Lower Bounds on Signatures From Symmetric Primitives

Boaz Barak[*]   Mohammad Mahmoody-Ghidary[†]

January 27, 2008

Abstract

We show that every construction of one-time signature schemes from a random oracle achieves black-box security at most $2^{(1+o(1))q}$, where q is the total number of oracle queries asked by the key generation, signing, and verification algorithms. That is, any such scheme can be broken with probability close to 1 by a (computationally unbounded) adversary making $2^{(1+o(1))q}$ queries to the oracle. This is tight up to a constant factor in the number of queries, since a simple modification of Lamport's one-time signatures (Lamport '79) achieves $2^{(0.812-o(1))q}$ black-box security using q queries to the oracle.

Our result extends (with a loss of a constant factor in the number of queries) also to the random permutation and ideal-cipher oracles. Since the symmetric primitives (e.g. block ciphers, hash functions, and message authentication codes) can be constructed by a constant number of queries to the mentioned oracles, as corollary we get lower bounds on the efficiency of signature schemes from symmetric primitives when the construction is black-box. This can be taken as evidence of an inherent efficiency gap between signature schemes and symmetric primitives.

1   Introduction

Digital signature schemes allow authentication of messages between parties without shared keys. Signature schemes pose an interesting disconnect between the worlds of theoretical and applied cryptography. From a theoretical point of view, it is natural to divide cryptographic tools into those that can be constructed using one-way functions and those that are not known to have such constructions. Signature schemes, along with private key encryption, message authentication codes, pseudorandom generators and functions, belong to the former camp. In contrast, the known constructions of *public key encryption* are based on *structured* problems that are conjectured to be hard (i.e., problems from number theory or the theory of lattices). From a practical point of view, it is more natural to divide the tools according to the *efficiency* of their best known constructions. The division is actually similar, since schemes based on structured problems typically require both more complicated computations and larger key size, as they often have non-trivial attacks (e.g., because of the performance of the best known factoring algorithms, to get 2^n security based on factorization one needs to use $\tilde{\Omega}(n^3)$ bit long integers).

Signature schemes are outlier to this rule: even though they can be constructed using one-way functions, applied cryptographers consider them as relatively inefficient since practical constructions are based on structured hard problems, and thus are significantly less efficient than private key encryption, message authentication codes, pseudorandom functions etc... In particular, very high speed applications shun digital signatures in favor of message authentication codes,[1] even though the latter sometime incur a significant cost in keeping shared private keys among the entities involved (e.g., see [PCST00] and the references therein). The reason is that known constructions of such schemes from one-way functions or other unstructured primitives are quite *inefficient*. This problem already arises in *one-time signatures* [Rab78, Lam79, Mer87], that are a relaxation of digital signatures offering security only in the case that the attacker observes at most a single valid signature. The best known constructions for this case require $\Omega(k)$ invocations of the one-way function (or even a random oracle) to achieve 2^k security. In contrast, there are known constructions of message authentication codes, private key encryptions, and pseudorandom generators and functions that use only $O(1)$ queries to a random oracle.

In this paper, we study the question of whether there exist more efficient constructions of signature schemes from symmetric primitives such as hash functions and block ciphers. We show to a certain extent that the inefficiency of the known constructions is *inherent*.

1.1 Our results

We consider the efficiency of constructions of one-time signatures using black boxes / oracles that model ideal symmetric primitives: the random oracle, the random permutation oracle, and the ideal cipher oracle (see Section 3 for definitions). We wish to study the security of such constructions as a function of the number of queries made to the oracle by the construction (i.e., by the generation, signing, and verification algorithms). Of course, we believe that one-time signatures exist and so there are in fact signature schemes achieving super-polynomial security without making any query to the oracle. Hence we restrict ourselves to bounding the *black-box* security of such schemes. We say that a cryptographic scheme using oracle $\mathcal{O}$ has *black-box security* S if for every $1 \leq T \leq S$, a (potentially computationally unbounded) adversary that makes at most T queries to $\mathcal{O}$ cannot break the scheme with probability larger than T/S (see Definition 3.6). Our main result is the following:

Theorem 1.1. *Any one-time signature scheme for n-bit messages using at most $q \leq n$ queries to a random oracle has black-box security at most $2^{(1+o(1))q}$ where $o(1)$ goes to zero with q.*

This is in contrast to other primitives such as message authentication codes, collision resistant hash functions, private-key encryption, and pseudorandom functions, that can all be implemented using one or two queries to a random oracle with black-box security that depends exponentially on the length of these queries. We note that Theorem 1.1 is tight up to a constant factor in the number of queries, since a simple modification of Lamport's scheme [Lam79] yields $2^{(\alpha-o(1))q}$ black-box security, where $\alpha \sim 0.812$ is equal to $H(c)/(1+c)$, where H is the Shannon entropy function and $c = (3 - \sqrt{5})/2$ (see Section 5). We also prove several extensions of the main result:

Other oracles. Since our goal is to find out whether signatures can be efficiently constructed from symmetric primitives, it makes sense to study also other primitives than the random

---

[1]In contrast to digital signatures that have a public verification key and secret signing key, *message authentication codes* have a single key for both verification and signing, and hence that key must be kept private to maintain security.

oracle. Theorem 1.1 extends (with a loss of a constant factor in the number of queries) to the *ideal cipher oracle* and *random permutation oracle* that are also sometimes used to model the idealized security of symmetric primitives such as block ciphers and one-way permutations.

Implementing adversary in BPP$^{\mathbf{NP}}$. The proof of Theorem 1.1 shows that for every q-query one-time signature scheme for $\{0,1\}^n$ from random oracle, there is an adversary that breaks it with probability close to 1 using at most $\mathrm{poly}(q)2^q$ queries. However, the *running time* of this adversary can be higher than that. This is inherent, as otherwise we would be proving unconditionally the non-existence of one-time signature schemes. However, we show that this adversary can be implemented in probabilistic polynomial-time using an oracle to an **NP**-complete problem. Thus, similar to what Impagliazzo and Rudich [IR89] showed for key-exchange, if there were a more efficient construction of signature schemes from random oracles with a proof of security relying on the adversary's efficiency, then this is also a proof that $\mathbf{P} \neq \mathbf{NP}$.

Imperfect completeness. While the standard definition of signature schemes requires the verifier to accept valid signatures with probability 1, one can also consider relaxed variants where the verifier has some small positive probability of rejecting even valid signatures. We say that such signature schemes satisfy *imperfect completeness*. We can extend Theorem 1.1 to this case, though to get an attack succeeding with high probability we lose a quadratic factor in the number of queries.

Efficiency of the verifier. Because the signing and the verification algorithms are executed more often than the key generation algorithm, it makes sense to study their efficiency separately rather than just studying the total number of queries. Although in the construction for signature schemes that we will see later (see Section 5), the signing algorithm asks only one oracle query and the total number of queries is optimal up to a constant factor, the question about the efficiency of the verifier still remains. We show that (keeping the number of signing queries fixed to one) there is a tradeoff between the number of queries asked by the verification algorithm and the total number of queries, conditioned on getting certain black-box security.

Black-box constructions. As mentioned above, all the symmetric primitives can be constructed from random oracle, random permutation oracle, or ideal cipher oracle by only $O(1)$ queries and get exponential security over the length of the queries. Therefore, our lower bounds on signatures from ideal oracles yield as corollaries lower bounds on the efficiency of signatures from symmetric primitives when the construction is black box. This holds even when the one-way permutation used in the construction has $n/2$ hardcore bits. The latter answers a question raised by [GGKT05]. Our results reject the existence of black-box constructions unconditionally (similar to [HHRS07], while the results of [GGKT05] show the existence of one-way function as a consequence. We prove the strongest possible form of lower bound on the efficiency of black box constructions of signatures from symmetric primitives. Namely, we show that black-box constructions of signature schemes for n-bit messages based on exponentially hard symmetric primitives of security parameter n, need to make at least $\Omega(n)$ calls to the primitive.

Note on the random oracle model. Although the *random oracle model* [BR93] (and its cousin the *ideal cipher model*) is frequently used as an idealization of the properties enjoyed by certain

constructions such as the SHA-1 hash function [Nat95] and the AES block cipher [DR02], it has drawn a lot of criticism as this idealization is not generally justified [CGH98]. However, for the sake of *lower bounds* (as is our concern here) this idealization seems appropriate, as it is a clean way to encapsulate all the attractive properties that could be obtained by constructions such as SHA-1,AES, etc..

Taxonomy of black-box reductions. Reingold, Trevisan and Vadhan [RTV04] study various notions of "black-boxness" of security proofs in cryptography according to whether a construction of a cryptographic tool based on an underlying primitive uses this primitive as a black box, and whether its security proof uses the adversary as a black box. Those definitions are not in the oracle model that we are concerned here. They call a construction for primitive A from primitive B black-box, if the implementation of A uses B as a black box. The security reduction which converts an adversary for the implementation of A to an adversary for B could have different levels of being black box [2]. However, in the oracle based constructions studied here, the implementation reduction is always forced to be black-box, and for the proof of security, there is no security measure defined for the primitive used (i.e. the oracle) to which we could reduce the security of our construction. One common way to prove security for oracle based constructions is to rely on the statistical properties of the oracle and show that any (even computationally unbounded) adversary breaking the implementation needs to ask *many* queries from the oracle. This gives a quantitative security guarantee and is called a *black-box* proof of security in the oracle model. A *non-black-box* proof of security in this model, is a proof showing that any adversary who runs in time $\text{poly}(n, T)$ where n is the input length and T the number of oracle queries it asks, needs to ask many queries from the oracle. In this work, we give a lower bound on the number of queries needed to get *black-box* security S for one-time signatures in various ideal oracle models, and also show that if $\mathbf{P} = \mathbf{NP}$, then this bound holds for *non-black-box* proofs of security as well. We note that if one-way functions exist, then there do exist constructions making no query to the random oracle with super-polynomial non-black-box security. As we mentioned before, our lower bounds in the ideal oracle models yield some lower bounds on the efficiency of one-time signatures from symmetric primitives in the standard model of [RTV04]. We also note that there do exist cryptographic constructions that use the primitive [GMW86, GMW87] or the adversary [Bar01] in a non-black-box way, but at the moment all of the known highly efficient cryptographic constructions (e.g., those used in practice) are black box, in the sense that if they use a generic underlying primitive (i.e., not based on specific problems such as factoring) then it's used as a black-box and if they have a proof of security then the proof treats the adversary as a black box.

1.2 Prior work

To the best of our knowledge, this is the first lower bound on the number of random oracle queries needed to construct signature schemes. Starting with the seminal paper of Impagliazzo and Rudich [IR89], that showed that there is no construction of a key exchange protocol from a random oracle with super-polynomial black-box security, and therefore rejecting black-box constructions of key exchange protocols from one-way function, several works have investigated the *existence* of black-box constructions reducing one kind of cryptographic scheme to another. However, only few works

---

[2]It could be fully black-box, semi black-box, or non-black-box, and if the implementation reduction is black box, the whole construction is called, (resp.) fully black-box, semi black-box, or weakly black-box.

studied the *efficiency* of such constructions [KST99, GGKT05]. Of these, the most relevant is the paper by Gennaro, Gertner, Katz, and Trevisan [GGKT05]. They considered the efficiency of basing various cryptographic primitives on one-way permutations (OWP) secure against S-sized circuits, and proved that to achieve super-polynomial security **(1)** pseudorandom generators with ℓ bits of stretch require $\Omega(\ell/\log S)$ invocations of the OWP, **(2)** universal one-way hash functions compressing their input by ℓ bits require $\Omega(\ell/\log S)$ invocations, **(3)** private key encryption schemes for messages of length n with key length k require $\Omega((n-k)/\log S)$ invocations, and (most relevant for us) **(4)** one-time signature schemes for n-bit messages require $\Omega(n/\log S)$ invocations.[3]

However, the one-way permutation oracle used by [GGKT05] was very far from being a random oracle.[4] Indeed, the applications **(1)**, **(2)**, and **(3)** can be implemented using only a constant number of calls to a random oracle, and correspondingly are considered to have efficient practical implementations. Thus, [GGKT05] did not answer the question of whether signature schemes can be efficiently constructed from efficient symmetric key primitives such as hash functions and block ciphers. It is this question that we are concerned with in this paper. Thus, on a technical level our work is quite different from [GGKT] (as we work with a random oracle and cannot "tamper" with it to prove our lower bound) and in fact is more similar to the techniques in the original work of Impagliazzo and Rudich [IR89]. We note that this work partially answers a question of [GGKT05], as it implies that any black-box construction of one-time signatures from one-way permutation $p : \{0,1\}^n \mapsto \{0,1\}^n$ with even $n/2$ hard-core bits requires at least $\Omega(n)$ queries to the permutation.

Several works [Mer87, EGM89, Vau92, BM94, BM96] considered generalizations of Lamport's one-time signature scheme. Some of these achieve shorter keys and signatures, although their relation between the number of queries and security (up to a constant factor) is at most a constant factor better than Lamport's scheme (as we show is inherent).

2 Our techniques

We now give a high level overview of the ideas behind the proof of Theorem 1.1. Our description ignores several subtle issues, and the reader is referred to Section 4 for the full proof. To understand the proof of the lower bound,[5] it is instructive to review the known *upper bounds* and in particular the simple one-time signature scheme of Lamport [Lam79]. To sign messages of length n with security parameter ℓ using a random oracle $\mathcal{O}$ (that we model as a random function from $\{0,1\}^\ell$ to $\{0,1\}^\ell$) the scheme works as follows:

- Generate the public verification key VK by choosing $2n$ random strings $\{x_i^b\}_{i\in[n],b\in\{0,1\}}$ in $\{0,1\}^\ell$ and setting VK to be the sequence $\{y_i^b\}_{i\in[n],b\in\{0,1\}}$ for $y_i^b = \mathcal{O}(x_i^b)$.

- To sign a message $\alpha \in \{0,1\}^n$, simply reveal the preimages in the set $\{x_i^b\}_{i\in[n],b\in\{0,1\}}$ that correspond to the bits of α. That is, the signature is $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}$.

- The verifier checks that indeed $\mathcal{O}(x_i^{\alpha_i}) = y_i^{\alpha_i}$ for every $i \in [n]$.

---

[3]Otherwise, we can construct a one-way function directly.

[4]They considered an oracle that applies a random permutation on the first t bits of its n-bit input, for $t \ll n$, and leaves the rest of the $n - t$ bits unchanged. This is a one-way permutation with $2^{\Omega(t)}$ security.

[5]We use the terms "lower bound" and "upper bound" in their traditional crypto/complexity meaning of negative results vs. positive results. Of course one can view Theorem 1.1 as either upper-bounding the security or lower-bounding the number of queries.

This scheme uses $3n$ queries. It can be shown that it has $2^{\Omega(\ell)}$ security. Note that in this case the security can be arbitrarily large *independently* of the number of queries. Indeed, note that Theorem 1.1 requires that the number of queries q is not larger than the length of the messages to be signed. Lamport's scheme can be easily modified to work for unbounded size messages by following the well known "hash-and-sign" paradigm: first use the random oracle to hash the message to length k, and then apply Lamport's scheme to the hashed value. This will result in a scheme with $3k + 2$ queries and (by the birthday bound) $2^{k/2}$ black-box security (see Section 5 for some improvements). We see that now indeed the security is bounded by $2^{O(q)}$ (where $q = 3k + 2$ is the number of queries), regardless of the length ℓ of the queries.

The above discussion shows that to prove Theorem 1.1, we will need to use the fact that there is a large number of potential messages, which is indeed what we do. Note that the reason that the hash-and-sign variant of Lamport's scheme only achieves $2^{k/2}$ security is that if a pair of messages α, β satisfies $\mathcal{O}_k(\alpha) = \mathcal{O}_k(\beta)$ (where $\mathcal{O}_k(x)$ denotes the first k bits of $\mathcal{O}(x)$), then they have the same signature, and so a signature for α allows an adversary to forge a signature on β. We will try to generalize this observation to arbitrary signature schemes. For every such scheme $\mathcal{S}$ and two messages α, β (after fixing the oracle and the randomness of the system), we will say that "α is useful for β" if they satisfy a certain condition. Then (roughly speaking) we will prove that: **(A)** if α is useful for β then a signature on α can be used to compute a signature on β by asking at most $2^{O(q)}$ oracle queries (where q is the total number of queries made by the scheme $\mathcal{S}$), and **(B)** if α and β are chosen at random from a large enough space of messages, then α will be useful for β with probability at least $2^{-O(q)}$. Together **(A)** and **(B)** imply that, as long as the space of possible messages is large enough, then the black-box security of $\mathcal{S}$ is bounded by $2^{O(q)}$, since the adversary can find a useful pair of messages α, β with probability 2^{-q}, ask for a signature on α and use that to forge a signature on β by asking 2^q queries.[6]

DEFINING THE USEFULNESS CONDITION. This proof strategy rests of course on the ability to find an appropriate condition "α is useful for β" for every one-time signature scheme $\mathcal{S}$. This is what we describe now. For now, we will assume that only the key generation algorithm of $\mathcal{S}$ is probabilistic, and that both the signing and verification algorithms are deterministic.[7] For every fixed randomness for the generation algorithm, fixed oracle, and a message α, we define G, S_α and V_α to be the sets of queries (resp.) made by the generation, signing, and verification algorithms where the last two are applied on the message α.

First attempt. Observe that in the hash-and-sign variant of Lamport's scheme, α and β have the same signature if $V_\alpha = V_\beta$. This motivates stipulating for every signature scheme that α is useful for β if $V_\beta \subseteq V_\alpha$. This definition satisfies Property **(A)** above: if we know all the queries that the verifier will make on a signature of β, then finding a signature that makes it accept can be done by an exponential-time exhaustive search that does not make any oracle queries at all. The problem is that it might not satisfy **(B)**: it's easy to make the verifier ask, when verifying a signature for α, a query that uniquely depends on α, thus ensuring $V_\beta \nsubseteq V_\alpha$ for every distinct α, β.

---

[6]The actual adversary we'll show will operate by asking $\text{poly}(q)2^q$ queries, and it succeeds with probability almost 1, see the proof of Theorem 4.1.

[7]We study the randomized verifier in Section 6.3, but assuming that the signer is deterministic is without loss of generality. That is because the key generator can give, through the secret key, a secret seed s to the signer, and the signer would use $\mathcal{O}(s, \alpha)$ as the randomness needed to sign the message α.

Second attempt. A natural intuition is that verifier queries that do not correspond to queries made by the generation algorithm are sort of "irrelevant"— after all, in Lamport's scheme all the queries the verifier makes are a subset of the queries made by the generation algorithm. Thus, we might try to define that α is useful for β if $V_\beta \cap G \subseteq V_\alpha$. Since G has at most q queries, and so at most 2^q subsets, this definition satisfies Property **(B)** since if α and β are randomly chosen from a set of size 2^q then α will be useful for β with probability at least 2^{-2q}. Unfortunately, it does not satisfy Property **(A)**: there is a signature scheme for which every pair of messages α, β satisfies this condition even when a signature for α cannot be used to forge a signature on β.[8]

Our actual condition. The condition we actually use, roughly speaking, is that α is useful for β if

$$V_\beta \cap (G \cup S_\alpha) \subseteq V_\alpha \ . \tag{1}$$

Using Bollobás's Inequality [Bol65] (see the proof of Claim 4.7) it can be shown that the condition (1) satisfies Property **(B)**. It's less obvious why it satisfies Property **(A)**— to see this we need to see how our adversary will operate. The high level description of our attack is as follows:

1. **Input: Key Generation.** The adversary receives the verification key VK.

2. **Request Signature.** Choose $\alpha \neq \beta \leftarrow_R \{0,1\}^n$ at random, and get σ_α, the signature of α.

3. **Learning Oracle Queries.** Run $\mathsf{Ver}(VK, \alpha, \sigma_\alpha)$ to learn the set V_α of oracle queries that it asks and their answers. (Later we will modify this step somewhat, and ask some more oracle queries.)

4. **Sampling a Possible Transcript.** Conditioned on knowing VK, σ_α, and answers of V_α, *guess*: the value of SK, the sets G and S_α, and their answers. Let $\tilde{SK}$, $\tilde{G}$, and $\tilde{S}_\alpha$ be the guesses.

5. **Forging.** Sign the message β by using $\tilde{SK}$ and sticking to the oracle answers guessed for queries in $\tilde{G} \cup \tilde{S}_\alpha$ to get σ_β. That is, if we wanted to ask a an oracle query in $\tilde{G} \cup \tilde{S}_\alpha$, use the guessed answer, and otherwise ask the real oracle $\mathcal{O}$. Output σ_β.

Note that the queries for which we might have guessed a wrong answer are in the set $(\tilde{G} \cup \tilde{S}_\alpha) \backslash V_\alpha$, because we did the guesses conditioned on knowing V_α and its answers. Suppose that during the verification of (β, σ_β), none of these queries is asked from the oracle (i.e. $V_\beta \cap (\tilde{G} \cup \tilde{S}_\alpha) \subset V_\alpha$). Then we can *pretend* that our guesses were correct. That is, because the answers to different queries of random oracle are independent, as far as the verifier is concerned our guesses could be right, and hence by definition, the verification of (β, σ_β) must accept with probability 1.

The description of the attack above shows that a similar condition to the condition (1), namely

$$V_\beta \cap (\tilde{G} \cup \tilde{S}_\alpha) \subset V_\alpha \ , \tag{2}$$

has Property **(A)**. But condition (2) might not have Property **(B)**. We cope with this by ensuring that the attacker has sufficient information so that (essentially) whenever (1) happens, (2) also

---

[8]Such an example can be obtained by the variant of Lamport's scheme where each signer uses the verification key VK to sign a new verification key VK' (the randomness for which is part of the secret key), and then signs the message using the secret key corresponding to VK'. In this case $V_\alpha \cap G = V_\beta \cap G$ for every pair α, β, even if a signature on α cannot be used to compute a signature on β.

happens. This is accomplished by learning more oracle queries before making the guesses. Namely, we learn all the queries that are in the set $\tilde{G} \cup \tilde{S}_\alpha$ with some noticeable probability (conditioned on what we know about them). We then use a careful hybrid argument (that involves the most technical part of the proof) to show that after performing this learning, the condition (2) occurs with probability at least as large as the probability that (2) occurs (up to some lower order terms). Thus our actual usefulness condition will be (2), though for the complete definition of the sets $\tilde{G}, \tilde{S}_\alpha$ involved in it, one needs to go into the details of the proof of Theorem 4.1).

3   Preliminaries

3.1   Basic Probability Facts

We recall some simple but useful well known facts and definitions about random variables.

Definition 3.1. The *statistical distance* of two finite random variables X, Y, denoted by $\mathsf{SD}(X, Y)$, is defined to be $\frac{1}{2} \sum_a |\Pr[X = a] - \Pr[Y = a]|$.

Lemma 3.2. *If A, B are random variables, and the event E is defined over $\mathsf{Supp}(A) \cup \mathsf{Supp}(B)$ (where $\mathsf{Supp}(X)$ denotes the support of the random variable X), then $|\Pr[E(A)] - \Pr[E(B)]| \le \mathsf{SD}(A, B)$.*

Lemma 3.3. *If the random variable A' is a function of random variable A, and the random variable B' is a function of B, then $\mathsf{SD}(A', B') \le \mathsf{SD}(A, B)$.*

Lemma 3.4. *If the event E is defined over the random variable A, and the event D is defined over the random variable B, and we have $\mathsf{SD}(A \mid E, B \mid D) = 0$, then $\mathsf{SD}(A, B) \le (\Pr[E] + \Pr[D])/2$.*

By U_n we mean the uniformly distributed random variable over n-bit strings.

3.2   Signature Schemes in Oracle Models

We define the notion of one-time signature schemes and their black-box security. We specialize our definition to the case that the signature schemes use an oracle $\mathcal{O}$ that may also be chosen from some probability distribution. We use the standard notation $A^{\mathcal{O}}(x)$ to denote the output of an algorithm A on input x with access to oracle $\mathcal{O}$.

Definition 3.5. An oracle *signature scheme* (with perfect completeness) for n bit messages is a triple of oracle algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ (where Gen could be probabilistic) with the following property: for every oracle $\mathcal{O}$, if (SK, VK) is a pair that is output by $\mathsf{Gen}^{\mathcal{O}}(1^n)$ with positive probability, then for every $\alpha \in \{0, 1\}^n$, $\mathsf{Ver}^{\mathcal{O}}(VK, \alpha, \mathsf{Sign}^{\mathcal{O}}(SK, \alpha)) = 1$. We call SK the *signing key* and VK the *verification key*.

One can also make a relaxed requirement that the verification algorithm only needs to accept valid signatures with probability 0.9 (where this probability is over the verifier's coins only). We say that such relaxed signature schemes have *imperfect completeness*, and we will consider such schemes in Section 6.3. If the oracle algorithms of the Definition 3.5 run in polynomial-time, then we call the signature scheme *efficient*. Note that we consider (not necessarily efficient) signature algorithms on a finite set of messages. For upper bounds (i.e., positive results) one would want

uniform efficient algorithms that could handle any size of message, but for a lower bound (i.e., a negative result), this simpler definition will do.

So far, we did not say anything about the security. In the following definition we specify the "game" in which the adversary participates and tries to break the system and give a quantitative measure for the security.

Definition 3.6. For every $S \in \mathbb{N}$, the oracle signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is a *one-time* signature scheme with *black-box security* S, if for every message $\alpha \in \{0,1\}^n$, $1 \leq T \leq S$, and adversary algorithm A that makes at most T queries to its oracle, $\Pr[\mathsf{Ver}(VK, \alpha^*, \sigma^*) = 1$ where $(\alpha^*, \sigma^*) = A^{\mathcal{O}}(VK, \mathsf{Sign}^{\mathcal{O}}(SK, \alpha))$ and $\alpha^* \neq \alpha] \leq \frac{T}{S}$, where $(SK, VK) = \mathsf{Gen}^{\mathcal{O}}(1^n)$, and this probability is over the coins of all algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}$, and $A)$, and the choice of the oracle $\mathcal{O}$.

This is a slightly weaker definition of security than the standard definition, since we are not allowing the adversary to choose the message α based on the public key. However, this is again fine for lower bounds (the known upper bounds do satisfy the stronger definition). Also, some texts use $1/S$ (rather than T/S) as the bound on the success probability. Security according to either one of these definitions is always at most quadratically related, but we feel Definition 3.6 is more precise.

In a *non-black-box* proof of security, the running time of the adversary is utilized in order to prove the security of the system:

Definition 3.7. For every $S \in \mathbb{N}$, the oracle signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is a one-time signature scheme with *non-black-box* security S, if for every message $\alpha \in \{0,1\}^n$, $T \leq S$, and adversary algorithm A_T that makes at most T oracle queries and runs in time $\mathrm{poly}(n, T)$, $\Pr[\mathsf{Ver}(VK, \alpha^*, \sigma^*) = 1$ where $(\alpha^*, \sigma^*) = A_T^{\mathcal{O}}(VK, \mathsf{Sign}^{\mathcal{O}}(SK, \alpha))$ and $\alpha^* \neq \alpha] \leq \frac{T}{S}$, where $(SK, VK) = \mathsf{Gen}^{\mathcal{O}}(1^n)$, and this probability is over the coins of all algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}$, and $A_T)$, and the choice of the oracle $\mathcal{O}$.

Oracles. In this work, as for the oracle signature schemes, we only use one of the following oracles: **(1)** The *random oracle* returns on input $x \in \{0,1\}^n$ the value $f(x)$ where f is a random function from $\{0,1\}^n$ to $\{0,1\}^n$.[9] **(2)** The *random permutation oracle* returns on input $x \in \{0,1\}^n$ the value $f(x)$ where f is a random permutation on $\{0,1\}^n$. **(3)** The *ideal cipher oracle* with message length n, returns on input (k, x, d) where $k \in \{0,1\}^*$, $x \in \{0,1\}^n$ and $d \in \{\mathsf{F}, \mathsf{B}\}$, $f_k(x)$ if $d = \mathsf{F}$ and $f_k^{-1}(x)$ if $d = \mathsf{B}$, where for every $k \in \{0,1\}^*$, f_k is a random permutation on $\{0,1\}^n$. These three oracles are standard idealizations of (respectively) hash functions, one-way permutations, and block ciphers (see also Section 7).

4   Proof of the main result

Theorem 4.1. *Let* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a one-time oracle signature scheme (with perfect completeness) in random oracle model for the space of messages* $\mathcal{M}$ *in which the total number of oracle queries asked by* $\mathsf{Gen}, \mathsf{Sign}$, *and* Ver *is at most* q, *and* $|\mathcal{M}| \geq \frac{\binom{q}{q/2}}{\lambda}$. *Then there is a (computationally unbounded) adversary which asks at most* $O(\frac{q^2 \binom{q}{q/2}}{\lambda \delta^2}) = O(\frac{q^{1.5} 2^q}{\lambda \delta^2})$ *oracle queries and breaks the scheme with probability* $1 - (\lambda + \delta)$. *This probability is over the randomness of the oracle as well as the coin tosses of the key generation algorithm and the adversary.*

---

[9]More generally, f can be a function from n to $\ell(n)$ for some function $\ell : \mathbb{N} \to \mathbb{N}$, but using standard padding arguments we may assume $\ell(n) = n$.

Theorem 4.1 implies Theorem 1.1 via the following corollary:

Corollary 4.2. *Let* (Gen, Sign, Ver) *be a one-time oracle signature for the messages* $\mathcal{M} = \{0,1\}^n$ *in the random oracle model in which the total queries asked by the scheme is at most q where $q \leq n$, then there is an adversary asking $2^{(1+o(1))q}$ queries breaking the scheme with probability at least $1 - o(1)$ and at least 0.49 for any $q \geq 1$.*

Proof. Let $\delta = \lambda = \binom{q}{q/2}/2^q = \theta(q^{-1/2}) = o(1)$, so we have $|\mathcal{M}| = 2^n \geq 2^q = \binom{q}{q/2}/\lambda$. Therefore we get an adversary asking $O(q^{3.5}\binom{q}{q/2}) = O(q^3 2^q) = 2^{(1+o(1))q}$ queries breaking the scheme with probability $1 - o(1)$. Thus the black-box security of the scheme is at most by $\frac{2^{(1+o(1))q}}{1-o(1)} = 2^{(1+o(1))q}$. For any $q \geq 1$, λ can be as small as $\binom{1}{0}/2^1 = 1/2$, and by taking $\delta = 0.01$ the success probability will be at least 0.49. $\square$

We now turn to proving Theorem 4.1. Let (Gen, Sign, Ver) be as in the theorem's statement. We assume that only Gen is probabilistic, and Sign and Ver are deterministic. We also assume that all the oracle queries are of length ℓ. Since we assume the signature has perfect completeness, these assumptions can be easily shown to be without loss of generality. (In the case of imperfect completeness the verifier algorithm is inherently probabilistic; this case is studied in Section 6.3.) We will show an adversary that breaks the signature system with probability $1 - (\lambda + O(\delta))$, which implies Theorem 4.1 by simply changing δ to δ/c for some constant c.

THE ADVERSARY'S ALGORITHM. Our adversary Adv will operate as follows:

Input: Key generation. The adversary receives a verification key VK, where $(VK, SK) = $ Gen(1^n).

Step 1: Request signature. Let $\beta_0, \ldots, \beta_{N-1}$ denote the first $N = \frac{\binom{q}{q/2}}{\lambda}$ distinct messages (in lexicographic order) in $\mathcal{M}$. Let $\alpha_0, \ldots, \alpha_{N-1}$ be a random permutation of $\beta_0, \ldots, \beta_{N-1}$. Adv asks for a signature on α_0 and verifies it (note that α_0 is chosen independently of the public key). We denote the obtained signature by σ_0, and we denote by T_0 the *transcript* of the algorithms run so far, which includes the random tape of the key generation algorithm, all the queries made by the key generation, signing, and verification algorithms, and the answers to these queries. So T_0 completely describes the running of the algorithms so far. (Note that Adv only has partial information on T_0.)

Step 2: Learning query/answer pairs. We denote by L_0 the information that Adv currently has on the oracle $\mathcal{O}$ and the randomness of the generation algorithm: that is, L_0 consists of VK, σ_0 and the queries made by the verifying algorithm Ver on input VK, σ_0, along with the answers to these queries. Let $\epsilon = \frac{\delta}{qN}$, and $M = \frac{q}{\epsilon\delta} = \frac{q^2 N}{\delta^2}$. For $i = 1, \ldots, M$, do the following:

1. Let $\mathbf{D}_{i-1}$ be the distribution of T_0, the transcript of the first step, conditioned on only knowing L_{i-1}.

2. We let $Q(L_{i-1})$ denote the queries appearing in L_{i-1}. If there exists a string $x \in \{0,1\}^\ell \setminus Q(L_{i-1})$ that is queried with probability at least ϵ in $\mathbf{D}_i$, then Adv lets L_i be L_{i-1} concatenated with the query/answer pair $(x_i, \mathcal{O}(x_i))$, where x_i is the lexicographically first such string. Otherwise, $L_i = L_{i-1}$.

Step 3: Sampling a possible transcript. Adv generates a random transcript $\tilde{T}_0$ according to the distribution $\mathbf{D}_M$. Note that $\tilde{T}_0$ also determines a secret signing key, which we denote by $\tilde{SK}$ ($\tilde{SK}$ may or may not equal the "true" signing key SK). $\tilde{T}_0$ may also determine some query/answer pairs that were not in L_M, and hence may not agree with the the actual answers of the "true" oracle $\mathcal{O}$. We denote by $\tilde{\mathcal{O}}$ the oracle that on input x, if x appears as a query in $\tilde{T}_0$ then $\tilde{\mathcal{O}}(x)$ outputs the corresponding answer, and otherwise $\tilde{\mathcal{O}}(x) = \mathcal{O}(x)$.

Step 4: Forging. For every $j = 1, \ldots, N-1$, Adv uses $\tilde{SK}$ and the oracle $\tilde{\mathcal{O}}$ to compute a signature on the message α_j, which it then tries to verify this time using VK and the "true" oracle $\mathcal{O}$. Adv outputs the first signature that passes verification.

ANALYSIS. The number of queries asked during the attack is at most $M + qN = \frac{q^2 N}{\delta^2} + qN \leq \frac{2q^2 N}{\delta^2} = O(\frac{q^2\binom{q}{q/2}}{\lambda\delta^2})$. To analyze the success probability of Adv we will prove the following lemma:

Lemma 4.3. *For every $j \in [0..N-1]$, let V_j denote the set of queries made by Adv when verifying the signature on α_j. Let $\tilde{G}$ and $\tilde{S}_0$ be the sets of queries made by the generation and signing algorithms according to the transcript $\tilde{T}_0$. For every $j \geq 1$, let E_j be the event that $V_j \cap (\tilde{G} \cup \tilde{S}_0) \subseteq V_0$. Then,*

$$\Pr[\cup_{j\in[1..N-1]}E_j] = 1 - (\lambda + 2\delta) \quad .$$

Note that the event E_j corresponds to the condition that "α_0 is useful for α_j" described in Section 2. Lemma 4.3 implies Theorem 4.1 since if the event E_j holds then when verifying the signature for α_j, the verifier never asks a query on which the oracles $\mathcal{O}$ and $\tilde{\mathcal{O}}$ differ (these oracles can differ only on queries in $(\tilde{G} \cup \tilde{S}_0) \setminus V_0$). But if the verifier uses the same oracle $\tilde{\mathcal{O}}$ used by the generation and signing algorithm, then by the definition of a signature scheme, it must accept the signature.

4.1 Proof of Lemma 4.3

It turns out that using known combinatorial techniques, one can show that $\cup_j E_j$ holds with high probability if all signatures and verifications were to use the "true" oracle $\mathcal{O}$ and signing key SK (as opposed to $\tilde{O}$ and $\tilde{SK}$). The idea behind the proof is to show this holds in our case using a hybrid argument. Specifically, we define four distributions $\mathbf{H}^0, \mathbf{H}^1, \mathbf{H}^2, \mathbf{H}^3$, where $\mathbf{H}^0$ corresponds to $\tilde{T}_0$ joint with all the oracle queries/answers that the adversary gets during the signing and verification algorithms on α_j for $j \geq 1$ (we call this information the *transcript* of the experiment), and $\mathbf{H}^3$ corresponds to T_0 (the real transcript of the first step) joint with the rest of the system's transcript if we use the "true" oracle and signing key (so the adversary is not doing anything in generating $\mathbf{H}^3$). We will prove the lemma by showing that the probability of $\cup_j E_j$ is almost the same in all these four distributions.

Definition of hybrid distributions. The four hybrid distributions $\mathbf{H}^0, .., \mathbf{H}^3$ are defined as follows:

$\mathbf{H}^0$**:** This is the distribution of $\tilde{T}_0, T_1, \ldots, T_{N-1}$, where $\tilde{T}_0$ denotes the transcript sampled by Adv in Step 3, while T_j (for $j \geq 1$) denotes the transcript of the jth signature (i.e., the queries and answers of the signing and verification algorithms on α_j) as generated by Adv in Step 4. Note that T_0 and $\tilde{T}_0$ describe also the running of the key generation while T_j for $j \geq 1$ do not.

11

$\mathbf{H}^1$: This is the same distribution as $\mathbf{H}^0$, except that now in Step 4 of the attack, the adversary uses the modified oracle $\tilde{\mathcal{O}}$ for both signing and verifying the signatures on $\alpha_1, \ldots, \alpha_{N-1}$ (recall that in $\mathbf{H}^0$ the oracle $\tilde{\mathcal{O}}$ is only used for signing).

$\mathbf{H}^2$: This is the same distribution as $\mathbf{H}^1$, except that we make a slight modification in the definition of $\tilde{\mathcal{O}}$: for every query x that was asked by the generation, signing, and verification algorithms in the Input step and Step 1 (i.e., for every query in T_0), we answer with $\mathcal{O}(x)$ *only* if x also appears in L_M. Otherwise, we answer this query with a completely random value. Note that all the queries of the verification are in L_0 and so in L_M as well. In other words, $\tilde{\mathcal{O}}$ agrees with $\mathcal{O}$ on all the queries that Adv has asked from $\mathcal{O}$ till the end of Step 2, and all the others are answered completely at random.

$\mathbf{H}^3$: This is the same distribution as the previous ones, with the difference that $\tilde{T}_0$ is chosen equal to T_0 (and so, there is no point in neither Step 2 of the attack nor defining $\tilde{\mathcal{O}}$ anymore). In other words, this is the transcript (randomness and all query/answer pairs) of the following experiment: **(1)** Generate signing and verification keys (SK, VK) using a random oracle $\mathcal{O}$ **(2)** for $j = 0 \ldots N - 1$, sign α_j and verify the signature using SK, VK and $\mathcal{O}$.

Note that the hybrid distributions $\mathbf{H}^i$ are over the coin tosses of the oracle, the key generation algorithm, and the adversary. Lemma 4.3 follows immediately from the following claims:

Claim 4.4. $\Pr_{\mathbf{H}^0}[\cup_{j \geq 1} E_j] = \Pr_{\mathbf{H}^1}[\cup_{j \geq 1} E_j]$.

Claim 4.5. $\mathsf{SD}(\mathbf{H}^1, \mathbf{H}^2) \leq 2\delta$. *Thus,* $\Pr_{\mathbf{H}^1}[\cup_{j \geq 1} E_j] \geq \Pr_{\mathbf{H}^2}[\cup_{j \geq 1} E_j] - 2\delta$.

Claim 4.6. $\mathbf{H}^2 \equiv \mathbf{H}^3$. *Thus,* $\Pr_{\mathbf{H}^2}[\cup_{j \geq 1} E_j] = \Pr_{\mathbf{H}^3}[\cup_{j \geq 1} E_j]$.

Claim 4.7. $\Pr_{\mathbf{H}^3}[\cup_{j \geq 1} E_j] \geq 1 - \lambda$.

4.2 Proof of Claims 4.4 to 4.7

We now complete the proof of Lemma 4.3 by proving Claims 4.4 to 4.7.

Claim 4.4 (Restated). $\Pr_{\mathbf{H}^0}[\cup_{j \geq 1} E_j] = \Pr_{\mathbf{H}^1}[\cup_{j \geq 1} E_j]$.

Proof. Suppose we sample the hybrid distributions $\mathbf{H}^0$ and $\mathbf{H}^1$ using the same oracle $\mathcal{O}$, same randomness for key generation, and the same randomness for the adversary. Then it is easy to see that for any j, the event E_j holds for $\mathbf{H}^0$ iff it holds for $\mathbf{H}^1$ and so is the event $\cup_{j \geq 1} E_j$. This shows that the probability of $\cup_{j \geq 1} E_j$ happening in both distributions is the same. $\square$

Claim 4.5 (Restated). $\mathsf{SD}(\mathbf{H}^1, \mathbf{H}^2) \leq 2\delta$. *Thus,* $\Pr_{\mathbf{H}^1}[\cup_{j \geq 1} E_j] \geq \Pr_{\mathbf{H}^2}[\cup_{j \geq 1} E_j] - 2\delta$.

Proof. Let B be the event that Adv asks a query in $Q(T_0) \setminus Q(L_M)$, where $Q(T_0)$ denotes the queries in the transcript T_0. It is easy to see that conditioned on B doesn't happen $\mathbf{H}^1$ and $\mathbf{H}^2$ are identically distributed. That is because if we use the same randomness for key generation, oracle and the adversary in the sampling of $\mathbf{H}^1$ and $\mathbf{H}^2$, conditioned on B not happening (in both of them), the value of $\mathbf{H}^1$ and $\mathbf{H}^2$ is equal. In particular it shows that the probability of of B is the same in both distributions. Therefore the statistical distance between $\mathbf{H}^1$ and $\mathbf{H}^2$ is bounded by the probability of B. In the following, we show that $\Pr_{\mathbf{H}^2}[B] \leq 2\delta$. In the the following all the probabilities will be in the experiment for $\mathbf{H}^2$.

Let ϵ, δ and M be as in Step 2 of Adv: $\epsilon = \frac{\delta}{qN}$, and $M = \frac{q}{\epsilon\delta}$. We start by showing: $\Pr[C] \leq \delta$ where the event C is defined as

$$C: \exists x \notin Q(L_M) \text{ that is obtained in } \mathbf{D}_M \text{ with prob } \geq \epsilon$$

and $\mathbf{D}_i$ is defined, as in Step 2 of Adv to be the distribution of the transcript of the first signature conditioned on the information in L_i.

> PROOF OF $\Pr[C] \leq \delta$. For every possible query x to the random oracle, let q_x denote the probability, taken over both the random oracle and the randomness used by Gen and Adv, that x is queried when generating a key and then signing and verifying α_0. Then $\sum_x q_x \leq q$ (*) since this sum is the expected number of queries in this process. Let p_x denote the probability that x is learned at some iteration of Step 2. Then, $q_x \geq \epsilon p_x$ (**). Indeed, if A_i is the event that x is learned at the ith iteration, then since these events are disjoint $q_x = \Pr[x \text{ is queried}] \geq \sum_{i=1}^{M} \Pr[x \text{ is queried} \mid A_i] \Pr[A_i]$. But by definition of the learning process , $\Pr[x \text{ is queried} \mid A_i] \geq \epsilon$ and hence $q_x \geq \epsilon \sum_{i=1}^{M} \Pr[A_i] = \epsilon p_x$. But the event C only occurs if M distinct queries are learned in Step 2. Hence, if it happens with probability more than δ then the expected number of queries learned, which is $\sum_x p_x$, is larger than δM. Yet combining (*) and (**), we get that $\delta M < \sum_x p_x \leq \sum_x q_x/\epsilon \leq q/\epsilon$, contradicting the fact that $M = q/(\epsilon\delta)$. $\square$

Now we will show that $\Pr[B \mid \neg C] \leq \delta$, and it means that $\Pr[B] \geq \Pr[\neg C]\Pr[B \mid \neg C] \geq (1-\delta)^2 > 1 - 2\delta$. Note that Adv makes all its operations in Step 4 based solely on the information in L_M, and the answers chosen for queries $Q(T_0)\backslash Q(L_M)$ does not affect it (because even if queries in $Q(T_0)\backslash Q(L_M)$ are asked by Adv, they will be answered at random). So, it means that the value of $\mathbf{H}^2$ is independent of T_0, conditioned on knowing L_M. Thus, instead of thinking of T_0 being chosen first, then L_M computed and then all queries of Step 4 being performed, we can think of L_M being chosen first, then Adv runs Step 4 based on L_M to sample $\mathbf{H}^2$, and then T_0 is chosen conditioned on L_M and $\mathbf{H}^2$. But because of the independence of T_0 and $\mathbf{H}^2$ conditioned on L_M, the distribution of T_0 conditioned on L_M and $\mathbf{H}^2$ is that conditioned on only L_M which has the distribution $\mathbf{D}_M$. Now assume that L_M makes the event $\neg C$ happen (note that C is defined by L_M.). Since at most qN queries are made in Step 4, and C has not happened, when T_0 is chosen from $\mathbf{D}_M$, the probability that $Q(T_0) \backslash Q(L_M)$ contains one of these queries is at most $\epsilon qN = \delta$. Therefore we get $\Pr[B \mid \neg C] \leq \delta$, and $\Pr[B] < 2\delta$. $\square$

Claim 4.6 (Restated). $\mathbf{H}^2 \equiv \mathbf{H}^3$. *Thus,* $\Pr_{\mathbf{H}^2}[\cup_{j\geq 1} E_j] = \Pr_{\mathbf{H}^3}[\cup_{j\geq 1} E_j]$.

Proof. In the sampling of $\mathbf{H}^3$ we can think of L_M being chosen first (although not needed), and then T_0 being chosen conditioned on L_M (i.e., from the distribution $\mathbf{D}_M$), and then Step 4 of the experiment is done while any query in $Q(L_M) \cup Q(T_0)$ is answered according to L_M, T_0, and any other query is answered randomly. (That is we sample L_M and T_0 in the reverse order.) The point is that during the sampling process of $\mathbf{H}^2$ we are also doing exactly the same thing. Again, we sample L_M first. Then $\tilde{T}_0$ is chosen from the distribution $\mathbf{D}_M$. Then Step 4 is done while any query in $Q(L_M) \cup Q(\tilde{T}_0)$ is answered according to $L_M, \tilde{T}_0$, and all other queries (even the ones in $Q(T_0)\backslash Q(L_M)$) are answered randomly. Therefore $\mathbf{H}^2$ and $\mathbf{H}^3$ have the same distribution. $\square$

Claim 4.7 (Restated). $\Pr_{\mathbf{H}^3}[\cup_{j\geq 1} E_j] \geq 1 - \lambda$.

Proof. We will prove that this holds for every fixed oracle and randomness of all parties, as long as the permutation $\alpha_0, \ldots, \alpha_{N-1}$ is chosen at random. For every fixing of the oracle and randomness and $j \in [0..N-1]$, let $U_j = G \cup S_{\beta_j}$ denote the set of queries made by either the key generation algorithm or the signing algorithm for message β_j, and let V_j be the set of queries made by the verification algorithm while verifying this signature. The proof will follow from this fact:

COMBINATORIAL LEMMA: If $U_1, \ldots, U_K, V_1, \ldots, V_K$ are subsets of some universe satisfying $|U_i| + |V_i| \leq q$ and $U_i \cap V_j \not\subseteq V_i$ for every $i \neq j$ then $K \leq \binom{q}{q/2}$.

The Combinatorial Lemma immediately implies Claim 4.7. Indeed, for every i, j with $i \neq j$, define the event $E_{i,j}$ to hold if $U_i \cap V_j \subseteq V_i$. Then, there must be at least $N - \binom{q}{q/2} = N(1 - \lambda)$ number of i's (i.e., $1 - \lambda$ fraction of them) such that $E_{i,j}$ holds for some j (otherwise we could remove all such i's and obtain a larger than $\binom{q}{q/2}$-sized family contradicting the combinatorial Lemma). But, if we choose a permutation $\alpha_0, \ldots, \alpha_{N-1}$ such that $\alpha_0 = \beta_i$ for such an i then the event $\cup_j E_j$ holds.

Thus, all that is left is to prove is the combinatorial lemma. It essentially follows from Bollobás's Inequality [Bol65], but we repeat the argument here. Assume for the sake of contradiction that there is a family $U_1, \ldots, U_K, V_1, \ldots, V_K$ satisfying conditions of the lemma with $K > \binom{q}{q/2}$. First, we can remove any elements from U_i that are also in V_i, since it will not hurt any of the conditions. It means that now we have: for every i, j, $U_i \cap V_j = \emptyset$ iff $i = j$. Now, take a random ordering of the universe $W = \bigcup_i (U_i \cup V_i)$, and let A_i be the event that all the members of U_i occur before the members of V_i in this order. The probability of A_i is $\frac{|U_i|!|V_i|!}{(|U_i|+|V_i|)!} = 1/\binom{|U_i|+|V_i|}{|V_i|} \geq 1/\binom{q}{|V_i|} \geq 1/\binom{q}{q/2}$. Hence if $K > \binom{q}{q/2}$, there is a positive probability that both A_i and A_j hold for some $i \neq j$. But it is not hard to see that in that case, either U_i and V_j are disjoint or U_j and V_i are disjoint, contradicting our hypothesis. $\square$

5   A One-Time Signature Scheme

The following Theorem shows that Theorem 1.1 is tight up to a constant factor in the number of queries.

Theorem 5.1. *There is a one-time signature scheme* (Gen, Sign, Ver) *for messages* $\{0,1\}^*$, *using a total of q queries to a random oracle that has security $2^{(0.812-o(1))q}$, where $o(1)$ is a term tending to 0 with q.*

Proof. The scheme is basically Lamport's Scheme [Lam79] with two changes: **(1)** we use a more efficient anti-chain (family of incomparable sets) than Lamport's scheme (a well-known optimization) and **(2)** we use a secret "salt" value for the hash function to prevent a birthday attack.

The Scheme Description. Let $c = (3 - \sqrt{5})/2$ and k be such that $(1 + c)k + 4 = q$.

- Generate the keys by choosing k random strings $x_i \in \{0,1\}^{q+i}$ for $0 \leq i \leq k - 1$, and an additional random string $z \in \{0,1\}^{2q}$. [10] The secret key consists of these values, and the public key is $\mathcal{O}(x_1), \ldots, \mathcal{O}(x_k), \mathcal{O}(z)$.

- Let $h(\alpha)$ be the first $\log \binom{k}{ck}$ bits of $\mathcal{O}(z, \alpha)$, which we identify with a ck-sized subset of $0, \ldots, k - 1$. The signature of α consists of $\{x_i\}_{i \in h(\alpha)}$ and the string z.

---

[10] If we choose all of them from $\{0,1\}^q$ the scheme is still as secure as we claimed, but now the analysis is simpler.

- To verify a signature, we first verify that $O(z)$ is equal to its alleged value, then we ask $\mathcal{O}(z, \alpha)$ to know $h(\alpha)$, and then we ask ck more queries to check that the released strings are indeed preimages of the corresponding entries of the public key indexed by $h(\alpha)$.

The number of queries is $q = (1 + c)k + 4$, while, as we will see, the security is at least $\Omega(\binom{k}{ck}) = 2^{(H(c)-o(1))k} = 2^{\frac{H(c)-o(1)}{1+c}q} > 2^{(0.812-o(1))q}$ where $H(\cdot)$ is the Shannon entropy function.

Let T be the total number of oracle queries asked by the adversary and $\alpha \neq \beta$ be (in order) the message for which she asks a signature and the message for which she tries to forge a signature. We assume without loss of generality that $T < 2^{q-1}$, because $2^{q-1} \gg \binom{k}{ck}$. We divide the winning cases for the adversary into three cases:

1. The adversary chooses some $z' \in \{0,1\}^{2q}, z' \neq z$ such that $\mathcal{O}(z) = \mathcal{O}(z')$, alleged to be the real z in the signature of β.

2. The adversary uses the real z in the signature of β and $h(\alpha) = h(\beta)$.

3. The adversary uses the real z in the signature of β and $h(\alpha) \neq h(\beta)$.

We will show that the probability that the adversary wins conditioned on being in case 3 is at most $O(T/\binom{k}{ck})$, and the probability that either case 1 or case 2 happens at all is also at most $O(T/\binom{k}{ck})$. So, the total probability of winning for the adversary will be at most $O(T/\binom{k}{ck})$ as well.

In case 1, even if we reveal z to the adversary in the first place (x_i's are irrelevant), she has the chance of at most $(1+T)/2^q$ to find some $z' \neq z$ such that $\mathcal{O}(z) = \mathcal{O}(z')$. That is because she gets to know at most T oracle query/answer pairs (other than $\langle z, \mathcal{O}(z)\rangle$), and the probability that she gets $\mathcal{O}(z)$ in one of them is at most $T/2^q$. If she does not see $\mathcal{O}(z)$ as an oracle answer, she needs to guess z' blindly which succeeds with probability at most $1/2^q$.

In the case 2, we reveal all x_i's to the adversary at the beginning, although they are indeed irrelevant to finding a pair $\alpha \neq \beta$ such that $h(\alpha) = h(\beta)$ (because they are of length $< 2q$). Before the adversary gives us α, it asks at most T queries of length $2q$. So, the probability that she gets some $z' \in \{0,1\}^{2q}$ such that $\mathcal{O}(z') = \mathcal{O}(z)$ is at most $T/2^{2q} = o(T/\binom{k}{ck})$. Let assume that this has not happened. So, we can pretend that when we receive α, the value of z is chosen at random different from the members of $\{0,1\}^{2q}$ that are asked from the oracle by Adv. Thus, the probability that any adversary's query so far with length more than $2q$ has the prefix z will be at most $T/(2^q - T) < T/2^{q-1} = O(T/\binom{k}{ck})$. It means that with probability $1 - O(T/\binom{k}{ck})$, so far were no query asked from the oracle which has z as prefix. Assuming this is the case, when we ask the query (z, α) from the oracle, $h(\alpha)$ is chosen uniformly at random from $\{0,1\}^{\log \binom{k}{ck}}$. Hence, if the adversary asks T more oracle queries of the form (z, γ) where $\gamma \neq \alpha$, one of them will give $h(\gamma) = h(\alpha)$ with probability at most $T/\binom{k}{ck}$, and if it does not happen for any of them, a blind guess β by the adversary will give $h(\alpha) = h(\beta)$ with probability $1/\binom{k}{ck}$. So, the probability of getting $\alpha \neq \beta$, $h(\alpha) = h(\beta)$ for the adversary is at most $O(T/\binom{k}{ck})$.

In the case 3, there always is some $i \in h(\beta)\backslash h(\alpha)$. We choose the smallest such i, call it i_0, and change the game slightly by revealing z to Adv from the beginning and revealing all x_j's for $j \neq i_0$ to the her after she gives us β. It only might increases her chance of success (although they are irrelevant because they have different length). For any fixed $i \in 0, \ldots, k-1$, we show that the probability of the adversary to find a preimage for $\mathcal{O}(x_i)$ conditioned on $i = i_0$ is at most $(T+1)/2^{q+i_0} < (T+1)/2^q$ (which is necessary for her to win), and then by the union bound, the

probability of success for the adversary in this case will be at most $k(T+1)/2^q = O(T/\binom{k}{ck})$. The reason is that the adversary can ask at most T oracle queries after we reveal in order to find a preimage for $\mathcal{O}(x_{i_0})$. The probability that for one of the queries x among these T queries she asks we have $\mathcal{O}(x_i) = \mathcal{O}(x)$ is at most $(T)/2^{q+i_0}$, and when it does not happen, the adversary has to guess a preimage for $\mathcal{O}(x_i)$ blindly, which will be correct with probability $1/2^{q+i_0}$.

$\qquad\square$

The constant c in the description of the scheme maximizes $\binom{k}{ck}$, conditioned on $q \approx (1+c)k$. The same ideas show that whenever $n \leq dq$ where $d \approx 0.812$ is obtained as above $(d = H(c)/(1 + c))$, then there is a one-time signature scheme for messages $\{0,1\}^n$ that makes only q queries and achieved security exponential in the length of its queries.

6   Extensions

Now we prove several extensions of Theorem 1.1.

6.1   Other oracles

Using minor changes to the proof of Theorem 4.1 we can get a similar lower bound for signature schemes based on the ideal cipher or a random permutation oracles. This is important as these oracles are also sometimes used to model highly efficient symmetric-crypto primitives, and so it is an interesting question whether such oracles can be used to construct signatures more efficiently.

Theorem 6.1. *Let $\mathcal{O}$ be either the ideal cipher oracle. Then, for every one-time signature scheme for messages $\{0,1\}^n$ using a total of $q \leq n/4$ queries to $\mathcal{O}$ there is an adversary making $2^{(4-o(1))q}$ oracle queries that breaks the scheme with probability $1 - o(1)$, where $o(1)$ denotes a term tending to 0 with q. In case of $\mathcal{O}$ being the random permutation oracle, only $q \leq n/2$ is needed to get and adversary asking $2^{(2-o(1))q}$ queries, breaking the scheme with probability $1 - o(1)$.*

Proof. We explain the proof for the ideal cipher oracle. Extending the proof for the random permutation oracle is straightforward.

We change both the signature scheme and the oracle for the sake of the analysis. We let the new oracle $\mathcal{O}'$ be the same as $\mathcal{O}$ except that $\mathcal{O}'$ does not answer queries of the form (k, x, d) whenever $|x| < 2(q + \log q)$. Instead it answers queries of the type (k, n) where $n < 2(q + \log q)$, to which it returns the long string containing the concatenation of $\mathcal{O}(k, x, \mathsf{F})$ for $x \in \{0,1\}^n$.

We change the signature scheme to get a new scheme $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ as follows: **(1)** use $\mathcal{O}'$ instead of $\mathcal{O}$ and **(2)** whenever an algorithm makes a query (k, x, d) and obtains an answer y, it will also make the "redundant" query $(k, y, \bar{d})$ (where $\bar{d} = \mathsf{B}$ if $d = \mathsf{F}$ and vice versa). Note that the total number of queries of the new scheme is at most $q' = 2q$.

Lemma 6.2. *Given the scheme $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$, there is an adversary Adv making at most $\mathrm{poly}(q')2^{q'}$ queries from $\mathcal{O}'$ that breaks the scheme with probability $1 - o(1)$.*

Lemma 6.2 implies Theorem 6.1 since any such adversary can be implemented using the oracle $\mathcal{O}$ with at most a $q^2 2^{2q}$ factor increase in the number of queries, and the total number of queries will be $\mathrm{poly}(q')2^{q'}q^2 2^{2q} = 2^{(4-o(1))q}$.

16

Proof. The description of the attack remains basically the same as that of Theorem 4.1 set by parameters in Corollary 4.2 (i.e. $N = 2^q, \lambda = \delta = \theta(q^{-1/2})$), and we have the same distributions $\mathbf{H}^0, \mathbf{H}^1, \mathbf{H}^2, \mathbf{H}^3$ as before. However, there are some minor changes as follows:

- During Step 2 of the attack, whenever learn a query, we add both the query and its dual to L_i.

- During Step 4 of the attach we might discover an inconsistency between the guesses we made in the sampled transcript $\tilde{T}_0$ and the answers we receive from the oracle $\mathcal{O}$. That is, we might get the same answer for two different plain texts with the same key. However, as we will see this will only happen with small probability, and we ignore this case safely. .

- The definition of $\mathbf{H}^2$ needs to change a little. Namely, in the experiment for the distribution $\mathbf{H}^2$, during the signing and verification of $\alpha_1, \ldots, \alpha_N$, whenever we make a new non-redundant query (k, x, d), we look at all queries of the form $(k, \cdot, d)$ appearing either in the transcript of the system so far (i.e. $\tilde{T}_0, T_1, \ldots$) or in the learned queries of L_M. Then we choose a random answer y from the set of unused answers and use it as the oracle answer for (k, x, d). The next redundant query $(k, y, \bar{d})$ is simply answered by x.

The differences between the proof in this case and the proof of Theorem 4.1 are the following:

- We need to include the condition in the event E_j that the queries made in the jth signing and verification are consistent with (the key generation part of) the transcript $\tilde{T}_0$ in the sense that they do not specify two queries $(k, x, d), (k, x', d), x \neq x'$ which map to the same answer y. The consistency condition guarantees (by definition) that if E_j occurs, then the verifier will accept the jth signature.

  The combinatorial condition $V_j \cap (\tilde{G} \cup \tilde{S}_0) \subseteq V_0$ still guarantees that the jth verification does not ask any query for which we have guessed the answer.[11]

  We can still prove that $\Pr_{\mathbf{H}^0}[\exists_j E_j] = \Pr_{\mathbf{H}^1}[\exists_j E_j]$ using basically the same proof as in Claim 4.4. We just have to note that as long as E_j happens in both experiments, there is no way to distinguish their j^{th} signing and verification, and the consistency also happens either in both or in none of them. s

- We again show $\mathsf{SD}(\mathbf{H}^1, \mathbf{H}^2) = o(1)$. The reason is that the difference between the distributions $\mathbf{H}^1$ and $\mathbf{H}^2$ is due to some events which happen with probability $o(1)$. That is there are events in the experiments of sampling $\mathbf{H}^1$ and $\mathbf{H}^2$ which happen with probability $o(1)$ and conditioned on they not happening, $\mathbf{H}^1$ and $\mathbf{H}^2$ have the same distribution.

  - Similar to Claim 4.5 one of the differences between the distributions $\mathbf{H}^1, \mathbf{H}^2$ might be because of Adv asking a query in $Q(T_0) \setminus Q(L_M)$. Because of the same analysis given in the proof of Claim 4.5 the probability that we ask any such query (in both experiments) is at most $2\delta = o(1)$. So, in the following we assume that this case does not happen.

---

[11]This also guarantees that there is no inconsistency between the jth verification and the transcript $\tilde{T}_0$, but later we will show that the total consistency happens with good probability

– In experiment of sampling $\mathbf{H}^1$, when a new non-redundant query (k, x, d) is asked in the $1 \leq i^{\text{th}}$ signing or verification, the returned answer y might be equal to a guessed answer for a query (k, x', d) of $\tilde{T}_0$ (we call this event F_1), but it is never equal to the answer of a query $(k, x'', d) \in Q(T_0) \setminus Q(L_M)$. The situation for $\mathbf{H}^2$ is the reverse: On a new non-redundant query (k, x, d) during the $1 \leq i^{\text{th}}$ signing or verification, the answer is never equal to a guessed answer for a query (k, x', d) in $\tilde{T}_0$, but it might be equal to the answer of a query $(k, x'', d) \in Q(T_0) \setminus Q(L_M)$ (we call this event F_2). Note that $(\mathbf{H}^1 \mid \neg F_1) \equiv (\mathbf{H}^2 \mid \neg F_2)$. As we will see, $\Pr[F_i] = o(1)$ for $i = 1, 2$ which shows that $\mathsf{SD}(\mathbf{H}^1, \mathbf{H}^2) = o(1)$.

The reason for $\Pr[F_1] = o(1)$ is that whenever we have a new non-redundant query in the $1 \leq i^{\text{th}}$ signing or verification, its answer is chosen from a set of size at least $q^2 2^{2q} - q' 2^{q'}$ which might hit a guessed answer for a query in $\tilde{T}_0$ with probability at most $q'/(q^2 2^{2q} - q' 2^{q'}) = o(1)$. The same argument holds for $\Pr[F_2] = o(1)$.

- Claim 4.6 still holds with the similar proof because of the way we defined $\mathbf{H}^2$ for the case of ideal cipher.

- Claim 4.7 is still correct with the same proof. Note that all the signing and verifications are consistent.

$\square$

A similar and simpler proof works for the case of a random permutation oracle. In this case, we again change the oracle by merging small queries into a single query with a huge answer, but we don't have the issue of adding "dual" queries, and therefore the condition $q \leq n/2$ (rather than $q \leq n/4$) is enough to get an adversary who breaks the scheme with probability $1 - o(1)$ by asking $2^{(2-o(1))q}$ queries (rather than $2^{(4-o(1))q}$ queries). $\square$

6.2 Implementing Adversary in $\mathbf{BPP}^{\mathbf{NP}}$.

If the signature scheme is efficient, using an $\mathbf{NP}$ oracle, our adversary can run in time $\mathrm{poly}(n, 2^q)$, where n is the length of the messages to be signed.[12] That is, we prove the following lemma:

Lemma 6.3. *If the signature scheme is efficient, the adversary of the proof of Theorem 4.1 can be implemented in $\mathrm{poly}(n, 2^q)$ time using an oracle to an $\mathbf{NP}$-complete problem.*

Lemma 6.3 can be interpreted as saying that a non-black-box proof of security for a signature scheme more efficient than the lower bounds provided by Theorem 4.1 will necessarily imply a proof that $\mathbf{P} \neq \mathbf{NP}$.

The only place in which the adversary uses its unbounded computational power is in Step 2 where it chooses x_i to be the lexicographically first unlearned string in $\{0, 1\}^l$ such that x_i is queried in $\mathbf{D}_i$ with probability at least ϵ, and in Step 3 when it samples a random $\tilde{T}_0$ from $\mathbf{D}_M$.

We show that:

---

[12]In general, the security parameter could be different from the length of the messages n. For example, in Section 5, the security parameter was q (so the security was $2^{\Omega(q)}$), and the running time of the algorithms was $\mathrm{poly}(n, q)$. Here, for simplicity, we assume that $\ell = \mathrm{poly}(n)$, and all the algorithms' queries are of length ℓ.

- Using an **NP** oracle, we can sample from a distribution $\mathbf{D}'_i$ in expected $\mathrm{poly}(n, 2^q)$ time such that $\mathsf{SD}(\mathbf{D}'_i, \mathbf{D}_i) \leq \epsilon$, where ϵ is as defined in Step 2.

- Using the $\mathbf{D}'_i$ sampler, we can implement the adversary in $\mathrm{poly}(n, 2^q)$ time with similar success probability.

We first show how to use a $\mathbf{D}'_i$ sampler to implement the adversary efficiently and then will show how to sample from $\mathbf{D}'_i$ efficiently using an **NP** oracle.

Efficient adversary using a $\mathbf{D}_i$ approximate-sampler. So, here we assume that we can sample efficiently from a distribution $\mathbf{D}'_i$ such that $\mathsf{SD}(\mathbf{D}'_i, \mathbf{D}_i) \leq \epsilon$. In order to choose x_i in the ith step of the learning phase, we do the following. Let $m = (l + \log M - \log \delta)/\epsilon^2$. We sample m times from $\mathbf{D}'_i$ to get $D^1_i, \ldots, D^m_i$. Then we choose x_i to be the lexicographically first unlearned query (i.e. not in L_{i-1}) which appears in at least 2ϵ fraction of $Q(D^j_i)$'s.

Claim 6.4. *With probability at least $1 - \delta$ we get the following: For every $x \in \{0,1\}^l$, and every $1 \leq i \leq M$:*

1. *If $\Pr[x \in Q(\mathbf{D}_i)] \geq 3\epsilon$, then x appears in more than 2ϵ fraction of $Q(D^j_i)$'s.*

2. *If $\Pr[x \in Q(\mathbf{D}_i)] \leq \epsilon$, then x appears in less than 2ϵ fraction of $Q(D^j_i)$'s.*

If the event above happens, it means that the learning algorithm learns all the 3ϵ-heavy queries in its M rounds with probability at least $1-\delta$ (using the same argument as before). Therefore we get a weaker, yet strong enough, version of Claim 4.6 saying that the $\mathsf{SD}(\mathbf{H}^1, \mathbf{H}^2) \leq 3\delta + \delta + \delta = o(\delta)$.

The Claim 6.4 follows from the Chernoff bound. The probability that any specific x violates the claim's condition in any of the rounds is at most $e^{-m\epsilon^2} < 2^{-m\epsilon^2} = 2^{-l - \log M + \log \delta}$. By union bound, the probability that the event is not violated at most $M2^l 2^{-l - \log M + \log \delta} = \delta$.

Sampling $\mathbf{D}'_i$ efficiently using an NP oracle. Note that L_i which captures our knowledge of the system after the ith round of the learning phase can be encoded with $\mathrm{poly}(n, 2^q)$ bits. The number of random bits used by the adversary till the end of the ith round of the learning phase is also $\mathrm{poly}(n, 2^q)$. For some technical reason which will be clear later, we add the randomness used by the adversary to the description of L_i. Similarly, any (possible) transcript D which $\Pr[\mathbf{D}_i = D] > 0$ can be represented with $\mathrm{poly}(n, q) < \mathrm{poly}(n, 2^q)$ bits. In the following we always assume that such encodings are used to represent L_i and D.

In order to sample from a distribution close to $\mathbf{D}_i$ we use the following Lemma:

Lemma 6.5. *There is a function $f : \{0,1\}^* \times \{0,1\}^* \mapsto \mathbb{N}$ which is efficiently computable (i.e. time $\mathrm{poly}(n, 2^q)$), with the following properties:*

1. *$f(L_i, D) = \lfloor c\mathbf{P}[\mathbf{D_i} = D] \rfloor$ for some constant c depending on L_i. So we have $f(L_i, D) = 0$ if $\Pr[\mathbf{D_i} = D] = 0$.*

2. *$f(L_i, D) \geq 10/\epsilon$ whenever $\Pr[\mathbf{D} = D] > 0$ where ϵ is as defined in Step 2.*

Before proving the lemma, we see how it is used.

Corollary 6.6. *We can sample from a distribution $\mathbf{D}'_i$ such that $\mathsf{SD}(\mathbf{D}_i, \mathbf{D}'_i) \le \epsilon$ in time $\mathrm{poly}(n, 2^q)$ (where the time $\mathrm{poly}(n, 2^q)$ is independent of i for $1 \le i \le M$).*

Proof. Let $W_i = \{(D, j) \mid 1 \le j \le f(L_i, D)\}$ be the set of "witnesses" for L_i, where f is the function in Lemma 6.5. Lemma 6.5 shows that the relation $R = \{(L_i, w) \mid w \in W_i\}$ is an **NP** relation. It is known [BGP00] that for any **NP** relation, there is a witness-sampling algorithm that given any x, samples one of the witnesses of x uniformly in expected $\mathrm{poly}(|x|)$ time. Therefore, we sample a random $w = (D, j)$ such that $w \in W_i$ in expected $\mathrm{poly}(n, 2^q)$-time, and output D. It is easy to that the distribution $\mathbf{D}'_i$ of our output has statistical distance at most ϵ from the distribution $\mathbf{D_i}$. $\qquad\square$

Proof. (Lemma 6.5) Recall that $\mathbf{D}_i$ is the distribution of transcripts T_0 conditioned on the information given in L_i. Let the event $E(L_i)$ be the event that during the running of the system (and our attack) adversary's knowledge about the system and its randomness after the ith round of the learning is what L_i denotes. Similarly, let $E(D)$ be the event that $D = T_0$ is the case in our experiment. Thus, for every transcript D, $\Pr[\mathbf{D}_i = D] = \Pr[E(D) \mid E(L_i)]$. If we could compute $\Pr[E(D) \mid E(L_i)]$, we could somehow use it in the Lemma 6.5, but instead of doing that, we will rather compute $\Pr[E(D) \wedge E(L_i)]$ which is proportional to $\Pr[E(D)|E(L_i)]$ up to a constant factor depending on L_i, and will scale it up to some big integer.

Given L_i and D, in order to compute $\Pr[E(D) \wedge E(L_i)]$, we track the whole experiment from the beginning in the following order:

- Key Generation

- Signing α_0

- The attack (which includes the verification of α_0 as its first step) to the end of the ith round of the Learning.

At any moment that some coin tossing is involved (i.e. in the key generation algorithm, in the attack, or fin an oracle answer), the result is determined by the description of L_i and D. Thus, we can calculate the probability that given values of L_i and D will be the ones in the real running of the experiment[13]. More quantitatively, during the simulation of the experiment, we receive any specific oracle answer with probability at least 2^{-l} whenever it is a possible answer and the probability of getting a specific random tape for the key generation and the adversary is at least $2^{-\mathrm{poly}(n, 2^q)}$. Since the total probability of $\Pr[E(D) \wedge E(L_i)]$ is the multiplication of all those probabilities that we get during the simulation of the system, and because the number of oracle queries that we examine is at most $2^{O(q)}$, we get $\Pr[E(D) \wedge E(L_i)] > 2^{-\mathrm{poly}(n, 2^q)}$ whenever $\Pr[E(D) \wedge E(L_i)] \ne 0$. Note that ϵ in the attack is $2^{-O(q)}$. Therefore, for a big enough constant $c = \mathrm{poly}(n, 2^q)$, the function $f(L_i, D) = \lfloor c \Pr[E(D) \wedge E(L_i)] \rfloor$ is computable in time $\mathrm{poly}(n, 2^q)$ and we have $f(L_i, D) > 10/\epsilon$ as well. $\qquad\square$

6.3 Handling imperfect completeness

While the typical definition of a signature scheme stipulates that a valid signature (generated by the signing algorithm with the correct key) should be accepted with probability one, it makes sense

---

[13]For the case of ideal cipher or random permutation oracles, we need to keep track of the oracle answers so far during the simulation of the experiment, in order to know what is that probability of receiving a specific answer from the oracle at any point.

to consider (especially in the context of negative results) also signatures where the verifier may reject such signatures with small probability, say $1/10$. We are able to extend our result to this case as well:

Theorem 6.7. *For every one-time signature scheme for messages $\{0,1\}^n$, accepting correct signatures with probability at least 0.9 (over the randomness of the verifier), and asking a total $q \leq \sqrt{n}/20$ queries to a random oracle, there is* **(1)** *an adversary making $2^{(1+o(1))q}$ queries that breaks the scheme with probability at least 2^{-q} and* **(2)** *an adversary making $2^{O(q^2)}$ oracle queries that breaks the scheme with constant probability.*

The proof of part **(1)** is a straightforward extension of the proof of Theorem 4.1 and so we bring here the proof of part **(2)**:

Lemma 6.8. *For every one-time signature scheme with imperfect completeness (i.e., verifier can reject valid signatures with probability at most $1/10$ over its coins) there is an adversary asking $2^{O(q^2)}$ queries that finds with probability $1 - o(1)$ a message/signature pair which passes the verification with probability at least 0.7.* [14]

Proof. The main difference between the proof of this lemma compared to that of Theorem 4.1 is the way we define the sets V_j's. They are not simply the queries that the verifications ask from the oracle. For sake of analysis, for every j, we define the set V_j to be the set computed by the following process: run the jth verification algorithm on the generated message/signature pair $m =$ times (for m to be defined later), and let V_j be the set of queries that appeared in at least a $1/(20q)$ fraction of these verifications. Hence, we have $|V_j| \leq 20q^2$. Note that the definition of V_j depends on the oracle used to do the verifications. We will treat the sets V_j's in the analysis similar to what we did to them with their previous definition. So, we define the new parameter $r = 20q^2$ to the upper bound on $|G| + |S_j| + |V_j|$, while q is still an upper bound for $|G| + |S_j|$. As we will see, the proof will be similar to that of Theorem 4.1 and the parameters are set similar to those of Corollary 4.2: $N = 2^r, \lambda = \delta = \frac{\binom{r}{r/2}}{2^r} = \theta(r^{-1/2}) = \theta(1/q), m = 20^3 q^4, \epsilon = \frac{\delta}{mqN}, M = \frac{q}{\epsilon\delta}$. Other than the parameters, the differences compared to the previous attack are:

1. When obtaining the signature σ_0 in Step 1, we run the verification algorithm m times and record in L_0 all the resulting query/answer pairs.

2. In Step 4 we test q^3 times each generated message/signature pair and output the first signature that passes the verification at least a 0.75 fraction of these q^3 times.

We also define the set U_j to be the set of queries that the jth verification asks from the oracle with probability at least $1/(10q)$ over its own randomness after we fix the random oracle. Hence we have $|U_j| \leq 10q^2$

We say that E_j holds if (as before) $V_j \subseteq (\tilde{G} \cup \tilde{S}_0) \cap V_0$. We also say that the event E holds if $U_j \subset V_j$ for every j.

Claim 6.9. *If $E_j \wedge E$ holds, then the jth signature will be accepted by the verifier with probability at least $0.9 - 0.1 = 0.8$ over the randomness of the verifier.*

---

[14] The probability 0.7 could be substituted by any constant less than 0.9 with changing the constants in the proof.

Proof. The only way this won't happen is that with probability at least $1/10$, the verifier makes a query in the (at most q-sized) set $\tilde{G} \cup \tilde{S}_0 \setminus V_0$. But if this happens, then there is a query in that set that is queried first with probability at least $1/(10q)$, yet because E holds that means that it will be contained in $U_j \subset V_j$, contradicting E_j. $\qquad\square$

For any specific $1 \leq j < N$, by Chernoff bound, the probability that the fraction of times that we accept the generated signature for α_j is 0.05 far from its real probability of being accepted by the verifier is at most $e^{-0.05^2 q^3}$ and by union bound, the probability that it happens for some j is at most $2^{20q^2} e^{-q^3/400} = o(1)$. Now suppose $E_j \wedge E$ holds for some $j = j_0$. So by Claim 6.3, fistly we will output a pair of message and signature, and secondly this pair is accepted by the verifier with probability at least 0.7.

Claim 6.10. *We have* $\Pr[E] \geq 1 - o(1)$.

Proof. By the Chernoff bound, the probability that a particular member of U_j is not in V_j is at most $e^{-(\frac{1}{20q})^2 m} = e^{-20q^2}$. By union bound over the members of U_j, and j we have $\Pr[\textbf{(2)}\text{ fails for some } j] \leq 10q^2 2^{20q^2} e^{-20q^2} = o(1)$. $\qquad\square$

Now that we know E holds almost always, it only remains to show that with high probability E_j happens for some j. This time we define the four hybrid distributions $\mathbf{H}^0, \mathbf{H}^1, \mathbf{H}^2, \mathbf{H}^3$ a bit different. Instead of putting in $\mathbf{H}^i$ the query/answer pairs that we received during one verification, we put in $\mathbf{H}^i$ all such pairs that we get at some point during the m times that we run the verification.

The proofs of Claims 4.4–4.7 also work basically in the same way as before:

- Claim 4.4 still holds with the same proof.

- Claim 4.5 still holds with the same proof because of the new smaller value of ϵ that we used.

- Claim 4.6 still holds with the same proof.

- Claim 4.7 is still correct with the same proof because the condition $q \leq \sqrt{n}/20$ guarantees that there is enough room to choose $N \leq 2^n$ different messages in the attack.

So, our adversary asks at most $Nmq + M + Nq^3 = \text{poly}(q)2^r = 2^{O(q^2)}$ queries, and with probability $1 - o(1)$ finds a pair of message/signature passing the verification with probability at least 0.7. $\qquad\square$

We note that the combination of all the above extensions holds as well (e.g., we can implement in $\mathbf{BPP^{NP}}$ an adversary that breaks any signature scheme with imperfect completeness that is based on the ideal cipher).

6.4 Efficiency of the verifier

Because the signing and verification algorithms are run more often than the key generation, lower bounds on their own efficiency is still meaningful. In Section 5 we saw that the signing algorithm can be very efficient while the total number of queries was almost optimal. Here we show that if we want to get an efficient verifier and exponential security at the same time, it makes the total number of queries to be inefficient.

Theorem 6.11. *For every one-time signature scheme for messages $\mathcal{M}$ with total q oracle queries where, if the verification asks at most v, $v \leq q/2$ oracle queries and $|\mathcal{M}| \geq \frac{\binom{q}{v}}{\lambda}$ then there is an adversary asking at most $O(\frac{q^2\binom{q}{v}}{\lambda\delta^2})$ queries that breaks the scheme with probability at least $1 - \lambda - \delta$.*

Before going over the proof note that for any $v, k \in \mathbb{N}$, where $3 \leq v \leq \frac{q}{2}$ (i.e. $1 \leq v - 2 \leq k$ where $v + k + 2 = q$) the scheme of Section 5 can be simply changed to get a new scheme in which the verifier asks v queries by revealing $v - 2$ sized subsets of x_i's as the signature rather than ck sized ones. A similar proof to that of Theorem 5.1 shows that this new scheme has security $\Omega(\binom{k}{v-2}) = \Omega(\binom{q-v-2}{v-2})$. So, if $v = dq$ for constant d, the maximum security S one can get by asking at most $v = q/d$ queries in verification and q queries totally is bounded as $H(\frac{1}{d-1})(1 - 1/d) - o(1) \leq \frac{\log S}{q} \leq H(\frac{1}{d}) + o(1)$ where $H(\cdot)$ is the Shannon's entropy function and $o(1)$ goes to zero with q.

Proof. (Theorem 6.11) The proof is almost the same as that of Theorem 4.1. The only difference is in Claim 4.7 in which we have a restriction that $|V_j| \leq v$, and we conclude that $K \leq \binom{q}{v}$. The only difference in the proof of Claim 4.7 is that now the event A_i has probability at least $\frac{|U_i|!|V_i|!}{(|U_i|+|V_i|)!} = 1/\binom{|U_i|+|V_i|}{|V_i|} \geq 1/\binom{q}{|V_i|} \geq 1/\binom{q}{v}$ because $v \leq q/2$. $\qquad\square$

7 Lower bounds on black-box constructions

In a construction for signature schemes, one might use a *standard primitive* (e.g., one way function) rather than one with ideal security (e.g., random function). These constructions could have different levels of "black-boxness" discussed thoroughly in [RTV04]. What we will call black-box, is called fully black-box in [RTV04]. Here we give a more quantitative definition of such constructions. For simplicity we only define the black-box constructions of signature schemes from hard one-way functions, and the others are similar. After giving the formal definitions we will prove strong lower bounds on the efficiency of signature schemes from symmetric primitives when the construction is black-box.

Definition 7.1. Let F_ℓ denote the set of all functions $f\colon \{0,1\}^\ell \to \{0,1\}^\ell$ over ℓ bits. We call a family of functions $\{f_\ell \mid \ell \in \mathbb{N}, f_\ell \in F_\ell\}$, *s-hard* (to invert), if for any probabilistic algorithm A running in time at most $s(\ell)$, we have $\Pr_{x \leftarrow_R \{0,1\}^\ell}[A(f(x)) \in f^{-1}(f(x))] \leq \frac{1}{s(\ell)}$ where the probability is over the choice of x and the coin tosses of A.

By S-hard functions, for a *set* of functions S, we mean all those which are s-hard for some $s \in S$. (Think of S as the set of all the functions which are super-polynomial, quasi-polynomial, or exponential etc...) So, we will keep the notation that the capital S denotes a set of functions.

For simplicity we use n, the length of the messages to be signed, as the security parameter of the signature scheme (i.e, the efficient schemes will run in time $\text{poly}(n)$ and for larger values of n the scheme becomes more secure).

Definition 7.2. A *black-box* construction of one-time signature schemes for n-bit messages from S-hard one-way functions, with security parameter contraction $\ell(n)$ is made of the following two families of reductions for all $n \in \mathbb{N}$:

- The implementation reduction $I = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ has three components which are algorithms running in time $\text{poly}(n)$ (Gen is probabilistic) and $I^f = (\mathsf{Gen}^f, \mathsf{Sign}^f, \mathsf{Ver}^f)$ satisfies in Definition 3.5 by setting $\mathcal{O} = f$ for any $f \in F_{\ell(n)}$.

- We call A a I^f-*breaker* if A is a (not necessarily efficient) adversary who breaks the security of I^f with non-negligible probability over its own randomness by playing in the game defined in Definition 3.6. The security reduction R is an algorithm running in time $t(n)$ where: **(1):** For any $f \in F_{\ell(n)}$ and any I^f-breaker A, $\Pr_{x \leftarrow_R \{0,1\}^{\ell(n)}}[R^{A,f}(f(x)) \in f^{-1}(f(x))] \geq \frac{1}{w(n)}$ where the probability is over the choice of x and the coin tosses of R and A, **(2):** $t(n)p(n) < s(\ell(n))$ for any $p(n) = \mathrm{poly}(n)$, any $s(\cdot) \in S$ and and large enough n, and **(3):** $w(n) < s(\ell(n))$ for any $s \in S$ and large enough n.

The security parameter contraction factor $\ell(n)$ in Definition 7.2 measures how small the length of the function used in the reduction is (i.e., the security parameter of the primitive used) compared to n (i.e., the security parameter of the signature scheme). The term "security parameter expansion" is used in [HHRS07] for the inverse of the contraction parameter.

Note that having such a reduction, the existence of any efficiently computable family of functions $f \colon \{0,1\}^\ell \to \{0,1\}^\ell$ which is s-hard to invert for some $s \in S$ implies the existence of (efficient) one-time signature schemes which are secure against polynomial-time adversaries. That is because **(1):** We get an efficient implementation of the scheme by efficiently implementing f for I^f, and **(2):** If A is a I^f-breaker running in time $\mathrm{poly}(n)$, the reduction R combined with its subroutine A breaks the s-hardness of f which is not possible.

Now we prove a strong lower bound on the efficiency of signature schemes relying on the efficiency of strong one-way functions. Then we will show how it generalizes to any symmetric primitive and also functions with many hard-core bits.

Theorem 7.3. *Let E denote the set of functions $E = \{f(\ell) \mid f = 2^{\Omega(\ell)}\}$. Any black-box construction of one-time signature schemes for n-big messages from E-hard one-way functions with security parameter contraction $\ell(n)$ needs to ask $\min(\Omega(\ell(n)), n)$ queries from the one-way function.*

Before going over the proof we make two observations. First, if construction uses E-hard functions, it means that we should have $t(n) = 2^{o(\ell(n))}$ and $w(n) = 2^{-o(\ell(n))}$ in the security reduction. Another point is that the existence of such a reduction regardless of how many queries it asks, makes $\ell(n)$ to be $\omega(\log n)$ for otherwise the condition $t(n)\mathrm{poly}(n) < s(\ell(n))$ in Definition 7.2 will be violated. Therefore without loss of generality, we assume that $q \geq \log n$, because otherwise we can ask $\log n$ redundant queries in the key generation algorithm without changing the condition $q \leq \min(\Omega(\ell(n)), n)$.

Proof. For sake of contradiction suppose that there is a black-box construction of signature schemes (I, R) where I asks $q \leq n$ queries from the one-way function and $\log n \leq q = o(\ell(n))$.

The proof will go in two steps. We will first show that any such construction results in a (computationally unbounded) adversary asking $2^{o(\ell)}$ queries from a a random function $f \leftarrow_R F_\ell$ and inverting it on a random point with probability at least $2^{-o(\ell)}$ (where this probability is also over the choice of f). Then we will show that it is not possible to have such an adversary, namely any adversary asking $2^{\ell/3}$ queries has chance of at most $2^{-\ell/3}$ for doing so.

STEP 1. Let A be the adversary of Corollary 4.2 for the implementation of the signature scheme I (note $q \leq n$) asking at most $2^{(1+o(1))o(\ell(n))}$ queries from the function f (note $q \leq o(\ell(n))$) and breaking I^f with probability at least $1 - o(1)$ when f is chosen at random $f \leftarrow_R F_{\ell(n)}$ where $o(1)$ goes to zero with q. For large enough $\ell(n)$, n becomes large enough too, and so does q (because $q \geq \log n$). Therefore A asks at most $2^{o(\ell(n))}$ queries from f and breaks the scheme with probability

24

at least $3/4$ when $f \leftarrow_R F_{\ell(n)}$ for large enough $\ell(n)$. By an average argument, with probability at least $1/2$ over the choice of f, A breaks I^f with probability at least $1/2$ over its own randomness. We call such f's the good ones. Whenever f is good, $R^{A^f,f}$ inverts f on a random point with probability at least $2^{-o(\ell(n))}$, and because f is good with probability at least $1/2$, $R^{A,f}$ inverts f on a random point with probability at least $2^{-o(\ell(n))}$ for a randomly chosen $f \leftarrow_R F_{\ell(n)}$ where the probability is over the choice of f, the choice of the image to be inverted, and the randomness of A. By merging the code of R with A, we get an adversary $B = R^A$ who asks at most $2^{o(\ell(n))}2^{o(\ell(n))} = 2^{o(\ell(n))}$ queries from $f \leftarrow_R F_{\ell(n)}$ and inverts it on a random point (i.e., $y = f(x)$ for $x \leftarrow_R \{0,1\}^{\ell(n)}$) with probability at least $2^{-o(\ell(n))}$.

STEP 2. Suppose B is an adversary asking $2^{\ell/3}$ queries from a random function $f \leftarrow_R F_\ell$ trying to find a preimage for $f(x)$ where $x \leftarrow_R \{0,1\}^\ell$. We can pretend that the value of f at each point is determined at random whenever it is asked for the first time. So, at first x is chosen, $f(x)$ is chosen, and it is given to B. At first B does not have any information about x, so the probability that B asks x in any of its $2^{\ell/3}$ queries is at most $2^{-2\ell/3}$. Assuming it does not ask x, the probability that B receives the answer $f(y) = f(x)$ by asking any $y \neq x$ is at most $2^{-2\ell/3}$. Assuming that none of the mentioned events happens, if it outputs y differen from all queries it has asked from f, $f(y) = f(x)$ happens with probability $2^{-\ell}$. So its chance of winning is at most $2^{-2\ell/3} + 2^{-2\ell/3} + 2^{-\ell} < 2^{-\ell/3}$ (for $\ell \geq 4$).

$\square$

As it is clear from the theorem, our lower bound becomes stronger for larger values of $\ell(n)$ which is also the case in the similar (unconditional) lower bound results [HHRS07, Wee07].

In order to extend the lower bound to other symmetric primitives (and functions with many hard-core bits) we can follow the same steps of the proof of Theorem 7.3 using the following lemma.

Lemma 7.4. *Let P be a symmetric primitive (i.e, one-way function, one-way permutation, collision resistent hash function, pesudorandom generator, pseodorandom function, message authentication code, or block cipher) , or the primitive of functions $f \colon \{0,1\}^\ell \to \{0,1\}^\ell$ with $\ell/2$ hard-core bits. Then, there is an implementation for P for security parameter ℓ with access to either, random oracle, random permutation oracle, or ideal cipher oracle which asks only a constant number of queries of length $\theta(\ell)$ from the oracle, and any (computationally unbounded) adversary Adv who asks at most $2^{o(\ell)}$ queries from the oracle has chance of at most $2^{-\Omega(\ell)}$ of breaking it (over the randomness of Adv and the oracle used).*

Proof. We will describe the natural implementations and will show the proof of security only for the case that P is the primitive of functions with $\ell/2$ hard-core bits. The security proofs for other implementations are also easy to get (in fact, we already gave the proof for the case of one-way function in the proof of Theorem 7.3).

- **One-way function** using random oracle: To define the value of the function f on input $x \in \{0,1\}^\ell$, we simply use the oracle's answer: $f(x) = \mathcal{O}(x)$.

- **One-way permutation** using random permutation oracle: To define the value of the permutation p on input $x \in \{0,1\}^\ell$, we simply use the oracle's answer: $p(x) = \mathcal{O}(x)$.

- **Collision resistent hash function** using random oracle: The value of the hash function h on input $x \in \{0,1\}^\ell$ is made by using the first $\ell/2$ bits of the oracle's answer: $h(x) = b_1 \ldots b_{\ell/2}$ where $\mathcal{O}(x) = b_1 \ldots b_\ell$.

- **Pseudorandom generator** using random oracle: The stretched output of the generator g on input $x \in \{0,1\}^\ell$ is the output of the oracle on the padded query: $g(x) = \mathcal{O}(x|0^\ell)$.

- **Pseudorandom function** using random oracle: Using the key $k \in \{0,1\}^\ell$ on input $x \in \{0,1\}^\ell$, the output of the function will be the first ℓ bits of the oracle's answer on the query made by attaching k and x: $f_k(x) = b_1 \ldots b_\ell$ where $\mathcal{O}(k|x) = b_1 \ldots b_{2\ell}$.

- **Message Authentication Code** using random oracle: Using the key $k \in \{0,1\}^\ell$, the authentication code of the message $x \in \{0,1\}^\ell$ is defined similar to that of pseudorandom function. The verification is clear.

- **Block cipher** using ideal cipher oracle: Using the key $k \in \{0,1\}^\ell$ and the input $x \in \{0,1\}^\ell$, and the direction d we simply use the oracle's answer $\mathcal{O}(k, x, d)$ as our cipher.

- **Function with $\ell/2$ hard-core bits** using random oracle: The value of the function f on input $x = x_1 \ldots x_\ell$ uses the oracle's answer: $f(x) = \mathcal{O}(x)$ and the hard-core bits for x will be the first $\ell/2$ bits of it: $HC(x) = x_1 \ldots x_{\ell/2}$.

Now we prove the claim for the last primitive (i.e., functions with $\ell/2$ hard-core bits). Suppose the adversary A asks at most $2^{\ell/4}$ queries from the function f. Again, we assume that f chooses its answers randomly whenever asked for the first time. In order to break the hard-core property of the function f, the adversary A needs to distinguish between two experiments. In the first one she is given $(f(x), U_{\ell/2})$ as in put, and in the second one she is given $(f(x), HC(x))$, and in both of the experiments $f \leftarrow_R F_\ell$ and $x \leftarrow_R \{0,1\}^\ell$ are chosen at random. Note that as long as A does not ask x from the oracle, the two experiments are the same. At the beginning A does not knows the second half of the bits of x. So the probability that she asks x from the oracle in one of her $2^{\ell/4}$ queries is at most $2^{\ell/4}2^{-\ell/2} = 2^{-\ell/4}$. Hence, if the probability that she outputs 1 in the experiment i is p_i (for $1 \le i \le 2$), we have $|p_1 - p_2| \le 2^{-\ell/4}$.

$\square$

So by using Lemma 7.4 and following the steps of the proof of Theorem 7.3 we get the following theorem:

Theorem 7.5. *Let E denote the set of functions $E = \{f(\ell) \mid f = 2^{\Omega(\ell)}\}$, and P be either a symmetric primitive or the primitive of functions with $\ell/2$ hard-core bits. Any black-box construction of one-time signature schemes for n-bit messages from an E-hard primitive P with security parameter contraction $\ell(n)$ needs to ask $\min(\Omega(\ell(n)), n/4)$ queries from the primitive P.*

8 Conclusions and open questions

We believe that lower bounds of this form— the efficiency of constructing various schemes using black box idealized primitives— can give us important information on the efficiency and optimality of various constructions. In particular, three natural questions related to this work are:

- Can one pinpoint more precisely the optimal number of queries in the construction of one-time signature schemes based on random oracles? In particular, perhaps our lower bound can be improved to show that the variant of Lamport's scheme given in Section 5 is optimal up to lower order terms.

- What is the threshold d that whenever $n \leq dq$, we can get signature schemes for messages $\{0,1\}^n$ using q oracle queries, and arbitrary large security? Again, it seems that the variant of Lamport's scheme given in Section 5 (working for $\log \binom{k}{ck}$ bit messages without hashing) gives this threshold (i.e., $d \approx 0.812$).

- Can we obtain a $2^{O(q)}$-query attack succeeding with high probability against signature schemes with imperfect completeness?

- Are there stronger bounds for *general* (not one-time) signatures? A plausible conjecture is that obtaining a T-time signature with black-box security S requires $\Omega(\log T \log S)$ queries.

References

[Bar01]   B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115. IEEE, 2001.

[BGP00]   M. Bellare, O. Goldreich, and E. Petrank. Uniform Generation of NP-Witnesses Using an NP-Oracle. *Inf. Comput*, 163(2):510–526, 2000.

[BR93]   M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security*, pages 62–73. ACM, November 1993.

[BM94]   D. Bleichenbacher and U. M. Maurer. Directed Acyclic Graphs, One-way Functions and Digital Signatures (Extended Abstract). In Y. G. Desmedt, editor, *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 75–82. Springer-Verlag, 21–25 Aug. 1994.

[BM96]   D. Bleichenbacher and U. M. Maurer. On the Efficiency of One-Time Digital Signatures. In K. Kim and T. Matsumoto, editors, *ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 145–158. Springer, 1996.

[Bol65]   B. Bollobás. On generalized graphs. *Acta Math. Acad. Sci. Hungar*, 16:447–452, 1965.

[CGH98]   R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *Proc. 30th STOC*, pages 209–218. ACM, 1998.

[DR02]   J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[EGM89]   S. Even, O. Goldreich, and S. Micali. On-Line/Off-Line Digital Signatures. *J. Cryptology*, 9(1):35–67, 1996. Preliminary version in CRYPTO '89.

[GGK03]   R. Gennaro, Y. Gertner, and J. Katz. Lower Bounds on the Efficiency of Encryption and Digital Signature Schemes. In *Proc. 35th STOC*. ACM, 2003.

[GGKT05] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the Efficiency of Generic Cryptographic Constructions. *SICOMP: SIAM Journal on Computing*, 35, 2005. Preliminary versions in FOCS' 00 [GT00] and STOC' 03 [GGK03].

[GT00] R. Gennaro and L. Trevisan. Lower Bounds on the Efficiency of Generic Cryptographic Constructions. In *Proc. 41st FOCS*, pages 305–313. IEEE, 2000.

[Gol04] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229. ACM, 1987. See [Gol04, Chap. 7] for more details.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *J. ACM*, 38(3):691–729, July 1991. Preliminary version in FOCS' 86.

[HHRS07] Haitner, Hoch, Reingold, and Segev. Finding Collisions in Interactive Protocols – A Tight Lower Bound on the Round Complexity of Statistically-Hiding Commitments. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2007.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proc. 21st STOC*, pages 44–61. ACM, 1989.

[KST99] J. H. Kim, D. R. Simon, and P. Tetali. Limits on the Efficiency of One-Way Permutation-Based Hash Functions. In *FOCS*, pages 535–542, 1999.

[Lam79] L. Lamport. Constructing Digital Signatures from a One-Way Function. Technical Report CSL-98, SRI International, Oct. 1979.

[Mer87] R. C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In C. Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer-Verlag, 1988, 16–20 Aug. 1987.

[Nat95] National Institute of Standards and Technology. *FIPS PUB 180-1: Secure Hash Standard*. National Institute for Standards and Technology, Apr. 1995.

[PCST00] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *IEEE Security and Privacy Symposium*, 2000.

[Rab78] M. O. Rabin. Digitalized Signatures. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[RTV04] O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of Reducibility between Cryptographic Primitives. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[Vau92]   S. Vaudenay. One-time identification with low memory. In *Eurocode 92*, number 339 in CISM Courses and Lectures, pages 217–228, Wien, 1992. Springer-Verlag, Berlin Germany.

[Wee07]   H. Wee. One-Way Permutations, Interactive Hashing and Statistically Hiding Commitments. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 419–433. Springer, 2007.