An Efficient Pseudo-Random Generator with Applications to Public-Key Encryption and Constant-Round Multiparty Computation

Ivan Damgård* and Jesper Buus Nielsen**

BRICS* Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Arhus C, Denmark

Abstract. We present a pseudo-random bit generator expanding a uniformly random bit-string r of length $k/2$, where k is the security parameter, into a pseudo-random bit-string of length $2k - \log^2(k)$ using one modular exponentiation. In contrast to all previous high expansion-rate pseudo-random bit generators, no hashing is necessary. The security of the generator is proved relative to Paillier's composite degree residuosity assumption.
As a first application of our pseudo-random bit generator we exploit its efficiency to optimise Paillier's crypto-system by a factor of (at least) 2 in both running time and usage of random bits.
We then exploit the algebraic properties of the generator to construct an efficient protocol for secure constant-round multiparty function evaluation in the cryptographic setting. This construction gives an improvement in communication complexity over previous protocols in the order of nk^2, where n is the number of participants and k is the security parameter, resulting in a communication complexity of $O(nk^2|C|)$ bits, where C is a Boolean circuit computing the function in question.

keywords: constant-round cryptographic protocols, multi-party computation, pseudo-random generator, public-key encryption.

1 Introduction

A pseudo-random bit generator is a deterministic polynomial time algorithm that takes a random k-bit string (a *seed*) as input and produces a output string of length $l > k$. Although the output is of course not a uniformly chosen string of length l, it *looks random*, i.e. it is polynomial time indistinguishable from a uniformly chosen string.

Pseudo-random generators play an important role in almost every area in cryptography, as well as in the analysis of probabilistic algorithms in general. They can be used directly for symmetric encryption, but are also useful in public-key cryptography[BG85].

The first generator known to be good relative to a complexity assumption is due to Blum and Micali [BM84] and was based on the discrete logarithm problem. Later Luby et al. [ILL89] used the Goldreich-Levin hard-core bit theorem[GL89] to show that existence of pseudo-random generators follow from existence of any one-way functions. Also, more efficient generators have been proposed, based on specific assumptions, see for instance [GR00,HSS93,Gen00].

In this paper, we propose a new generator based on Paillier's composite degree residuosity assumption (DCRA). This generator expands a uniformly chosen bit-string r of

* ivan@brics.dk.
** buus@brics.dk.
*** Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

length $k/2$ bits, where k is the security parameter, into a pseudo-random bit-string of length $2k - \log^2(k)$ using one modular exponentiation. Compared to earlier high expansion rate generators based on assumptions related to factoring [GR00,HSS93], we note the following differences: our generator is based on a stronger assumption (DCRA implies that factoring is hard). In return, we get a simpler generator where no hashing is necessary to extract the output (in contrast to [GR00,HSS93]), where the computing time spent per output bit is the same or less than with earlier generators, and where any desired expansion rate can be realized without having to iterate the underlying function. Finally (and most importantly) our generator has some convenient algebraic properties making it extremely useful in connection with public-key encryption and multiparty computation, as detailed below.

As a first application of the generator, we observe that it can be used to implement the encryption function in Paillier's cryptosystem in half the time that was needed by Paillier's original suggestion. In [DJ], Damgård and Jurik introduce a generalisation of Paillier's cryptosystem using arithmetic modulo N^{s+1}, where N is an RSA-modulus and s is an integer ≥ 1. Here, $s = 1$ corresponds to Paillier's original suggestion. Our optimised encryption procedure applies to the generalisation as well, and improves the usage of random bits and number of multiplications by a factor of about $2s$.

The second, and probably most important application is to secure multiparty computation in a constant number of rounds. Here, we are concerned with the classical secure function evaluation problem: n players want to compute a function value $f(x_1, ..., x_n)$ for a given function f, where player i holds the input value x_i. It must be ensured that the result computed is correct, and that the inputs are kept as private as possible, i.e., the result is the *only* new information that is leaked about the input values. This must hold even in presence of adversarial behaviour by some of the players. This is usually modelled by assuming that some subset of them have been *corrupted* by an *adversary*, who learns the inputs and all message exchanged by corrupted players. Different types of adversaries may be considered, in this paper we will only be concerned with *active* and *static* adversaries, i.e. the adversary can control completely the behaviour of corrupted players, and the set of corrupted players is fixed from the beginning. Also, we will be in the *cryptographic* setting, where the adversary is a polynomially bounded machine, who gets access to all messages sent (also between uncorrupted players). We will assume a threshold type of adversary, i.e., any subset of size less than $n/2$ may be the corrupted set.

In this model, it is known that any function that can be efficiently computed, can also be computed securely, if public-key encryption schemes exist [GMW87]. Thus, the important question becomes: how efficiently can it be done? Two parameters are important in practice here: the number of bits we need to send, and the number of *rounds* of communication we need. Based on specific assumptions, the most communication efficient protocol known is from [CDN00] (assuming a given set-up of keys) where one needs to broadcast $O(nk|C|)$ bits, where k is a security parameter and $|C|$ is the size of a Boolean circuit C computing the function in question. The round complexity of that protocol is $O(d)$, where d is the depth of C (this also holds for [GMW87]). The only previously known protocol that works for any function and has a *constant* number of rounds is by Beaver, Micali and Rogaway [BMR90]. Building on Yao's technique for "circuit encryption" [Yao86] they devised a protocol that can be based on any procedure for verifiable secret sharing and an arbitrary pseudo-random bit generator. The idea is to start from a Boolean circuit computing the desired function and compute in constant round an encrypted version of the circuit, plus encrypted representations of the inputs. From this, all players can compute the result on their own without further interaction.

This protocol can work in our communication and adversary model under the same assumption as used by [GMW87]. Its communication complexity depends on the precise generator and VSS used, but is generally speaking very high if we use generic solutions for the generator and VSS.

A first naive idea to make a more efficient solution would be to plug in the the efficient generator we present here and in addition use the idea from [CDN00] where one replaces each VSS by a single ciphertext in a threshold crypto-system. This results in a protocol that still has rather large complexity: one will need to broadcast $\Omega(n^2k^4|C|)$ bits.

In this paper, we describe how the techniques from [CDN00] and our generator can be combined in a different and much more efficient way, namely we show a new way to generalise Yao's circuit encryption to the multiparty case, which in turn allows us to exploit the algebraic properties of our generator. We obtain a constant-round protocol where one needs to broadcast only $O(nk^2|C|)$ bits — only a factor k from the best known non-constant-round protocol. If one is willing to accept a sub-exponential security bound, the communication complexity can become as low as $O(\log^2(k)\max(n,k)k)$. Our protocol assumes for simplicity that keys for a threshold version of Paillier's cryptosystem [DJ] are given, but in fact these keys can be established from scratch (still in constant-round) using techniques from [DK00] and can be reused in many function evaluations.

2 Preliminaries

A distribution ensemble is an indexed family $X = \{X(k,a)\}_{k\in\boldsymbol{N},a\in D}$, where k is the security parameter, D is some arbitrary domain, typically $\{0,1\}^*$, and $X(k,a)$ is a random variable.

Definition 1 (computationally indistinguishable ensembles [GM84,Yao82]). *Let D be any TM which is PPT in its first input, let $k \in \boldsymbol{N}$, $a \in D$, and let $w \in \{0,1\}^*$ be some arbitrary auxiliary input. By the advantage of D on these inputs we mean* $\mathrm{adv}_D(k,a,w) = |\Pr[D(1^k,a,w,X(k,a)) = 1] - \Pr[D(1^k,a,w,Y(k,a)) = 1]|$ *where the probabilities are taken over the random variables $X(k,a)$ and $Y(k,a)$ and the random choices of D.*

We say that two distribution ensembles X and Y indexed by D are computationally indistinguishable *if there exists a negligible function $\delta : \boldsymbol{N} \to [0,1]$ such that for every adversary D, there exists k_D such that for every $k > k_D$, all $a \in D$, $w \in \{0,1\}^*$ we have that* $\mathrm{adv}_D(k,a,w) < \delta(k)$

Definition 2 (pseudo-random function family). *A* pseudo-random function family *is a tuple $(\mathcal{K},F)$, where $F = \{f_K : A_K \to B_K | K \in \mathcal{K}\}$ is a set of* functions *indexed by the* keys *$\mathcal{K}$. Let $\mathcal{K}_k$ the set of keys for security level k. We require that the following can be done in PPT in k: sample a uniformly random key from $\mathcal{K}_k$; for all $K \in \mathcal{K}_k$ sample a uniformly random element from A_K; and for all $a \in A_K$ compute the value $f_K(a)$.*

Further more we require that the two random variables $[K \leftarrow_R \mathcal{K}_k;\ a \leftarrow_R A_K;\ b \leftarrow f_K(a) : (K,b)]$ and $[K \leftarrow_R \mathcal{K}_k;\ b \leftarrow_R B_K : (K,b)]$ are computationally indistinguishable.

3 An Efficient Pseudo-Random Bit Generator

In this section we will define and prove cryptographically secure a pseudo-random function and a corresponding pseudo-random bit generator.

Definition 3. *For each $s \geq 0$ we define a function family* $\mathrm{prf}^{s+1} = (\mathcal{K},F)$. *For $k \in \boldsymbol{N}$ let $\mathcal{K}_k$ consist of the elements (N,g), where $N = pq$ is a k-bit RSA-modulus such that $p = 3 \bmod 4$, $q = 3 \bmod 4$, $\gcd(p-1,q-1) = 2$, and $g \in \boldsymbol{Z}_N^*$ is an element such*

that $\boldsymbol{Z}_N^* = \langle -1, g \rangle$. *For* $(N, g) \in \mathcal{K}_k$ *let* $\mathrm{prf}_{N,g}^{s+1} : \{0,1\}^{\lfloor k/2 - \log(k) \rfloor} \to \boldsymbol{Z}_{N^{s+1}}$ *be given by* $(b, r) \mapsto (-1)^b (g^{N^s})^r \bmod N^{s+1}$, *where* $\|b\| = 1$ *and* $\|r\| = \lfloor k/2 \rfloor - \log(k) - 1$. [1]

The length $\lfloor k/2 - \log(k) \rfloor$ of the seed can be $\lfloor k/2 - O(\log(k)) \rfloor$, but for notational reasons we will use the length $\lfloor k/2 - \log(k) \rfloor$ throughout the proofs.

Let $H_N^{s+1} = \{x^{N^s} \bmod N^{s+1} | x \in \boldsymbol{Z}_N^*\}$ and let $\psi^{s+1} : \boldsymbol{Z}_{N^s} \times \boldsymbol{Z}_N^* \mapsto \boldsymbol{Z}_{N^{s+1}}^*$ be the map given by $f(x, y) = (N+1)^x y^{N^s} \bmod N^{s+1}$. Then the following lemma follows using basic number theory. See [DJ,Pai99] for details.

Lemma 1. *For* $N = pq$ *an RSA-modulus and for* $0 \leq s < p, q$ *we have that the map* ψ^{s+1} *is an isomorphism.*

The following conjecture provides the computational assumption underlying our pseudo-random function. It asserts that a random element from H_N^2 is computationally indistinguishable from a random element chosen from all of $Z_{N^2}^*$.

Conjecture 1 ([Pai99]). The random variables $[(N, g) \leftarrow_R \mathcal{K}_k;\ x \leftarrow_R H_N^2 : (N, x)]$ and $[(N, g) \leftarrow_R \mathcal{K}_k;\ x \leftarrow_R \boldsymbol{Z}_{N^2}^* : (N, x)]$ are computationally indistinguishable.

If Conjecture 1 holds, then the following conjecture claiming that factoring is hard also holds. Little is known about the other direction. For a discussion of Conjecture 1 see [Pai99].

Conjecture 2. For any PPT algorithm the following probability $\Pr[(N, g) \leftarrow_R \mathcal{K}_k : A(N) = p]$ is negligible.

Theorem 1. *For* $s \geq 0$ *the function family* prf^{s+1} *from Definition 3 is a pseudo-random function family relative to Conjecture 1.*

Proof: For prf^{s+1} we can trivially do the following in PPT in k: for $K \in \mathcal{K}_k$ sample $A_K = \{0,1\}^{\lfloor k/2 - \log(k) \rfloor}$ and for $(b, r) \in \{0,1\}^{\lfloor k/2 - \log(k) \rfloor}$ compute $(-1)^b (g^{N^s})^r \bmod N^{s+1}$. Now let

$$
\begin{aligned}
Y_k^{s+1} &= [(N, g) \leftarrow_R \mathcal{K}_k;\ y \leftarrow_R \boldsymbol{Z}_{N^{s+1}}^* : (N, g, y)] \\
X_k^{s+1} &= [(N, g) \leftarrow_R \mathcal{K}_k;\ x \leftarrow_R H_N^{s+1} : (N, g, x)] \\
W_k^{s+1} &= [(N, g) \leftarrow_R \mathcal{K}_k;\ b \leftarrow_R \{0,1\};\ r \leftarrow_R \{0,1\}^{k+\delta_1(k)};\ w \leftarrow \mathrm{prf}^{s+1}(b, r) : (N, g, w)] \\
V_k^{s+1} &= [(N, g) \leftarrow_R \mathcal{K}_k;\ b \leftarrow \{0,1\};\ r \leftarrow_R \{0,1\}^{\lfloor k/2 - \log(k) \rfloor};\ v \leftarrow \mathrm{prf}^{s+1}(b, r) : (N, g, v)]\ .
\end{aligned}
$$

Since random elements from $\boldsymbol{Z}_{N^{s+1}}^*$ and $\boldsymbol{Z}_{N^{s+1}}$ are statistically $2^{-k/2+2}$-close, to prove pseudo-randomness it is enough to prove that the random variables Y_k^{s+1} and V_k^{s+1} are computationally indistinguishable. This follows from Lemmas 2 (proving Y_k^{s+1} and X_k^{s+1} computationally indistinguishable), 3 (proving X_k^{s+1} and W_k^{s+1} statistically indistinguishable for $d_2(k) = k/2$), and 4 (proving W_k^{s+1} and V_k^{s+1} computationally indistinguishable for $\delta_1(k) = k/2$). What remains is to prove that we can sample $\mathcal{K}_k$ in PPT. This easily follows from elementary number theory. Details are omitted here for lack of space. □

Lemma 2. *If Conjecture 1 holds, then for* $s \geq 0$ *the random variables* X_k^{s+1} *and* Y_k^{s+1} *are computationally indistinguishable.*

Proof: Assume for the sake of contradiction, that there exists a PPT distinguisher D and a $c \in \boldsymbol{N}$ such that for infinitely many k it is the case that $|x_{\mathcal{K}_k} - y_{\mathcal{K}_k}| \geq \frac{1}{k^c}$, where $x_{\mathcal{K}_k} = \Pr[D(X_k^{s+1}) = 1]$ and $y_{\mathcal{K}_k} = \Pr[D(Y_k^{s+1}) = 1]$.

[1] Of course the value g^{N^s} can be preprocessed to reduce the computation time.

By [DJ] the random variables $\tilde{X}_k^{s+1} = [(N,g,x) \leftarrow_R X_k^{s+1} : (N,x)]$ and $\tilde{Y}_k^{s+1} = [(N,g,x) \leftarrow_R Y_k^{s+1} : (N,x)]$ are computationally indistinguishable relative to Conjecture 1. Now assume that that k is a value for which $|x_{\mathcal{K}_k} - y_{\mathcal{K}_k}| \geq \frac{1}{k^c}$ and consider the adversary in Fig. 1.

Since for any N and a random $g \leftarrow \boldsymbol{Z}_N^*$ the probability that $(N,g) \in \mathcal{K}_k$ is larger than $\frac{1}{k}$ it follows by a straightforward computation (Appendix A contains details) that

```
proc D̃(N, z) ≡
  g ←_R Z*_N;
  x ←_R H^{s+1}_N; b_x ←_R D(N, g, x);
  y ←_R Z*_{N^{s+1}}; b_y ←_R D(N, g, y);
  if b_x ≠ b_y then b̄_{N,g} ← b_y
                else b̄_{N,g} ←_R {0,1} fi
  b_z ←_R D(N, g, z);
  return b̄_{N,g} ⊕ b_z
```

Fig. 1. Distinguisher for $\tilde{X}_x^{s+1}$ and $\tilde{Y}_x^{s+1}$.

$$\Pr[\tilde{D}(\tilde{X}_k^{s+1}) = 1] - \Pr[\tilde{D}(\tilde{Y}_k^{s+1}) = 1] \geq \frac{1}{8k^{3c+1}} \ ,$$

contradicting that $\tilde{X}_k^{s+1}$ and $\tilde{Y}_k^{s+1}$ are computationally indistinguishable.□

Lemma 3. *For $s \geq 0$ the random variables W_k^{s+1} and X_k^{s+1} are statistically $2^{-\delta_1(k)}$-close.*

Proof: We have that $\mathrm{prf}(b,r) = (-1)^b (g^{N^s})^r \bmod N^{s+1}$. By doing a binomial expansion it is easy to see that $(N-1)^{N^s} \equiv -1 \bmod N^{s+1}$. Therefore $\mathrm{prf}(b,r) = ((N-1)^{N^s})^b (g^{N^s})^r = ((N-1)^b g^r)^{N^s} \bmod N^{s+1}$. Since $\langle -1, g\rangle = \boldsymbol{Z}_n^*$ it is therefore enough to prove that for all (N,g) random elements from $\langle g^{N^s}\rangle$ and elements from the random variable $[r \leftarrow_R \{0,1\}^{k+\delta_1(k)};\ w' \leftarrow (g^{N^s})^r \bmod N^{s+1} : w']$ are statistically $2^{-\delta_1(k)}$-close. This easily follows using a subset of the arguments used in the proof of Lemma 5 below (a similar result was proved already in [GR00]).□

Lemma 4. *If Conjecture 2 holds, then for all $s \geq 0$ the random variables V_k^{s+1} and W_k^{s+1} are computationally indistinguishable.*

Proof: Using the isomorphism from Lemma 1 it is easy to see that $V_k^{s+1} = [(N,g,v) \leftarrow_R V_k^1;\ v' \leftarrow v^{N^s} \bmod N^{s+1} : (N,g,v')]$ and $W_k^{s+1} = [(N,g,w) \leftarrow_R W_k^1;\ w' \leftarrow w^{N^s} \bmod N^{s+1} : (N,g,w')]$. Therefore if the lemma holds for $s = 0$, then it holds for all $s \geq 0$.

To prove the lemma for $s = 0$ assume that we have a distinguisher for V_k^1 and W_k^1. This immediately gives a distinguisher D for the random variables $\tilde{V}_k^1 = [(N,g) \leftarrow_R \mathcal{K}_k;\ r \leftarrow_R \{0,1\}^{\lfloor k/2 - \log(k)\rfloor};\ v \leftarrow g^r \bmod N : (N,g,v)]$ and $\tilde{W}_k^1 = [(N,g) \leftarrow_R \mathcal{K}_k;\ r \leftarrow_R \{0,1\}^{k+\delta_1(k)};\ w \leftarrow g^r \bmod N : (N,g,w)]$ with advantages $\frac{1}{k^c}$ on infinitely many k. This implies that for $\delta = 2k^c$ the subset $\mathcal{K}_k^\delta \subset \mathcal{K}_k$ of keys (N,g) for which $|\Pr[\tilde{D}(N,g,v) = 1] - \Pr[\tilde{D}(N,g,w) = 1]| \geq \frac{1}{\delta}$ has size at least $\frac{1}{\delta}$. Therefore, if given a random N as in Conjecture 2 we draw a random g in $\boldsymbol{Z}_N^*$, then $\Pr[(N,g) \in \mathcal{K}_k^\delta] > \frac{1}{2k^{c+1}}$. Since by [GR00] a pair (N,g) for which $|\Pr[\tilde{D}(N,g,v) = 1] - \Pr[\tilde{D}(N,g,w) = 1]| \geq \frac{1}{2k^{c+1}}$ allows us to factor N in probabilistic polynomial time we have that for infinitely many k we can factor N with probability larger than $\frac{1}{2k^{c+1}}$ contradicting Conjecture 2.□

Theorem 1 provides us with an efficient pseudo-random function having any desirable expansion factor without using iteration of the function. Note however, that the cost of computing the function increases super-linear in s as the modular multiplications and exponentiations are done in $\boldsymbol{Z}_{N^{s+1}}^*$. The time-optimal value of s depends on the running time of the multiplication algorithm used.

If iteration of the function is used to increase the expansion factor another problem has to be addressed. Namely, the output of the function is pseudo-random in $\boldsymbol{Z}_{N^{s+1}}^*$ and is

thus not a pseudo-random bit-string. Therefore one cannot use the output of the function as input for the next iteration. As done in [GR00,HSS93], hashing can be used to extract a pseudo-random bit-string from a pseudo-random number. However, it turns out, that a much simpler technique applies, which allows us to maintain the simplicity of the function.

Lemma 5. *For $\delta_2(k) \in [0..(k-1)]$ and $l(k) = (s+1)(k-1) - \delta_2(k)$ the random variables $[y \leftarrow_R \boldsymbol{Z}_{N^{s+1}};\ x \leftarrow y \bmod 2^{l(k)} : x]$ and $[x \leftarrow_R \{0,1\}^{l(k)} : x]$ are statistically $2^{-\delta_2(k)}$-close.*

Proof: Let $M = N^{s+1}$, and let X denote the random variable $[y \leftarrow_R \boldsymbol{Z}_M;\ x \leftarrow x \bmod 2^l : x]$. Define m and r by $M = m2^l + r$, where $0 \leq r < 2^l$. Then for $0 \leq x < r$ we have that $\Pr[X = x] = \frac{m+1}{M} \geq 2^{-l}$ and for $r \leq x < 2^l$ we have that $\Pr[X = x] = \frac{m}{M} \leq 2^{-l}$. It then follows that the statistical distance between X and the uniform distribution over $[0..2^l-1]$ is given by $\frac{1}{2}(r(\frac{m+1}{M} - 2^{-l}) + (2^l - r)(2^{-l} - \frac{m}{M})) = \frac{1}{2}(\frac{r(m+1)-(2^l-r)m}{M} + \frac{-r+(2^l-r)}{2^l}) = \frac{1}{2}(\frac{2r(m+1)-M}{M} + \frac{2^l-2r}{2^l}) = \frac{1}{2}(\frac{2^l 2r(m+1)-2^l M+M2^l-M2r}{M2^l}) = \frac{r(2^l m+2^l-M)}{M2^l} = \frac{r(2^l-r)}{M2^l} \leq \frac{2^{l-1}}{M}$, where the last inequality follows from the fact that $r2^l - r^2$ is maximal in 2^{l-1}. Now since $l = (s+1)(k-1) - \delta_2(k)$ and $M \geq 2^{(s+1)(k-1)}$ it follows that the statistical distance is bounded by $2^{-\delta_2(k)}$. result.□

Theorem 2. *For $\delta_2(k) \in [\log^2(k)..(k-1)]$ and $l(k) = (s+1)(k-1) - \delta_2(k)$ and for random (N, g) as above, the function $\mathrm{prg}^{s+1} : \{0,1\}^{\lceil k/2 - \log(k) \rceil} \to \{0,1\}^{l(k)}, (b,x) \mapsto \mathrm{prf}^{s+1}(b,x) \bmod 2^{l(k)}$ is a pseudo-random bit generator relative to Conjecture 1.*

Proof: Since $2^{-\log^2(k)}$ is negligible, this follows directly from Theorem 1 and Lemma 5. □

Depending on the choice of δ_2 the security level will be bounded by a function between $2^{-\log^2(k)}$ and 2^{-k}.

3.1 Comparison to Other Generators

In this section we compare our new pseudo-random bit generator for $s = 1$ to the two previously most efficient exponentiation modular a composite based pseudo-random bit generators.

In [GR00] it is used and proved that relative to the hardness of factoring the output of $g^x \bmod N$ for $x \in_R \{0,1\}^{k/2}$ is pseudo-random in $\langle g \rangle$. To obtain a pseudo-random bit generator they use families of hash-functions from [GW94]. The hash-function h is specified by $O(\log^2 k)$ bits and the generator is given by $GR(h,x) = (h, h(g^x \bmod N))$.

In [HSS93] it is proved and used that relative to the hardness of factoring the $k/2$ least significant bits of the function $g^x \bmod N$ are simultaneously hard. To obtain a pseudo-random bit generator they use universal hashing. The hash-function h is specified using $2k$ bits and the generator is given by $HSS(h,x) = (h, h(g^x \bmod N), x \bmod 2^{\frac{k}{2}})$. The hash-function can however be specified using just $O(\log^2 n)$ bits using the families from [GW94] yielding a larger expansion factor. However the price for this, which is also paid by GR, is a sub-exponential security. To obtain a fair comparison we will use $\delta_2(k) = \log^2(k)$ in our scheme to have the same security level.

Our generator uses the stronger Conjecture 1 to obtain a larger expansion factor. We do not use any hash-function. On the contrary we obtain a simple function specified by $\mathrm{prg}^{s+1}(b,x) = ((-1)^b g^x \bmod N^{s+1}) \bmod 2^l$, where we let g denote the preprocessed value $g^{N^s} \bmod N^{s+1}$.

In the following table we summarise the quantitative properties of the three generators.

	seed length	output length	expansion rate	mult.s	bits per mult.
HSS	$k + \log^2(k)$	$1.5k$	$\frac{1.5k}{k+\log^2(k)} \approx 1.5$	$0.5k$	$\frac{1.5k-k-\log^2(k)}{0.5k} \approx 1$
GR	$0.5k + \log^2(k)$	k	$\frac{k}{0.5k+\log^2(k)} \approx 2$	$0.25k$	$\frac{k-0.5k-\log^2(k)}{0.25k} \approx 2$
prg^2	$0.5k - \log(k)$	$2k - \log^2(k)$	$\frac{2k-\log^2(k)}{0.5k-\log(k)} \approx 4$	$0.25k$	$\frac{2k-\log^2(k)-(0.5k-\log(k)}{0.25k} \approx 6$

Our generator has twice the expansion factor of the best of the two previous generators. Further more it extracts three times as many pseudo-random bits per modular multiplication. However, it should be noted that our function uses multiplications of $2k$-bit numbers. Using a standard divide-and-conquer technique, these multiplications can be done using 3 multiplications modulo k-bit numbers in which case our generator is similar to the best of the two previous generators in bits per elementary bit operation. For more efficient multiplication algorithms our generator will be the most efficient w.r.t. to this measure.

4 Optimising Paillier's Crypto-System

In [Pai99] Paillier proposed a probabilistic public-key crypto-system semantically secure relative to Conjecture 1. The clear-text space is $\boldsymbol{Z}_N$ and the cipher-text space is $\boldsymbol{Z}^*_{N^2}$. The public key is (N, t), where N is a RSA-modulus and t is a random element of order N in $\boldsymbol{Z}^*_{N^2}$, i.e. a generator of $\boldsymbol{Z}^*_{N^2}/H^2_N$. An encryption of $x \in \boldsymbol{Z}_N$ is given by $PAI(x, r) = t^x r^N \bmod N^2$, where r is a uniformly random element in $\boldsymbol{Z}^*_N$.

In [DJ] a version of Paillier's crypto-system with smaller expansion factor was introduced. Now the clear-text space is $\boldsymbol{Z}_{N^s}$ for some integer $s \geq 1$ and the cipher-text space is $\boldsymbol{Z}^*_{N^{s+1}}$. The public key is N and an encryption is given by $DJ^{s+1}(x, r) = (N+1)^x r^{N^s} \bmod N^{s+1}$, where r is uniformly random in $\boldsymbol{Z}^*_{N^{s+1}}$. Thus DJ^2 is PAI with $t = N+1$. Now let (N, g) be chosen uniformly in $\mathcal{K}_k$. Consider the function $\widetilde{DJ}^{s+1}(x, b, r) = (N+1)^x (-1)^b (g^{N^s})^r \bmod N^{s+1}$, where r is random of length $k/2 - \log(k)$, as in our generator. We have the following result.

Theorem 3. *The function $\widetilde{DJ}^{s+1}(x, b, r)$ as defined above is a semantically secure public-key crypto-system relative to Conjecture 1.*

Proof: It is easy to see that an adversary breaking the semantic security of this system, can also break our pseudo-random generator. More formally, by the definition of semantic security we have to prove that for all PPT algorithms A, which on a key (N, g) returns a message $m \in \boldsymbol{Z}_{N^s}$ and some state information $s \in \{0,1\}^*$, the two random variables $U^{s+1}_k = [(N, g) \leftarrow_R \mathcal{K}_k;\ (s, m) \leftarrow_R A(N, g);\ c \leftarrow_R \boldsymbol{Z}^*_{N^{s+1}} : (N, g, s, m, c)]$ and $T^{s+1}_k = [(N, g) \leftarrow_R \mathcal{K}_k;\ (s, m) \leftarrow_R A(N, g);\ c \leftarrow_R \widetilde{DJ}(m) : (N, g, s, m, c)]$ are computationally indistinguishable. Let however $\lambda(N, g, z)$ denote the following PPT algorithm $(N, g, z) \mapsto [(s, m) \leftarrow_R A(N, g);\ c \leftarrow (N+1)^m z : (N, g, s, m, c)]$. Then it is easy to see that $\lambda(Y^{s+1}_k) = U^{s+1}_k$ and $\lambda(V^{s+1}_k) = T^{s+1}_k$ and thus the theorem is a corollary to the proof of Theorem 1.□

In the function DJ^{s+1}, the part $(N+1)^x \bmod N^{s+1}$ can be simplified, and can be computed using $2s$ multiplications (see [DJ]). Hence we need on average $2s + 1.5sk$ multiplications using square and multiply techniques for r^{N^s} and $(s+1)k$ random bits. The modified encryption function $\widetilde{DJ}^{s+1}$ uses $k/2$ random bits and on average $2s + 3k/4$ multiplications using preprocessing[2]. Thus, measured in usage of random bits and multiplications, the modified scheme is better by a factor of about $2s$.

[2] Since both $(N+1)$ and g are known in advance.

5 Secure Constant-Round Multiparty Function Evaluation

In this section we describe how our pseudo-random function can be used to construct a secure constant-round protocol for evaluating any boolean circuit.

First we are going to construct from our pseudo-random generator a special kind of symmetric encryption $C = \mathcal{E}_{N,g}(K, M)$ called a Crypto-Table, where $M = (M_{0,0}, M_{0,1}, M_{1,0}, M_{1,1})$ are four messages to be encrypted, $K = (K^{1,0}, K^{1,1}, K^{2,0}, K^{2,1})$ are four keys used for encrypting, and where C is such that one can learn M_{b_1,b_2} iff one knows the keys (K^{1,b_1}, K^{2,b_2}). We think of this as a kind of encrypted Boolean gate where the keys represent input bits b_1 on line 1 and input b_2 on line 2 and M_{b_1,b_2} represents the output.

Let $l(k) = k/2 - 2$, $K^{c,b} \in \{0,1\}^{l(k)}$ and $M_{b_1,b_2} \in \{0,1\}^{l(k)+1}$. Sets U, V will be the index sets for messages and keys, respectively: $U = \{0,1\}^2$, $V = \{1,2\} \times \{0,1\}$. The symbol $\text{needs}(b_1, b_2) = \{(1, b_1), (2, b_2)\}$ will denote the indexes of the keys one needs in order to learn M_{b_1,b_2} from the encryption. For $V' \subset V$ let $\text{visi}(V') = \{u \in U | \, \text{needs}(u) \subset V'\}$, i.e. $\text{visi}(V')$ are the indexes of messages that are "visible" if one knows keys with indexes in V'. Finally, for $v \in V$ let $\text{hides}(v) = \{u | v \in \text{needs}(u)\}$.

To encrypt we start by doing a 2-out-of-2 secret-sharing for each M_{b_1,b_2}: Let $M^{1,b_1}_{b_1,b_2} \leftarrow_R \{0,1\}^{2/3k}$ and let $M^{2,b_2}_{b_1,b_2} \leftarrow M_{b_1,b_2} + M^{1,b_1}_{b_1,b_2}$. Then for $v \in V$, we construct M^v, a string that we will encrypt with key K^v: $M^v = 2^k M^v_{u0} + M^v_{u1}$ where $\text{hides}(v) = \{u_0, u_1\}$. In the following for $V' \subset V$ we let $M^{V'} = \{M^v | v \in V'\}$ and let $K^{V'} = \{K^v | v \in V'\}$. Observe that if one knows $M^{\text{needs}(b_1,b_2)}$, then one especially knows $M^{1,b_1}_{b_1,b_2}, M^{2,b_2}_{b_1,b_2}$ and can compute $M_{b_1,b_2} = M^{2,b_2}_{b_1,b_2} - M^{1,b_1}_{b_1,b_2}$. The following lemma implies that this is all you can learn from $M^{\text{needs}(b_1,b_2)}$.

Lemma 6. *Let V' be any subset of key indices and let M and $\overline{M}$ by any two messages for which $M_u = \overline{M}_u$ for $u \in \text{visi}(V')$. Then $M^{V'}$ and $\overline{M}^{V'}$ are statistically indistinguishable.*

Proof: If $u \in \text{visi}(V')$, then $M_u = \overline{M}_u$ and thus M^v_u and $\overline{M}^v_u$ for $v \in \text{needs}(u)$ are identically distributed. If $u \notin \text{visi}(V')$, then M^v_u, for some $v \in \text{needs}(u)$, is not given by $M^{V'}$ and thus the remaining $M^{v'}_u$ (for $v' \in \text{needs}(u) \setminus \{v\}$) is distributed statistically indistinguishable from a random element from $\{0,1\}^{2/3k}$. The same holds for $\overline{M}^{j'}_i$ and the lemma follows.□

Now to finish the encryption, for $v \in V$ let $C^v \leftarrow \text{prg}^2(K^v) + M^v \bmod N^2$ and let $C = \mathcal{E}_{N,g}(K, M) = \{C^v\}_{v \in V}$.

Lemma 7. *Let V' be any subset of key indices and let M and $\overline{M}$ by any two messages for which $M_u = \overline{M}_u$ for $u \in \text{visi}(V')$. Then the random variables $[(N, g) \leftarrow_R \mathcal{K}_k;\ K \leftarrow_R \{0,1\}^{4l(k)};\ C \leftarrow \mathcal{E}_{N,g}(K, M) : (C, K^{V'})]$ and $[(N, g) \leftarrow_R \mathcal{K}_k;\ K \leftarrow_R \{0,1\}^{4l(k)};\ \overline{C} \leftarrow \mathcal{E}_{N,g}(K, \overline{M}) : (\overline{C}, K^{V'})]$ are computationally indistinguishable.*

Proof: If the values $\text{prg}^2(K^v)$ for $v \in V$ had been uniformly random in $\boldsymbol{Z}_{N^2}$ and not just pseudo-random, the lemma would follow directly from the above lemma. Therefore the lemma follows by a hybrid argument using Lemma 2 four times.□

We now describe how to put Crypto-Tables together to obtain a so-called Crypto-Circuit. We only use binary Crypto-Tables, however the above construction generalises to unary tables and tables with higher fane-in by similar definitions of U, V, and needs.

Let $G = (G_1, \ldots, G_\alpha, G_{\alpha+1}, \ldots G_\beta)$ be a circuit given by α input gates and $\beta - \alpha$ binary boolean gates. The input gates are only syntax. The binary gates G_i are tuples $(i_1, i_2, G_{i,0,0}, G_{i,0,1}, G_{i,1,0}, G_{i,1,1})$, where i_1 and i_2 specifies that G_{i_1} and G_{i_2} is the input

gates for gate G_i and $G_{i,b_1,b_2} \in \{0,1\}$ specifies the boolean gate. Let $G_{\beta-\gamma+1}, \ldots, G_\beta$ be the output gates, i.e. those that are not used as input to other gates.

Given an input $x \in \{0,1\}^\alpha$ let for input gates $\text{out}(G_i, x) = x_i$ and for binary gates inductively $\text{out}(G_i, x) = G_{i,\text{out}(G_{i_1},x),\text{out}(G_{i_2},x)}$. Let $\text{out}(G, x) = (\text{out}(G_1, x), \ldots, \text{out}(G_\beta, x))$ and let $\text{eval}(G, x) = (\text{out}(G_{\beta-\gamma+1}), \ldots, G_\beta)$.

Given a circuit G and a key $K = (K^{ib} | i \in [1..\beta], b \in \{0,1\}) \in \{0,1\}^{2\beta l(k)}$, we define the Crypto-Circuit $\mathcal{E}_{N,g}(K, G)$ as follows. For each binary gate G_i and $(b_1, b_2) \in \{0,1\}^2$ let $M_{i,b_1,b_2} = (G_{i,b_1,b_2}, K_{i,G_{i,b_1,b_2}})$, let $M_i = (M_{i,0,0}, M_{i,0,1}, M_{i,1,0}, M_{i,1,1})$, let $K_i = (K_{i_1,0}, K_{i_1,1}, K_{i_2,0}, K_{i_2,1})$, and let $C_i = \mathcal{E}_{N,g}(K_i, M_i)$. Let $\mathcal{E}_{N,g}(K, G) = C = (C_{\alpha+1}, \ldots, C_\beta)$. For $x \in \{0,1\}^\alpha$ let $K_x = ((x_1, K^{1,x_1}), \ldots, (x_\alpha, K^{\alpha,x_\alpha}))$ and for $C = \mathcal{E}_{N,g}(K, G)$ let $C_x = (C, K_x)$.

proc $\mathcal{E}_{N,g}(K, M) \equiv$
  for $(b_1, b_2) \in \{0,1\}^2$ **do**
    $M^{1,b_1}_{b_1,b_2} \leftarrow_R \{0,1\}^{2/3k}$;
    $M^{2,b_2}_{b_1,b_2} \leftarrow M_{b_1,b_2} + M^{1,b_1}_{b_1,b_2}$ **od**;
  $C^{1,0} \leftarrow \text{prf}^2_{N,g}(K^{1,0}) + (2^k M^{1,0}_{0,0} + M^{1,0}_{0,1}) \bmod N^2$;
  $C^{1,1} \leftarrow \text{prf}^2_{N,g}(K^{1,1}) + (2^k M^{1,1}_{1,0} + M^{1,1}_{1,1}) \bmod N^2$;
  $C^{2,0} \leftarrow \text{prf}^2_{N,g}(K^{2,0}) + (2^k M^{2,0}_{0,0} + M^{2,0}_{1,0}) \bmod N^2$;
  $C^{2,1} \leftarrow \text{prf}^2_{N,g}(K^{2,1}) + (2^k M^{2,1}_{0,1} + M^{2,1}_{1,1}) \bmod N^2$

Fig. 2. The Crypto-Table Generator.

Now for $x \in \{0,1\}^\alpha$ assume that C_x is known. Then one knows $(\text{out}(G_i, x), K^{i,\text{out}(G_i,x)})$ for all input gates. Assume then inductively for $G_i = (i_1, i_2, G_{i,0,0}, G_{i,0,1}, G_{i,1,0}, G_{i,1,1})$. that one knows (b_1, K^{i_1,b_1}) for $b_1 = \text{out}(G_{i_1}, x)$ and (b_2, K^{i_2,b_2}) for $b_2 = \text{out}(G_{i_2}, x)$. Then by looking up in C^i one can learn $(c, K^{i,c})$ for $c = G_{i,b_1,b_2} = G_{i,\text{out}(G_{i_1},x),\text{out}(G_{i_2},x)} = \text{out}(G_i, x)$. Therefore there exists a PPT algorithm $\mathcal{D}$ such that $\mathcal{D}_{N,g}(C_x) = \text{out}(G, x)$ (or $\mathcal{D}_{N,g}(C_x) = \text{eval}(G, x)$ depending on the context).

The value $\text{out}(G, x) = (o_1, \ldots, o_\beta)$ contains information about x and one row of each gate. We want to prove that given C_x one only learns this information. We do this by simulating the distribution of C_x given just $o = \text{out}(G, x)$. Let $\text{sim}(o)$ be the PPT algorithm working as follows. Given o define a circuit G^o. For $i = \alpha + 1, \ldots, \beta$ set $G^o_{i,o_{i_1},o_{i_2}} = o_i$ and set the remaining three rows $(b_1, b_2) \neq (o_{i_1}, o_{i_2})$ to 0. Compute x as $o_1 \ldots o_\alpha$ and let $C^o_x = (\mathcal{E}_{N,g}(K, G^o), K_x)$.

Lemma 8. *Let G be any circuit and let x by any input for the circuit. Then $[(N, g) \leftarrow_R \mathcal{K}_k;\ K \leftarrow_R \{0,1\}^{2\beta l(k)};\ C_x \leftarrow (\mathcal{E}_{N,g}(K, G), K_x) : (N, g, C_x)]$ and $[(N, g) \leftarrow_R \mathcal{K}_k;\ o \leftarrow \text{out}(G, x);\ C^o_x \leftarrow \text{sim}_{N,g}(o) : (N, g, C^o_x)]$ are computationally indistinguishable.*

Proof: Assume for the sake of contradiction that there exists G, x and D such that the distributions are distinguishable by D. Let for $i = \alpha, \ldots, \beta$ the circuit G^i be the one where $(G^i_1, \ldots, G^i_i) = (G_1, \ldots, G_i)$ and $(G^i_{i+1}, \ldots, G^i_\beta) = (G^o_i, \ldots, G^o_\beta)$. Then $G^\alpha = G^o$ and $G^\beta = G$ and thus there exists $i \in [(\alpha + 1)..\beta]$ such that D distinguishes $[(N, g) \leftarrow_R \mathcal{K}_k;\ K \leftarrow_R \{0,1\}^{2\beta l(k)};\ C^{i-1}_x \leftarrow (\mathcal{E}_{N,g}(K, G^{i-1}, x), K_x) : (N, g, C^{i-1}_x)]$ and $[(N, g) \leftarrow_R \mathcal{K}_k;\ K \leftarrow_R \{0,1\}^{2\beta l(k)};\ C^i_x \leftarrow (\mathcal{E}_{N,g}(K, G^i, x), K_x) : (N, g, C^i_x)]$.

It is easy to see that $\mathcal{D}_{N,g}(C^{i-1}_x) = o = \mathcal{D}_{N,g}(C^i_x)$. Therefore the values o_{i_0}, o_{i_1}, and o_i is the same in both distributions. Now let $((o_{i_1}, K_{i_1,o_{i_1}}), (o_{i_2}, K_{i_2,o_{i_2}}), \tilde{C}_i)$ be any Crypto-Table with lookup-key for which $M_{i,o_{i_1},o_{i_2}} = (o_i, K_{i,o_i})$ and $K_{i_1,o_{i_1}}$, $K_{i_2,o_{i_2}}$, and K_{i,o_i} are uniformly random values. Define a distribution as follows. Compute G^i and let $C^i \leftarrow \mathcal{E}_{N,g}(G^i)$ as above with the sole exception that the values $K_{i_1,o_{i_1}}$, $K_{i_2,o_{i_2}}$, and K_{i,o_i} are used at the appropriate points in the computation instead of new uniformly random values. This obviously does not change the distribution of C^i. Now let $\tilde{C}^i$ be C^i where C_i is replace by $\tilde{C}_i$. Then if $\tilde{C}_i$ is computed as in $\mathcal{E}_{N,g}(G^{i-1})$, then $\tilde{C}^i$ is distributed exactly

as $\mathcal{E}_{N,g}(G^{i-1})$ and if $\tilde{C}_i$ is computed as in $\mathcal{E}_{N,g}(G^i)$, then $\tilde{C}^i$ is distributed exactly as $\mathcal{E}_{N,g}(G^i)$ and we can then use D to distinguish these two distributions of $\tilde{C}^i$. But since the difference between the two distributions is only the values of G_{i,b_1,b_2} for $(b_1,b_2) \neq (o_{i_0}, o_{i_1})$ this contradicts Lemma 7.□

The value of $\text{out}(G,x)$ reveals the output from every gate, but no more information about gates than that. In the multiparty function evaluation problem we only want to learn $\text{eval}(G,x)$ and do not care whether the actual gates are known (typically the circuit is agreed upon by the parties before the computation and it is the inputs that should be kept secret). It therefore seems, that the function $\text{out}(G,x)$ is doing exactly the opposite of what we want in the multiparty function evaluation problem. However, there exists a simple reduction $(G',x') \leftarrow_R \text{scram}(G,x)$ which allows us to evaluate $f(x) = \text{eval}(G,x)$ secretly using $\text{out}(G',x')$.

Given (G,x) we define $(G',x') \leftarrow_R \text{scram}(G,x)$ as follows. For the gates $G_1,\ldots,G_{\beta-\gamma}$ draw a uniformly random bit λ_i and for each output gate G_i let $\lambda_i = 0$. For each input gate let $G'_i = G_i$ and let $x'_i = x_i \oplus \lambda_i$ and for each binary gate $G_i = (i_1, i_2, G_{i,0,0}, G_{i,0,1}, G_{i,1,0}, G_{i,1,1})$ let $G'_i = (i_1, i_2, G'_{i,0,0}, G'_{i,0,1}, G'_{i,1,0}, G'_{i,1,1})$, where $G'_{i,b_1,b_2} = G_{i,b_1\oplus\lambda_{i_1},b_2\oplus\lambda_{i_2}} \oplus \lambda_i$. The following lemma follows directly from the definitions.

Lemma 9. *For all circuits G, all input x to the circuit, and $(G',x') \leftarrow_R \text{scram}(G,x)$, it holds that $\text{out}(G'_i,x') = \text{out}(G_i,x) \oplus \lambda_i$, where λ_i is the value used in $\text{scram}(G,x)$.*

Now define a PPT algorithm by $R_{N,g}(G,x) = [(G',x') \leftarrow_R \text{scram}(G,x);\ K \leftarrow_R \{0,1\}^{2\beta l(k)};\ R_x = (\mathcal{E}_{N,g}(K,G'),K_{x'}) : R_x]$. Then by Lemma 9 and the fact that $\lambda_i = 0$ for output gates it follows that one can compute $\text{eval}(G,x)$ from R_x. We want to prove that from $R_{N,g}(G,x)$ one only learns $y = \text{eval}(G,x)$. For this purpose we describe a PPT function $\text{sim}_{N,g}(G,y)$. The algorithm proceeds as follows. Let $o_{\beta-\gamma+i} = y_i$ for $i = 1,\ldots,\gamma$, let $o_i \leftarrow_R \{0,1\}$ for $i = 1,\ldots,\beta-\gamma$, and let $C_o \leftarrow_R \text{sim}_{N,g}(o)$.

Lemma 10. *Let G be a any circuit and let x be any input. Then $[(N,g) \leftarrow_R \mathcal{K}_k;\ R_x \leftarrow_R R_{N,g}(G,x) : (N,g,R_y)]$ and $[(N,g) \leftarrow_R \mathcal{K}_k;\ y = \text{eval}(G,x);\ R_y \leftarrow_R \text{sim}_{N,g}(G,y) : (N,g,R_y)]$ are computationally indistinguishable.*

Proof: By Lemma 8 the random variables $[(N,g) \leftarrow_R \mathcal{K}_k;\ C_x \leftarrow_R \mathcal{E}_{N,g}(\text{scram}(G,x)) : (N,g,C_x)]$ and $[(N,g) \leftarrow_R \mathcal{K}_k;\ o \leftarrow_R \text{out}(\text{scram}(G,x));\ C_o \leftarrow_R \text{sim}_{N,g}(o) : (N,g,C_0)$ are computationally indistinguishable. By Lemma 9 we have that $\text{out}(\text{scram}(G,x)) = (\text{out}(G_1,x_0) \oplus \lambda_1,\ldots,\text{out}(G_\beta,x_\beta) \oplus \lambda_\beta)$. Since the λ_i for inputs gates and intermediate gates are uniformly distributed and is 0 for output gates it is trivial that $\text{out}(\text{scram}(G,x))$ is distributed *identically* to o as defined as part of the $\text{sim}_{N,g}(G,y)$ algorithm. Therefore the random variables $[(N,g) \leftarrow_R \mathcal{K}_k;\ o \leftarrow_R \text{out}(\text{scram}(G,x));\ C_o \leftarrow_R \text{sim}_{N,g}(o) : (N,g,C_0)]$ and $[(N,g) \leftarrow_R \mathcal{K}_k;\ y \leftarrow \text{eval}(G,x);\ R_y \leftarrow_R \text{sim}_{N,g}(y) : (N,g,R_y)]$ are identically distributed. Combining these two result the lemma follows directly. □

Assume now that we are executing in an environment, where n parties, party i having secret input $x_i \in \{0,1\}^*$, have access to a authenticated broadcast channel and that they want to evaluate some fixed agreed upon circuit G on their inputs in such a way, that all parties learn $y \leftarrow \text{eval}(G,x_1,\ldots,x_n)$, but no party learns anything about the other parties' inputs, except that which is evident from y. Assume further more, that a key $(N,g) \leftarrow_R \mathcal{K}_k$ is known to and agreed upon by all parties and that the parties have access to a trusted party $\mathcal{T}_R$ for computing $R_{N,g}$. All parties are assumed to share a perfectly secure point-to-point channel with $\mathcal{T}$. Then they can evaluate G on x as follows.

1. Every party P_i sends his input x_i to $\mathcal{T}_R$ over the secure point-to-point channel.

2. $\mathcal{T}_R$ computes $R_x \leftarrow_R R_{N,g}(G, x_1, \ldots, x_n)$ and broadcast the value on the public channel.
3. Each party locally computes $y = \text{eval}(G, x_1, \ldots, x_n)$ from R_x and outputs y.

We are going to prove that the protocol securely evaluates the function $f(x_1, \ldots, x_n) = \text{eval}(G, x_1, \ldots, x_n)$ in the sense of [Can00] against any PPT static and active adversary $\mathcal{A}$. To formalise this we need to define two distribution ensembles.

Denote the above protocol by π. Let $x_1, \ldots, x_n$ be any input for the protocol, let $\mathcal{A}$ be any PPT algorithm, called the π-adversary, and let C be any subset of the parties, called the corrupted parties[3]. Then an attack by $\mathcal{A}$ on the execution of π proceeds as follows. First $\mathcal{A}$ receives the inputs $\{x_i\}_{i \in C}$ of the corrupted parties. Then the adversary fully controls the corrupted parties. In the above one-round protocol this amounts to sending faulty inputs $\{x'_i\}_{i \in C}$ to $\mathcal{T}_R$. Then $\mathcal{T}_R$ computes $R_{x'} \leftarrow_R R_{N,g}(G, x'_1, \ldots, x'_n)$, where $x'_i = x_i$ for the uncorrupted parties, and broadcasts $R_{x'}$. After seeing this value $\mathcal{A}$ outputs some value a and the output of the protocol is taken to be y' as computed from $R_{x'}$ (observe that $y' = \text{eval}(G, x'_1, \cdots, x'_n)$). Let $\text{Exec}_\pi(x_1, \ldots, x_n, \mathcal{A}, C) = (y', a)$.

Assume instead that the parties had access to a trusted party for computing $f(x) = \text{eval}(G, x)$ directly. The protocol, called the ideal protocol, would then have been.

1. Every party P_i sends his input x_i to $\mathcal{T}_f$ over the secure point-to-point channel.
2. $\mathcal{T}_f$ computes $y \leftarrow f(x_1, \ldots, x_n)$ and broadcast the value on the public channel.
3. Each party outputs y.

An adversary $\mathcal{S}$, called an ideal adversary, would in an attack against the ideal evaluation have the power to send faulty inputs to $\mathcal{T}_f$ and nothing else. After seeing y' the ideal adversary then outputs some value s and we let $\text{Exec}_f(x_1, \ldots, x_n, \mathcal{S}, C) = (y', s)$.

To prove π secure in the sense of [Can00] we need to prove that for every π-adversary $\mathcal{A}$ there exists an ideal adversary $\mathcal{S}$ such that for every input $x_1, \ldots, x_n$ and every set of corrupted parties C the random variables $\text{Exec}_\pi(x_1, \ldots, x_n, \mathcal{A}, C)$ and $\text{Exec}_f(x_1, \ldots, x_n, \mathcal{S}, C)$ are computationally indistinguishable. To prove this, for any π-adversary $\mathcal{A}$ let $\mathcal{S}(\mathcal{A})$ be the following ideal-adversary.

1. Get $(C, \{x_i\}_{i \in C})$ as input and run $\mathcal{A}$ as a sub-routine initialised with $\{x_i\}_{i \in C}, C$ as it would be in an attack on the execution of π.
2. Receive $\{x'_i\}_{i \in C}$ from $\mathcal{A}$ and use these values in the oracle call to compute $y' = f(x'_1, \ldots, x'_n)$, where $x'_i = x_i$ for $i \notin C$.
3. From y' compute $R_{y'} \leftarrow_R \text{sim}_{N,g}(G, y')$ and hand $R_{y'}$ to $\mathcal{A}$ (claiming that it is the value broadcast by $\mathcal{T}_R$ in an execution of π).
4. Let a be the output of $\mathcal{A}$ after seeing $R_{y'}$ and let $s = a$ be the output of $\mathcal{S}(\mathcal{A})$.

Then by Lemma 10 the random variables $\text{Exec}_\pi(x_1, \ldots, x_n, \mathcal{A}, C)$ and $\text{Exec}_f(x_1, \ldots, x_n, \mathcal{S}(\mathcal{A}), C)$ are computationally indistinguishable and we have thus proven the following theorem.

Theorem 4. *For any circuit G, the protocol π securely (relative to Conjecture 1), in the sense of of [Can00], computes the function $f(x) = \text{eval}(G, x)$ in the presence of static and active adversaries corrupting any subset of the parties.*

We now turn our attention to computing $R_{N,g}$ securely. Here we will appeal to a result from [CDN00], which constructs a general secure protocol FuncEval for function evaluation. It is shown there that such a protocol can be constructed based on any secure threshold

[3] The trusted party $\mathcal{T}_R$ cannot be corrupted.

public-key cryptosystem with a number of additional algebraic properties. They assume (as we also do here) that keys for this system have been set up in advance, by a trusted initialiser or by a once-and-for-all multiparty computation. We will use the Damgård-Jurik generalisation of Paillier as this threshold crypto-system, with $s = 2$ so that the public key is N, computations take place mod N^3, and the plaintext space is Z_{N^2} (DJ^3 denotes this system, see Section 4). It is shown in [CDN00] that DJ^3 has the required properties. We will assume that key generation takes place by sampling from $\mathcal{K}_k$ and including both N and g in the public key. This is necessary, although DJ^3 actually only needs N as public key: we will be computing $R_{N,g}$, so all players need to know the constant g. In the same way as in Theorem 3, it can be shown that making g public does not harm the security of DJ^3. The following can now be easily abstracted from the results in [CDN00]:

Theorem 5 ([CDN00]). *Let F be any arithmetic circuit over $\boldsymbol{Z}_{N^2}$, where N is a uniformly random RSA-modulus and the circuit contains the following types of gates: 1) input gates of elements from $\boldsymbol{Z}_{N^2}$, 2) input gates of elements from $\{0,1\}$, 3) random gates outputting a uniformly random $\boldsymbol{Z}_{N^2}$-element, 4) random gates outputting 0 or 1 with probability $\frac{1}{2}$, 5) addition gates, 6) multiplication-with-constant gates ($x \mapsto a \cdot x \bmod N^2$), 7) multiplication gates ($(x,y) \mapsto x \cdot y \bmod N^2$), 8) inversion gates[4] ($x \mapsto x^{-1} \bmod N^2$), and 9) output gates. Let $f : (\boldsymbol{Z}_{N^2})^\alpha \to (\boldsymbol{Z}_{N^2})^\gamma$ be the function specified by the circuit. Then the protocol* FuncEval_F *securely evaluates the function f in the presence of static and active adversaries corrupting any minority of the parties. The communication complexity is $O(k(\alpha + \min(k,n)(\rho_3 + n\rho_4) + n\beta))$, where α is the number of gates of type 1 and 2, ρ_3 is the number of gates of type 3, ρ_4 is the number of gates of type 4, and β is the number of gates of type 7, 8, and 9. The round complexity is $O(\delta)$, where δ is the number of gates of type 7, 8, and 9 in the circuit, where an unbounded fane-in multiplication $(x_1, \ldots, x_m) \mapsto \prod_{i=1}^m x_i \bmod N^2$ of invertible elements only counts for one gate.*

Our goal is to specify $R_{N,g}$ as a constant-depth circuit over $\boldsymbol{Z}_{N^2}$ with gates of the appropriate types and then appeal to the above theorem. Before doing so we sketch how the FuncEval protocol works.

Assume for notational reasons, that party i has just one input $x_i \in \boldsymbol{Z}_{N^2}$. The protocol works as follows. First each party publishes an encryption $\overline{x_i} = DJ^3(x_i, r_i)$ in the Damgård-Jurik crypto-system of his input and prove knowledge of x_i using a zero-knowledge proof of knowledge. If an input is required to be from $\{0,1\}$ the party supplying the input proves in zero-knowledge that this is the case. Addition gates are handled using the additive homomorphism $DJ^3(x_1, r_1) \cdot DJ^3(x_2, r_2) \equiv (N+1)^{x_1} r_1^{N^2} (N+1)^{x_2} r_2^{N^2} \equiv (N+1)^{x_1 + x_2 \bmod N^2} (r_1 r_2 \bmod N)^{N^2} \equiv DJ^3(x_1 + x_2 \bmod N^2, r_1 r_2 \bmod N) \pmod{N^3}$ and can thus be computed locally by the parties (we write $\overline{x_1 + x_2} = \overline{x_1} \boxplus \overline{x_2}$). In the same way $DJ^3(x,r)^c \equiv DJ^3(cx \bmod N^2, r^c \bmod N) \pmod{N^3}$ handles multiplications by constants ($\overline{c \cdot x} = c \boxdot \overline{x}$). The remaining types of gates require interaction and are handled by secure protocols described in [CDN00].

To specify a circuit for $\mathcal{E}_{N,g}(G,x)$ we generate λ_i using random 0/1 gates, we generate $K^{c,b} \in \{0,1\}^{l(k)}$ using $l(k)$ random 0/1 gates for $K_0^{c,b}, \ldots, K_{l(k)_1}^{c,b}$ and then let $K^{c,b} = \sum_{i=0}^{l(k)-1} 2^i K_i^{c,b}$ (using $l(k)$ multiplications with constants and $l(k) - 1$ additions). We generate $M_{b_1,b_2}^{1,b_1} \in \{0,1\}^{\lceil 2/3k \rceil}$ in the same way. The cost of this is $O(k \min(n,k) nk)$ per gate and the round complexity is $O(1)$ as all the bits for all the gates can be generated in parallel.

[4] The circuit must be such that the input to an inversion gate is always invertible.

Given input-lines b and $r_0, \ldots, r_{l(k)-2}$ with b and the bits of r we compute $(-1)^b g^r$ as follows. Let $(-1)^b = 1 - 2 \cdot b \bmod N^2$, let $g^r = \prod_{i=0}^{l(k)-2}(g^{2^i} \cdot r_i + (1 - r_i)) \bmod N^2$ using unbounded fane-in multiplication, and multiply to obtain $(-1)^b g^r \bmod N^2$. The communication complexity of this is $O(k^2 n)$ bits per gate and the round complexity is $O(1)$ as the elements $g^{2^i} r_i + (1 - r_i) \equiv g^{r_i 2^i} \pmod{N^2}$ are guaranteed to be invertible.

After having all these values set up for each gate we can easily compute the scrambling and $\mathcal{E}_{N,g}(K, M)$ using only a constant number of additions and multiplications modulo N^2 per gate. Therefore the communication complexity is $O(k \min(n, k)nk)$ bits per gate and the round complexity is $O(1)$.

Theorem 6. *Given any circuit G we can efficiently generate a constant-round protocol, which securely evaluates the function $(x_1, \ldots, x_n) \mapsto \mathrm{eval}(G, x_1, \ldots, x_n)$ in the presence of static and active adversaries corrupting any minority of the parties. The communication complexity is $O(\min(n, k)nk^2\beta)$, where β is the number of gates in the circuit.*

Proof: By Theorem 4 we can construct such a protocol given an oracle for computing $R_{g,N}$. By the above discussion we can implement a protocol for $R_{g,N}$ securely in the presence of static and active adversaries corrupting any minority of the parties. Therefore by the composition theorem of [Can00] by replacing the oracle for $R_{g,N}$ by the implementation, the overall protocol is secure. □

5.1 An optimisation

The above protocol obtained a communication complexity in the order of $\min(n, k)nk^2$ bits per gate. We will here sketch a further optimisation allowing us to use only in the order of nk^2 bits per gate.

The expensive part of the above protocol is generating K and M and computing $\mathrm{prf}^2(K)$. The reason is the large number of random bits generated. Before sketching the optimisation it is illustrative to see why generating the random bits is expensive.

In [CDN00] a subset of the parties containing at least one honest party (except with negligible probability) is picked for generating random values. If $n < k$ the group is just all the parties. If $n > k$ the group is a $O(k)$ random subset, which contains a honest party except with negligible probability as at most half the parties are corrupted. Note that the size of the random group is $O(\min(n, k))$. Assume that the parties in the random group are indexed $1, \ldots, r(k, n)$.

A secret and random element from $\boldsymbol{Z}_{N^2}$ is then generated as follows. Each party in the random group generates a random element $x_i \in \boldsymbol{Z}_{N^2}$, broadcasts an encryption $\overline{x_i}$, and proves in zero-knowledge that he knows x_i. Then all parties compute $\overline{x} = \boxplus_{i=1}^{r(k,n)} \overline{x_i} \bmod N^2$. Since the random group contains at least one honest party except with negligible probability the secret value x will be uniformly random in $\boldsymbol{Z}_{N^2}$ and unknown to all the parties and the adversary. The communication complexity of this is $O(r(k, n)k)$ bits as each encryption is $O(k)$ bits long and the proofs of knowledge use $O(k)$ bits of communication.

A random bit is generated as follows. Each party in the random group picks a random bit b_i, publishes an encryption $\overline{b_i}$ of the bit, and proves that it is either 0 or 1; this costs $O(r(k, n)k)$ bits of communication. All n parties then computes $\overline{b_1 \oplus \cdots \oplus b_{r(k,n)}} = ((\boxdot_{i=1}^{r(k,n)}(2 \boxdot \overline{b_i} \boxminus 1)) \boxplus 1) \boxdot 2^{-1} \bmod N^2$ using FuncEval. Since at least one party in the group is honest (except with negligible probability) the resulting secret bit b is uniformly random. Using an unbounded multiplication the round complexity is $O(1)$ and the communication complexity is $O(r(k, n)nk)$ for generating just one bit.

To avoid generating these communication expensive bits we use the idea at a higher level. Each party in the random group picks a random key $(b_i, r_i) \in \{0,1\}^{l(k)}$, computes $\mathrm{prf}^2(b_i, r_i) = (-1)^{b_i} g^{r_i} \bmod N^2$, broadcasts $\overline{b_i}, \overline{r_{i,0}} \ldots, \overline{r_{i,l(k)-2}}$ and

$$\overline{\mathrm{prf}^2_{N,g}(b_i, r_i)} = (1 \boxminus 2 \boxdot \overline{b_i}) \boxdot \left[\boxdot_{j=1}^{l(k)-2} \left(g^{2^j} \boxdot \overline{r_{i,j}} \boxplus (1 \boxminus \overline{r_{i,j}}) \right) \right] \bmod N^2 \qquad (1)$$

and proves that $\overline{\mathrm{prf}^2_{N,g}(b_i, r_i)}$ was computed correctly. This costs $O(k^2)$ bits per party in the random group as the proof of correct computation can be done using $O(k)$ bits per gate in the circuit for (1) using the techniques of [CDN00]. Then all the parties compute $\overline{b} = \boxplus_{i=1}^{r(k,n)} \overline{b_i}$, $\overline{r} = \boxplus_{i=1}^{r(k,n)} \overline{r_i}$, and $\overline{y} = \boxdot_{i=1}^{r(k,n)} \overline{\mathrm{prf}^2(b_i, r_i)} = (-1^{\overline{b}} g^{\overline{r}}) \bmod N^2$. The cost of this is $O(r(k,n)nk)$. The total cost is therefore $O(r(k,n)k^2 + r(k,n)nk) = O(r(k,n)\max(n,k)k)$. Since $r(k,n) = \min(n,k)$ it then follows that the total number of bits broadcast is $O(nk^2)$.

Now as at least one of the parties of the random group is honest y will be the encryption of a product of a pseudo-random element from $\boldsymbol{Z}_{N^2}$ (the value of $\mathrm{prf}^2(b_i, r_i)$ contributed by the honest party) and an invertible element from $\boldsymbol{Z}_{N^2}$ (the product of the invertible elements $\mathrm{prf}^2(b_i, r_i)$ contributed by the possible corrupted parties). Therefore y will be an encryption of a pseudo-random element from $\boldsymbol{Z}_{N^2}$ and thus has exactly the same distributed as before. We then code the values b and r as a $(\lceil k/2 \rceil + 2\lceil \log(r(k,n)) \rceil)$-bit number by $\overline{K} = (2^{\lceil k/2 \rceil + \lceil \log(r(k,n)) \rceil} \boxdot \overline{b} \boxplus \overline{r}) \bmod N^2$. Now since $r(k,n) \in O(k)$, if M_{b_1,b_2} is an $(\lceil k/2 \rceil + 2\lceil \log(r(k,n)) \rceil)$-bit number and $M^{1,b_1}_{b_1,b_2}$ is a uniformly random $\lceil 2/3k \rceil$-bit number, the value $M^{2,b_2}_{b_1,b_2} = M^{1,b_1}_{b_1,b_2} + M_{b_1,b_2} \pmod{N^2}$ hides the value of M_{b_1,b_2} as $M^{2,b_2}_{b_1,b_2} = M^{1,b_1}_{b_1,b_2} + M_{b_1,b_2}$ and $M^{2,b_2}_{b_1,b_2} = M^{1,b_1}_{b_1,b_2} + M'_{b_1,b_2}$ are statistically indistinguishable for any $(\lceil k/2 \rceil + 2\lceil \log(r(k,n)) \rceil)$-bit numbers M_{b_1,b_2} and M'_{b_1,b_2}. If therefore we let each party in the random group pick a secret random $\lceil 2/3k \rceil$-bit number $M_{b_1,b_2,i}$ (by picking and publishing encryptions of the bits $\overline{M_{b_1,b_2,i,j}}$ individually and letting $\overline{M_{b_1,b_2,i}} = \boxplus_{j=0}^{\lceil 2/3k \rceil - 1} 2^j \cdot \overline{M_{b_1,b_2,i,j}}$) and compute $\overline{M^{1,b_1}_{b_1,b_2}} = \boxplus_{i=1}^{r(k,n)} \overline{M_{b_1,b_2,i}}$, then as at least one party in the random group is honest (except with negligible probability) the value $M^{2,b_2}_{b_1,b_2} = M^{1,b_1}_{b_1,b_2} + M_{b_1,b_2}$ also hides M_{b_1,b_2}. Note that since $M_{b_1,b_2,i}$ is a $\lceil 2/3k \rceil$-bit number $M^{1,b_1}_{b_1,b_2}$ and $M^{2,b_2}_{b_1,b_2}$ can both be represented using less than k bits and therefore we can still encode two of them, e.g. $M^{1,0}_{0,0}$ and $M^{0,1}_{0,0}$, as an element in $\boldsymbol{Z}_{N^2}$ by $2^k M^{1,0}_{0,0} + M^{1,0}_{0,1}$ (see Fig. 2).

If we adopt this way of generating $\mathcal{E}_{N,g}(K, M)$ the equivalents of Lemma 6 and Lemma 7 still hold and consequently the equivalent of Lemma 10 still holds, i.e. from the output of the circuit one can compute $f(x)$ and no other information about x. However we cannot prove the equivalent of Theorem 6 in the way that theorem was proved. The reason for this is solely a technical problem.

Whereas we could prove the protocol in the previous section secure by a clean-cut reduction to [CDN00] by specifying a circuit over $\boldsymbol{Z}_{N^2}$ for $R_{N,g}$ and then applying the composition theorem of [Can00], the above optimised protocol cannot be proven secure that way, as the distribution of the output of the optimised circuit is not independent of the adversary (the adversary can bias the size of the keys K). The model in [Can00] requires that a function has a well-defined distribution independent of the adversary to specify the ideal model. Therefore the function $R_{N,g}$ as defined by the optimised circuit does not fit the framework of [Can00]. However, this technical problem only applies to the sub-protocol for computing $R_{N,g}$. The composed protocol still has a well-defined functionality as $\mathcal{D}_{N,g}(R_{N,g}(x)) = f(x)$ independently of whether the keys are large or not. Therefore the composed protocol can still be proven secure. The only problem is that we cannot do the cut using the composition theorem, we have to prove the composed protocol secure

as a whole, which ultimately involves going into all the technical details of [CDN00]. This has been omitted, both due to lack of space and the fact that the security of the optimised protocol is intuitively obvious given the security of the protocol in the previous section.

If one is satisfied with a sub-exponential security bound one can decrease the size of the random group to any size in $\omega(\log(k))$ and still have a negligible probability that all the parties in the random group are corrupted. If e.g. we let the size be $\log^2(k)$ the communication complexity drops to $O(\log^2(k)\max(n,k)k)$ bits per gate.

References

[ACM89] *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, Seattle, Washington, 15–17 May 1989.

[BBS86] L. Blum, M. Blum, and M. Shub. A simple secure unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.

[BG85] M. Blum and S. Goldwasser. An efficient probabilistic public key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of Crypto '84*, pages 289–302, Berlin, 1985. Springer-Verlag. Lecture Notes in Computer Science Volume 196.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 503–513, Baltimore, Maryland, 14–16 May 1990.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, winter 2000.

[CDN00] Ronald Cramer, Ivan B. Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. Research Series RS-00-14, BRICS, Department of Computer Science, University of Aarhus, June 2000. To appear at EuroCrypt 2001. Updated version available at Cryptology ePrint Archive, record 2000/064, `http://eprint.iacr.org/`.

[DJ] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. To appear in Proc. of Public Key Cryptography 2001. Obtainable from `http://www.daimi.au.dk/~ivan/papers.html`.

[DK00] Ivan B. Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. Research Series RS-00-30, BRICS, Department of Computer Science, University of Aarhus, November 2000. 14 pp.

[Gen00] Rosario Gennaro. An improved pseudo-random generator based on discrete log. In Mihir Bellare, editor, *Advances in Cryptology - Crypto 2000*, pages 469–481, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science Volume 1880.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In ACM [ACM89], pages 25–32.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, 25–27 May 1987.

[GR00] Oded Goldreich and Vered Rosen. On the security of modular exponentiation with application to the construction of pseudorandom generators. Cryptology ePrint Archive, record 2000/064, `http://eprint.iacr.org/`, December 2000.

[GW94] Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 574–583, Montréal, Québec, Canada, 23–25 May 1994.

[HSS93] J. Håstad, A. W. Schrift, and A. Shamir. The discrete logarithm modulo a composite hides $O(n)$ bits. *Journal of Computer and System Science*, 47:376–404, 1993.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In ACM [ACM89], pages 12–24.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In Jacques Stern, editor, *Advances in Cryptology - EuroCrypt '99*, pages 223–238, Berlin, 1999. Springer-Verlag. Lecture Notes in Computer Science Volume 1592.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, 27–29 October 1986. IEEE.

A Proof of Lemma 2

Lemma 2. *If Conjecture 1 holds, then for $s \geq 0$ the random variables X_k^{s+1} and Y_k^{s+1} are computationally indistinguishable.*

Proof: Assume for the sake of contradiction, that there exists a PPT distinguisher D and a $c \in \boldsymbol{N}$ such that for infinitely many k it is the case that $|x_{\mathcal{K}_k} - y_{\mathcal{K}_k}| \geq \frac{1}{k^c}$, where $x_{\mathcal{K}_k} = \Pr[D(X_k^{s+1}) = 1]$ and $y_{\mathcal{K}_k} = \Pr[D(Y_k^{s+1}) = 1]$. Assume for the rest of the proof that k is a value for which this is true.

By [DJ] the random variables $\tilde{X}_k^{s+1} = [(N, g, x) \leftarrow_R X_k^{s+1} : (N, x)]$ and $\tilde{Y}_k^{s+1} = [(N, g, x) \leftarrow_R Y_k^{s+1} : (N, x)]$ are computationally indistinguishable relative to Conjecture 1. We reach a contradiction by constructing a distinguisher $\tilde{D}$ of $\tilde{X}_k^{s+1}$ and $\tilde{Y}_k^{s+1}$ from D with advantage $\frac{1}{8k^{3c+1}}$. The distinguisher is given in Fig. 3. In the analysis of the distinguisher we let $x_{N,g} = \Pr[D(N, g, x) = 1 | x \in_R H_N^{s+1}]$, let $y_{N,g} = \Pr[D(N, g, x) = 1 | x \in_R \boldsymbol{Z}_{N^{s+1}}^*]$, let $b_{N,g} = 0$ if $x_{N,g} > y_{N,g}$ and 1 otherwise, let $\mathcal{E}_{N,g}$ be the event that $\tilde{D}$ gets N as input and $\tilde{D}$ draws g during execution, let $p_{N,g}$ be the probability of $\mathcal{E}_{N,g}$, and let $\Pr^{N,g}[\cdot]$ be the probability function conditioned on $\mathcal{E}_{N,g}$. Then

$$\begin{aligned}
\underline{\mathbf{proc}}\ \tilde{D}(N, z)\ \equiv\ & \\
& g \leftarrow_R \boldsymbol{Z}_n^*; \\
& x \leftarrow_R H_N^{s+1};\ b_x \leftarrow_R D(N, g, x); \\
& y \leftarrow_R \boldsymbol{Z}_{N^{s+1}}^*;\ b_y \leftarrow_R D(N, g, y); \\
& \underline{\mathbf{if}}\ b_x \neq b_y\ \underline{\mathbf{then}}\ \overline{b}_{N,g} \leftarrow b_y \\
& \qquad\qquad \underline{\mathbf{else}}\ \overline{b}_{N,g} \leftarrow_R \{0, 1\}\ \underline{\mathbf{fi}} \\
& b_z \leftarrow_R D(N, g, z); \\
& \underline{\mathbf{return}}\ \overline{b}_{N,g} \oplus b_z
\end{aligned}$$

Fig. 3. Distinguisher for $\tilde{X}_x^{s+1}$ and $\tilde{Y}_x^{s+1}$.

$$\begin{aligned}
\Pr^{N,g}[\tilde{D}(N, H_N^{s+1}) = 1] =& \frac{1}{2} \Pr^{N,g}[b_x = b_y] + \Pr^{N,g}[b_x = 0, b_y = 1, b_z = 0 | x \in_R H_N^{s+1}] \\
& + \Pr^{N,g}[b_x = 1, b_y = 0, b_z = 1 | x \in_R H_N^{s+1}] \\
=& \frac{1}{2} \Pr^{N,g}[b_x = b_y] + (1 - x_{N,g}) y_{N,g} (1 - x_{N,g}) + x_{N,g} (1 - y_{N,g}) x_{N,g}
\end{aligned}$$

and in the same way

$$\Pr^{N,g}[\tilde{D}(N, \boldsymbol{Z}_{N^{s+1}}^*) = 1] = \frac{1}{2} \Pr^{N,g}[b_x = b_y] + (1 - x_{N,g}) y_{N,g} (1 - y_{N,g}) + x_{N,g} (1 - y_{N,g}) y_{N,g}\ .$$

From this it follows by collecting terms that

$$\Pr^{N,g}[\tilde{D}(N, H_N^{s+1}) = 1] - \Pr^{N,g}[\tilde{D}(N, \boldsymbol{Z}_{N^{s+1}}^*) = 1] = (x_{N,g} - y_{N,g})^2\ .$$

Now let $\delta = 2k^c$ and let $\mathcal{K}_k^\delta$ be the set of keys $(N, g) \in \mathcal{K}_k$ where $|x_{N,g} - y_{N,g}| \geq 1/\delta$. Then it is easy to see that $\Pr[(N, g) \in \mathcal{K}_k^\delta | (N, g) \in_R \mathcal{K}_k] > \frac{1}{2k^c}$. Assume namely otherwise, then

$$\begin{aligned}
E_{(N,g) \in \mathcal{K}_k}[|x_{N,g} - y_{N,g}|] =& \Pr[(N, g) \in \mathcal{K}_k^\delta | (N, g) \in_R \mathcal{K}_k] E_{(N,g) \in \mathcal{K}_k^\delta}[|x_{N,g} - y_{N,g}] \\
& + \Pr[(N, g) \in \mathcal{K}_k \setminus \mathcal{K}_k^\delta | (N, g) \in_R \mathcal{K}_k] E_{(N,g) \in \mathcal{K}_k \setminus \mathcal{K}_k^\delta}[|x_{N,g} - y_{N,g}] \\
<& \Pr[(N, g) \in \mathcal{K}_k^\delta | (N, g) \in_R \mathcal{K}_k] + \frac{1}{\delta} \leq \frac{1}{k^c}
\end{aligned}$$

which contradicts that $|x_{\mathcal{K}_k} - y_{\mathcal{K}_k}| \geq \frac{1}{k^c}$. Therefore

$$\begin{aligned}
\Pr[\tilde{D}(\tilde{X}_k^{s+1}) = 1] - \Pr[\tilde{D}(\tilde{Y}_k^{s+1}) = 1] &= \sum_{(N,g)} p_{N,g}(\Pr^{N,g}[\tilde{D}(N, H_N^{s+1}) = 1] - \Pr^{N,g}[\tilde{D}(N, \boldsymbol{Z}^*_{N^{s+1}}) = 1]) \\
&\geq \sum_{(N,g)} p_{N,g}(x_{N,g} - y_{N,g})^2 \geq \sum_{(N,g)\in\mathcal{K}_k^\delta} p_{N,g}\frac{1}{\delta^2} \\
&= \frac{1}{\delta^2}\Pr[(N,g) \in \mathcal{K}_k^\delta | (N,g) \in_R \mathcal{K}_k]\Pr[(N,g) \in \mathcal{K}_k] \\
&= \Pr[(N,g) \in \mathcal{K}_k]\frac{1}{8k^{3c}} \geq \frac{1}{8k^{3c+1}} \ . \ \square
\end{aligned}$$