\sum \epsilon_j = \epsilon$ is achieved when all the ϵ_j are equal, that

[13] We shall ignore the possibility that the j-th H-query repeats an earlier query, in which case we need to choose a different j, and also the (negligible) probability that $h_1 = h_2$.

[14] The splitting lemma [62,63], which is proved by a simple counting argument, says the following. Suppose we have two sets A and B containing a and b elements, respectively. Suppose that a pair $(\alpha, \beta) \in A \times B$ has probability $\ge \epsilon$ of having a certain property, that is, there are at least ϵab such "good" pairs. Let $A_0 \subset A$ be the set such that $\alpha_0 \in A_0$ if at least $\epsilon b/2$ pairs (α_0, β) $(\beta \in B)$ are "good." The splitting if at least $\epsilon b/2$ pairs (α_0, β) $(\beta \in B)$ are "good." The splitting lemma says that there are at least $\epsilon a/2$ elements in A_0.

is, when $\epsilon_j = \epsilon/q_H$ for all j. Thus, the probability of success in one iteration of the above algorithm is at least $\frac{1}{8q_H} \cdot q_H \cdot (\epsilon/q_H)^3 = (\epsilon/2q_H)^3$. This gives us a non-negligible success probability, although with a very large tightness gap. □

Remark 7 Because of the tightness gap, Theorem 3 of [19] can be viewed as much superior. On the other hand, Brown's result uses the generic group assumption, which is much stronger than intractability of the discrete log problem. So having a second security reduction based on the discrete log problem is perhaps worthwhile.

7.4 Schnorr signatures

We modify the Schnorr signature scheme in [66] by using an elliptic curve group and also by determining the ephemeral key k deterministically rather than randomly. We call the resulting scheme ECSchnorr*. Thus, Alice signs a message as follows. As before, her private key is a random integer K mod q, and her public key is $Q = KP$, where P is the basepoint; and H and H' are hash functions whose values are integers mod q. To sign a message M, Alice sets $k = H'(K, M)$ and computes $R = kP$. Let $\widetilde{R}$ denote a bitstring that includes the x-coordinate of R and an additional bit indicating the y-coordinate of R. Alice computes $h = H(\widetilde{R}, M)$ and sets $s = k + hK$ mod q; her signature is (h, s). To verify the signature Bob computes the point $S = sP - hQ$ and then $H(\widetilde{S}, M)$, and accepts the signature if this hash value is equal to h.

There are two main positive results about the security of Schnorr signatures, both of which carry over to ECSchnorr* under the random oracle assumption for H':

- Pointcheval and Stern [62,63] proved unforgeability under chosen-message attack, assuming intractability of the discrete log problem (DLP) and the random oracle model for the hash function H. Their result (even after subsequent improvements) has a large tightness gap, however.
- Neven, Smart, and Warinschi [56], working in the generic group model, proved existential unforgeability under chosen-message attack if the hash function H is preimage-resistant in certain senses (this is a weaker condition than collision-resistance). Their reduction is tight.[15]

There are also two negative results due to Paillier and Vergnaud [60]; see also [67]. Informally speaking, they showed that it is unlikely that a reduction from the DLP to unforgeability of Schnorr signatures exists under a "standard" assumption (meaning that the hash function cannot be modeled by a random oracle, and the elliptic curve group cannot be modeled by a generic group), and even in the random oracle model it is unlikely that a tight reduction exists. In [46] we ask

> What do we make of the circumstance that, apparently, no tight reduction from the DLP to forgery is possible in the random oracle model, and no reduction at all is likely in a standard model? As usual, several interpretations are possible. Perhaps this shows that reductions in the random oracle model are dangerous, because they lead to security results that cannot be achieved in a standard model. On the other hand, perhaps we can conclude that the random oracle model should be used, because it can often come closer to achieving what our intuition suggests should be possible.

[15] Tightness issues arise in [56] if one wants to use short hash values, which would give a 25 % reduction in signature length compared with ECDSA and compared with Schnorr with full-length hash.

Based on the two types of security results listed above, in [56] Neven, Smart, and Warinschi contrast Schnorr with ECDSA, arguing in favor of the former:

> The proof [by Brown for ECDSA] is quite involved and reduces the security of EC-DSA to a set of non-standard properties of the hash function and the "conversion function." We feel our result for Schnorr is cleaner, and the associated hash function properties are more natural. Combining our results in the generic group model with the advantage of additionally having a security proof in the random oracle model, we feel that Schnorr signatures are to be preferred over EC-DSA.

However, whether Schnorr signatures have a significant reductionist security advantage over ECDSA is debatable, because of the tightness issue. In the generic group model both schemes have tight security results. The result for Schnorr signatures in [56] makes milder assumptions on the hash function H than Brown's Theorem 3 in [19]. Because collision-resistance is not needed, in principle one can shorten hash lengths by a factor of two. However, that causes a loss of tightness in the reduction in [56]. In [56] the tightness issue is discussed in this connection and in connection with the Pointcheval–Stern results, but it is considered to be of secondary importance. The authors point out that no attack is known that exploits the tightness gaps, and that practitioners rarely modify their parameters to take non-tightness into account. In other words, even if the concrete security guarantee with the chosen parameters that results from the security reduction turns out to be meaningless, that should not be a cause of great concern.

We discussed this issue at length in [46]—see also [26]—and we don't find this argument convincing. If one believes that rigorous security reductions are of some importance as part of an evaluation of a scheme—and we accepted this premise at the beginning of this section—then one would hope that the actual concrete guarantee given by the proof would be taken seriously.

In the random oracle model the Pointcheval–Stern results for Schnorr are comparable to the corresponding results for ECDSA (Theorem II.10 of [20]) or ECDSA$^+$ (our claim and informal proof in Sect. 7.3). Brown's Theorem II.10 not only has a tightness gap of q_H, but assumes intractability of an unnatural and little studied problem (semi-logarithms) that is possibly easier than the DLP and in fact is similar to the forgery problem itself. Our result for ECDSA$^+$ gives a reduction to the DLP, but it is very non-tight.

It is not clear to us which of ECDSA+ and ECSchnorr* is better from a "provable security" standpoint. But one thing that is clear is that if one shuns the random oracle model and the analogous generic group model, then no security reduction is known for either scheme. If one accepts those "non-standard" models, then formal security analysis is possible in both cases.

* * *

In their seminal paper [6] Bellare and Rogaway succinctly summarized the value of the random oracle model as follows: "Goals which are possible but impractical in the standard setting become practical in the random oracle setting." Over twenty years later this is still the case, as shown by the elliptic curve signature schemes ECDSA*, ECDSA$^+$, and ECSchnorr*.

Acknowledgments We would like to thank Dan Brown for valuable discussions of security reductions for ECDSA, Kenwrick Mayo for useful discussions of obfuscation constructions, Sanjit Chatterjee for thoughtful comments on an earlier draft, and Ann Hibner Koblitz for helpful editorial suggestions. We would also like to thank Dan Bernstein for informing us of the work [11] and Francisco Rodríguez-Henríquez for bringing the paper [54] to our attention. Finally, we thank the referees for their helpful comments.

References

1. Apon D., Huang Y., Katz J., Malozemoff A.: Implementing cryptographic program obfuscation, Crypto 2014 rump session (2014). http://eprint.iacr.org/2014/779.
2. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S., Yang K.: On the (im)possibility of obfuscating programs. J. ACM **59**, 6 (2012).
3. Barwood G.: Digital signatures using elliptic curves (1997). http://groups.google.com/group/sci.crypt/msg/b28aba37180dd6c6.
4. Beame P.W., Cook S.A., Hoover H.J.: Log depth circuits for division and related problems. SIAM J. Comput. **15**, 994–1003 (1986).
5. Bellare M.: Caught in between theory and practice. In: Crypto 2014 IACR Distinguished Lecture (2014). https://www.youtube.com/watch?v=SPVWSG7-i_E.
6. Bellare M., Rogaway P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM, New York (1993).
7. Bellare M., Rogaway P.: Optimal asymmetric encryption—how to encrypt with RSA. In: Advances in Cryptology—Eurocrypt'94. LNCS, vol. 950, pp. 92–111. Springer, Berlin (1994).
8. Bellare M., Boldyreva A., Palacio A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Advances in Cryptology—Eurocrypt 2004. LNCS, vol. 3027, pp. 171–188. Springer, Berlin (2004).
9. Bellare M., Hoang V.T., Keelveedhi S.: Instantiating random oracles via UCEs. In: Advances in Cryptology—Crypto 2013 (Part II). LNCS, vol. 8042, pp. 398–415. Springer, Berlin (2013); full version available at http://eprint.iacr.org/2013/424.
10. Bernstein D., Duif N., Lange T., Schwabe P., Yang B.-Y.: High-speed high-security signatures. J. Cryptogr. Eng. **2**, 77–89 (2012).
11. Bernstein D., Hülsing A., Lange T., Niederhagen R.: Bad directions in cryptographic hash functions, preprint (2015); available at http://obviouscation.cr.yp.to/obviouscation-20150223.
12. Blake-Wilson S., Menezes A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Public Key Cryptography—PKC 1999. LNCS, vol. 1560, pp. 156–170. Springer, Berlin (1999).
13. Boneh D., Boyen X.: Short signatures without random oracles. In: Advances in Cryptology—Eurocrypt 2004. LNCS, vol. 3027, pp. 56–73. Springer, Berlin (2004).
14. Boneh D., DeMillo R., Lipton R.: On the importance of checking cryptographic protocols for faults. J. Cryptol. **14**, 101–119 (2001).
15. Boneh D., Lynn B., Shacham H.: Short signatures from the Weil pairing. In: Advances in Cryptology—Asiacrypt 2001. LNCS, vol. 2248, pp. 514–532. Springer, Berlin (2001).
16. Boneh D., Wu D., Zimmerman W.: Immunizing multilinear maps against zeroizing attacks (2014). Available at http://eprint.iacr.org/2014/930.
17. Boyen X., Mei Q., Waters B.: Direct chosen ciphertext security from identity-based techniques. In: 12th ACM Conference on Computer and Communications Security—CCS'05, pp. 320–329. ACM, New York (2005).
18. Brickell E., Pointcheval D., Vaudenay S., Yung M.: Design validations for discrete logarithm based signature schemes. In: Public Key Cryptography—PKC 2000. LNCS, vol. 1751, pp. 276–292. Springer, Berlin (2000).
19. Brown D.: Generic groups, collision resistance, and ECDSA. Des. Codes Cryptogr. **35**, 119–152 (2005).
20. Brown D.L.: On the provable security of ECDSA. In: Blake I., Seroussi G., Smart N. (eds.) Advances in Elliptic Curve Cryptography, pp. 21–40. Cambridge University Press, Cambridge (2005).
21. Brown D., Gallant R.: The static Diffie–Hellman problem (2004). http://eprint.iacr.org/2004/306.
22. Buterin V.L.: Critical vulnerability found in Android wallets. http://bitcoinmagazine.com/6251/critical-vulnerability-found-in-android-wallets/. Accessed 11 Aug 2013.
23. Camenisch J., Neven G., Shelat A.: Simulatable adaptive oblivious transfer. In: Advances in Cryptology—Eurocrypt 2007. LNCS, vol. 4515, pp. 573–590. Springer, Berlin (2007).
24. Canetti R., Goldreich O., Halevi S.: The random oracle model, revisited. In: Proceedings of 30th Annual Symposium Theory of Computing, pp. 209–218, ACM, New York (1998); full version available at http://eprint.iacr.org/1998/011.
25. Chatterjee S., Karabina K., Menezes A.: Fault attacks on pairing-based protocols revisited. IEEE Trans. Comput. (to appear); available at http://eprint.iacr.org/2014/492.
26. Chatterjee S., Menezes A., Sarkar P.: Another look at tightness. In: Selected Areas in Cryptography—SAC 2011. LNCS, vol. 7118. Springer, Berlin (2012); available at http://anotherlook.ca.
27. Cheon J.: Security analysis of the strong Diffie–Hellman problem. In: Advances in Cryptology—Eurocrypt 2006. LNCS, vol. 4004, pp. 1–11. Springer, Berlin (2006).

28. Cheon J., Han K., Lee C., Ryu, H., Stehlé D.: Cryptanalysis of the multilinear map over the integers. In: Advances in Cryptology—Eurocrypt 2015, Part I. LNCS, vol. 9056, pp. 3–12 . Springer, New York (2015).
29. Coron J.-S., Lepoint T., Tibouchi M.: Practical multilinear maps over the integers. In: Advances in Cryptology—Crypto 2013. LNCS, vol. 8042, pp. 476–493, Springer, Berlin (2013); full version available at http://eprint.iacr.org/2013/183.
30. Coron J.-S., Lepoint T., Tibouchi M.: Cryptanalysis of two candidate fixes of multilinear maps over the integers (2014). Available at http://eprint.iacr.org/2014/975.
31. Coron J.-S., Lepoint T., Tibouchi M.: New multilinear maps over the integers (2015). Available at http://eprint.iacr.org/2015/162.
32. Dang Q.: Randomized hashing for digital signatures, NIST Special Pub. 800–106 (2009). http://csrc.nist.gov/publications/nistpubs/800-106/NIST-SP-800-106.
33. Dodis Y., Oliveira R., Pietrzak K.: On the generic insecurity of the full domain hash. In: Advances in Cryptology—Crypto 2005. LNCS, vol. 3621, pp. 449–466. Springer, Berlin (2005).
34. Fildes J.: iPhone hacker publishes secret Sony PlayStation 3 key, 6 Jan 2011. www.bbc.com/news/technology-12116051.
35. Freire E., Hofheinz D., Paterson K., Striecks C.: Programmable hash functions in the multilinear setting. In: Advances in Cryptology—Crypto 2013. LNCS, vol. 8042, pp. 513–530. Springer, Berlin (2013); full version available at http://eprint.iacr.org/2013/354.
36. Gallant R.: The static Diffie–Hellman problem. In: Presented at ECC (2005). Available at http://cacr.waterloo.ca/conferences/2005/ecc2005/gallant.
37. Gennaro R., Halevi S., Rabin T.: Secure hash-and-sign signatures without the random oracle. In: Advances in Cryptology—Eurocrypt'99. LNCS, vol. 1592, pp. 123–139. Springer, Berlin (1999).
38. Gentry C.: Practical identity-based encryption without random oracles. In: Advances in Cryptology—Eurocrypt 2006. LNCS, vol. 4004, pp. 445–464. Springer, Berlin (2006).
39. Gentry C., Halevi S., Maji H., Sahai A.: Zeroizing without zeroes: cryptanalyzing multilinear maps without encodings of zero (2014). Available at http://eprint.iacr.org/2014/929.
40. Goldreich O.: On post-modern cryptography (2006). http://eprint.iacr.org/2006/461.
41. Goldwasser S., Tauman Kalai Y.: On the (in)security of the Fiat–Shamir paradigm. In: Proceedings of the 44th Annual Symposium Foundations of Computer Science, pp. 102–113. IEEE (2003); full version available at http://eprint.iacr.org/2003/034.
42. Goldwasser S., Micali S., Rivest R.: A paradoxical solution to the signature problem. In: Proceedings of the 25th Annual IEEE Symposium on the Foundations of Computer Science, pp. 441–448 (1984).
43. Green M., Katz J., Malozemoff A., Zhou H.-S.: A unified approach to idealized model separations via indistinguishability obfuscation (2015). Available at: http://eprint.iacr.org/2014/863.
44. Hohenberger S., Sahai A., Waters B.: Replacing a random oracle: full domain hash from indistinguishability obfuscation. In: Advances in Cryptology—Eurocrypt 2014. LNCS, vol. 8441, pp. 201–220. Springer, Berlin (2014).
45. Jao D., Yoshida K.: Boneh–Boyen signatures and the strong Diffie–Hellman problem. In: Pairing-Based Cryptography—Pairing 2009. LNCS, vol. 5671, pp. 1–16. Springer, Berlin (2009); full version available at http://eprint.iacr.org/2009/221.
46. Koblitz N., Menezes A.: Another look at provable security. In: II Progress in Cryptology—Indocrypt 2006. LNCS, vol. 4329, pp. 148–175. Springer, Berlin (2006); available at http://anotherlook.ca.
47. Koblitz N., Menezes A.: Another look at provable security. J. Cryptol. **20**, 3–37 (2007); available at http://anotherlook.ca.
48. Koblitz N., Menezes A.: Another look at generic groups. Adv. Math. Commun. **1**, 13–28 (2007); available at http://anotherlook.ca.
49. Koblitz N., Menezes A.: The brave new world of bodacious assumptions in cryptography, Not. Am. Math. Soc. **57**, 357–365 (2010); available at http://anotherlook.ca.
50. Koblitz N., Menezes A.: Another look at security definitions. Adv. Math. Commun. **7**, 1–38 (2013); available at http://anotherlook.ca.
51. Koblitz N., Menezes A.: Another look at security theorems for 1-key nested MACs. In: Open Problems in Mathematics and Computational Science, pp. 69–89. Springer, Berlin (2014).
52. Lenstra A.K., Hughes J.P., Augier M., Bos J., Kleinjung T., Wachter C.: Public keys. In: Advances in Cryptology—Crypto 2012. LNCS, vol. 7417, pp. 626–642. Springer, Berlin (2012).
53. Lysyanskaya A.: Unique signatures and verifiable random functions from the DH–DDH separation. In: Advances in Cryptology—Crypto 2002. LNCS, vol. 2442, pp. 597–612. Springer, Berlin (2002).
54. Malone-Lee J., Smart N.: Modifications of ECDSA. In: Selected Areas in Cryptography—SAC 2003. LNCS, vol. 2595, pp. 1–12. Springer, Berlin (2002).
55. Menezes A., van Oorschot P., Vanstone S.: Handbook of Applied Cryptography. CRC, Boca Raton (1996).

56. Neven G., Smart N., Warinschi B.: Hash function requirements for Schnorr signatures. J. Math. Cryptol. **3**, 69–87 (2009).
57. Nielsen J.B.: Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. In: Advances in Cryptology—Crypto 2002. LNCS, vol. 2442, pp. 111–126. Springer, Berlin (2002).
58. Nguyen P., Shparlinski I.: The insecurity of the elliptic curve digital signature algorithm with partially known nonces. Des. Codes Cryptogr. **30**, 201–217 (2003).
59. Page D., Vercauteren F.: A fault attack on pairing-based cryptography. IEEE Trans. Comput. **55**, 1075–1080 (2006).
60. Paillier P., Vergnaud D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Advances in Cryptology—Asiacrypt 2005. LNCS, vol. 3788, pp. 1–20. Springer, Berlin (2005).
61. Perlroth N., Larson J., Shane S.: N.S.A. able to foil basic safeguards of privacy on web. The New York Times, 5 Sept 2013.
62. Pointcheval D., Stern J.: Security proofs for signature schemes. In: Advances in Cryptology—Eurocrypt'96. LNCS, vol. 1070, pp. 387–398. Springer, Berlin (1996).
63. Pointcheval D., Stern J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**, 361–396 (2000).
64. Pornin T.: Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA), RFC 6979, IETF, August (2013).
65. Ramchen K., Waters, B.: Fully secure and fast signing from obfuscation. In: Proceedings of ACM CCS'14, pp. 659–673. ACM, New York (2014).
66. Schnorr C.P.: Efficient signature generation for smart cards. J. Cryptol. **4**, 161–174 (1991).
67. Seurin Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: Advances in Cryptology—Eurocrypt 2012. LNCS, vol. 7237, pp. 554–571. Springer, Berlin (2012).
68. Whelan C., Scott M.: The importance of the final exponentiation in pairings when considering fault attacks. In: Pairing-Based Cryptography—Pairing 2007. LNCS, vol. 4575, pp. 225–246. Springer, Berlin (2007).
69. Wigley J.: Removing need for rng in signatures (1997). http://groups.google.com/group/sci.cryp/msg/a6da45bcc8939a89.
70. Zimmerman J.: How to obfuscate programs directly. In: Advances in Cryptology—Eurocrypt 2015, Part II. LNCS, vol. 9057, pp. 439–467. Springer, Berlin (2015).