On the (Im-)Possibility of Extending Coin Toss

Dennis Hofheinz[1], Jörn Müller-Quade[2], and Dominique Unruh[2]

[1] CWI, Cryptology and Information Security Group, Prof. Dr. R. Cramer,
Dennis.Hofheinz@cwi.nl

[2] IAKS, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth, Universität Karlsruhe,
{muellerq,unruh}@ira.uka.de

Abstract. We consider the cryptographic two-party protocol task of extending a given coin toss. The goal is to generate n common random coins from a single use of an ideal functionality which gives $m < n$ common random coins to the parties. In the framework of Universal Composability we show the impossibility of securely extending a coin toss for statistical and perfect security. On the other hand, for computational security the existence of a protocol for coin toss extension depends on the number m of random coins which can be obtained "for free".

For the case of stand-alone security, i.e., a simulation based security definition without an environment, we present a novel protocol for unconditionally secure coin toss extension. The new protocol works for superlogarithmic m, which is optimal as we show the impossibility of statistically secure coin toss extension for smaller m.

Combining our results with already known results, we obtain a (nearly) complete characterization under which circumstances coin toss extension is possible.

Keywords: coin toss, universal composability, reactive simulatability, cryptographic protocols.

1 Introduction

Manuel Blum showed in [5] how to flip a coin over the telephone line. His protocol guaranteed that even if one party does not follow the protocol, the other party still gets a uniformly distributed coin toss result. This general concept of generating common randomness in a way such that no dishonest party can dictate the result proved very useful in cryptography, e.g., in the construction of protocols for general secure multi-party computation.

Here we are interested in the task of *extending* a given coin toss. That is, suppose that two parties already have the possibility of making a single m-bit coin-toss. Is it possible for them to get $n > m$ bits of common randomness? The answer we come up with is basically: "it depends."

The first thing the extensibility of a given coin toss depends on is the required security type. One type of security requirement (which we call "stand-alone simulatabiliy" here) can simply be that the protocol imitates an ideal coin toss functionality in the sense of [13], where a simulator has to invent a realistic protocol run after learning the outcome of the ideal coin-toss. A stronger type of requirement is to demand universal composability, which basically means that the protocol imitates an ideal coin toss functionality even in arbitrary protocol environments. Security in the latter sense can conveniently be captured in a simulatability framework like the Universal Composability framework [6, 8] or the Reactive Simulatability model [16, 3].

Orthogonal to this, one can vary the level of fulfilment of each of these requirements. For example, one can demand stand-alone simulatability of the protocol with respect to polynomial-time adversaries in the sense that real protocol and ideal functionality are only computationally indistinguishable. This specific requirement is already fulfilled by the protocol of Blum. Alternatively, one can demand, e.g., universal composability of the protocol with respect to unbounded adversaries. This would then yield statistical or even perfect security. We show that whether such a protocol exists depends on the asymptotic behaviour of m.

Our results are summarized in the table below. A "yes" or "no" indicates whether a protocol for coin toss extension exists in that setting. "Depends" means that the answer depends on the size of the seed (the m-bit coin toss available by assumption), and **boldface** indicates novel results.

Security type ↓ / level →	Computational	Statistical	Perfect
stand-alone simulatability	yes	**depends**[3]	**no**
universal composability	**depends**[4]	**no**	**no**

Known results in the perfect and statistical case. A folklore theorem states, that (perfectly non-trivial) statistically secure coin-toss is impossible from scratch (even in very lenient security models). By Kitaev, this result was extended even to protocols using quantum communication (cf. [1]). [4] first investigated the problem of extending a coin-toss. They presented a statistically secure protocol for extending a given coin-toss (pre-shared using a VSS), if less than $\frac{1}{6}$ of the parties are corrupted. Note that their main attention was on the efficiency of the protocol, since in that scenario arbitrary multi-party computations and therefore in particular coin-toss from scratch are known to be possible. The result does not apply to the two-party case.

Our results in the perfect and statistical case. Our results in the perfect case are most easily explained. For the perfect case, we show impossibility of *any* coin toss extension, no matter how (in-)efficient. We show this for stand-alone simulatability (Coro. 1) and for universal composability. Now for the statistical case. When demanding only stand-alone simulatability, the situation depends on the number of the already available common coins. Namely, we give an efficient protocol to extend m common coins to any polynomial number (in the security parameter), if m is superlogarithmic (Th. 5). Otherwise, we show that there can

[3] Coin toss extension is possible if and only if the seed has superlogarithmic length.

[4] Coin toss extension is impossible if the seed does not have superlogarithmic length. The possibility result depends on the complexity assumption we use, cf. Section 3.1.

even be no protocol that derives $m+1$ common random coins (Coro. 1). In the universal composability setting, the situation is more clear: we show that there simply is no protocol that derives from m common coins $m+1$ coins, no matter how large m is (Th. 6). (However, here we restrict to protocols that run in a polynomial number of rounds.)

Known results in the computational case. The possibility of coin tossing (in a non-simulation based model) was first shown by [5] and this protocol can be proven secure in a stand-alone security model. For the UC framework coin-toss was proven to be impossible in [9], unless a helping functionality like a CRS is given. In [12], the task of coin-toss is considered in a scenario slightly different from ours: in [12], protocol participants may not abort protocol execution without generating output. In that setting, [12] show that coin-toss is generally not possible even against computationally limited adversaries. However, to the best of our knowledge, an *extension* of a given coin toss has not been considered so far in the computational setting.

Our results in the computational case. We answer the question concerning the minimal size necessary for a coin-toss to be extensible: If an m-bit coin-toss functionality is given, and m is not superlogarithmic, then it is already impossible for the parties to derive $m+1$ common random coins (in a universally composable way) from it (Th. 2). However, we also show that under strengthened computational assumptions, there are protocols that extend m to any polynomial number (in the security parameter) of common random coins, if m is superlogarithmic (Th. 1). In that sense, we give the remaining parts for a complete characterization of the computational case.

Notation

- A function f is *negligible*, if for any $c > 0$, $f(k) \leq k^{-c}$ for sufficiently large k (i.e., $f \in k^{-\omega(1)}$).
- f is *polynomially bounded*, if for some $c > 0$, $f(k) \leq k^c$ for sufficiently large k (i.e., $f \in k^{O(1)}$).
- f is *polynomially-large*, if there is a $c > 0$ s.t. $f(k)^c \geq k$ for sufficiently large k (i.e., $f \in k^{\Omega(1)}$).
- f is *superpolynomial*, if for any $c > 0$, $f(k) > k^c$ for sufficiently large k (i.e., $f \in k^{\omega(1)}$).
- f is *superlogarithmic*, if $f/\log k \to \infty$ (i.e., $f \in \omega(\log k)$). It is easy to see that f is superlogarithmic if and only if 2^{-f} is negligible.
- f is *superpolylogarithmic*, if for any $c > 0$, $f(k) > (\log k)^c$ for sufficiently large k (i.e., $f \in (\log k)^{\omega(1)}$).
- f is *exponentially-small*, if there exists a $c > 1$, s.t. $f(k) \leq c^{-k}$ for sufficiently large k (i.e., $f \in \Omega(1)^{-k} = 2^{-\Omega(k)}$).
- f is *subexponential*, if for any $c > 1$, $f(k) < c^k$ for sufficiently large k (i.e., $f \in o(1)^k = 2^{o(k)}$).

2 Security definitions

In this section we roughly sketch the security definitions used throughout this paper. We distinguish between two notions: stand-alone simulatability as defined in [13],[5] and Universal Composability (UC) as defined in [6].

Stand-alone simulatability. In [13] a definition for the security of two-party secure function evaluations is given (called *security in the malicious model*). We will give a sketch, for more details we refer to [13].

A protocol consists of two parties that alternatingly send messages to each other. The parties may also invoke an ideal functionality, which is given as an oracle (in our cases, they invoke a smaller coin-toss to realise a larger one).

We say the protocol π stand-alone simulatably realises a probabilistic function f, if for any efficient adversary A that may replace none or a single party, there is an efficient simulator S s.t. for all inputs the following random variables are computationally indistinguishable:

- *The real protocol execution.* This consists of the view of the corrupted parties upon inputs x_1 and x_2 for the parties and the auxiliary input z for the adversary, together with the outputs I of the parties.
- *The ideal protocol execution.* Here the simulator first learn the auxiliary input z and possibly the input for the corrupted party (the simulator must corrupt the same party as the adversary). Then he can choose the input of the corrupted party for the probabilistic function f, the other inputs are chosen honestly (i.e., the first input is x_1 if the first party is uncorrupted, and the second input x_2 if the second party is).

  Then the simulator learns the output I of f (we assume the output to be equal for all parties). It may now generate a fake view v of the corrupted parties. The ideal protocol execution then consists of v and I.

Of course, in our case the probabilistic function f (the coin-toss) has no input, so the above definition gets simpler.

What we have sketched above is what we call *computational* stand-alone simulatability. We further define *statistical* stand-alone simulatability and *perfect* stand-alone simulatability. In these cases we do not consider efficient adversaries and simulators, but unlimited ones. In the case of statistical stand-alone simulatability we require the real and ideal protocol execution to be statistically indistinguishable (and not only computationally), and in the perfect case we even require these distributions to be identical.

Universal Composability. In contrast to stand-alone simulatability, Universal Composability [6] is a much stricter security notion. The main difference is the existence of an environment, that may interact with protocol and adversary (or with ideal functionality and simulator)

[5] In fact, [13] does not use the name stand-alone simulatability but simply speaks about *security in the malicous model*. We adopt the name stand-alone simulatability for this paper to be able to better distinguish the different notions.

and try to distinguish between real and ideal protocol. This additional strictness brings the advantage of a versatile composition theorem (the Universal Composition Theorem [6]).

We only sketch the model here and refer to [6] for details.

A protocol consists of several machines that may (a) get input from the environment, (b) give output to the environment (both also during the execution of the protocol), and (c) send messages to each other.

The *real protocol execution* consists of a protocol π, an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$. Here the environment may freely communicate with the adversary, and the latter has full control over the network, i.e., it may deliver, delay or drop messages sent between parties. We assume the authenticated model in this paper, so the adversary learns the content of the messages but may not modify it. When $\mathcal{Z}$ terminates, it gives a single bit of output. The adversary may choose to corrupt parties at any point in time.[6]

The *ideal protocol execution* is defined analogously, but instead of a protocol π there is an *ideal functionality* $\mathcal{F}$ and instead of the adversary there is a simulator $\mathcal{S}$. The simulator can only learn and influence protocol data, if (a) the functionality explicitly allows this, or (b) it corrupts a party (note that the simulator may only corrupt the same parties as the adversary). In the latter case, the simulator can choose inputs into the functionality in the name of that party and gets the outputs appartaining to that party. In the case of uncorrupted parties, the environment is in control of the corresponding in- and output of the ideal functionality.

We say a protocol π universally composably (UC)-implements an ideal functionality $\mathcal{F}$ (or short π is universally composable if $\mathcal{F}$ is clear from the context), if for any efficient adversary $\mathcal{A}$, there is an efficient simulator $\mathcal{S}$, s.t. for all efficient environments $\mathcal{Z}$ and all auxiliary inputs z for $\mathcal{Z}$, the distributions of the output-bit of $\mathcal{Z}$ in the real and the ideal protocol execution are indistinguishable.

What has been sketched above we call *computational* UC. We further define *statistical* and *perfect* UC. In these notions, we allow adversary, simulator and environment to be unlimited machines. Further, in the case of perfect UC, we require the distributions of the output-bit of $\mathcal{Z}$ to be *identical* in real and ideal protocol execution.

The Ideal Functionality for Coin Toss. To describe the task of implementing a universally composable coin-toss, we have to define the ideal functionality of n-bit coin-toss.

In the following, let n denote a positive integer-valued function.

Below is an informal description of our ideal functionality for a n-bit coin toss. First, the functionality waits for initialization inputs from both parties P_1 and P_2. As soon as both parties have this way signalled their willingness to start, the functionality selects n coins in form of an n-bit string κ uniformly and sends

[6] It is then called an *adaptive* adversary. If the adversary can only corrupt parties before the start of the protocol, we speak of *static corruption*. All results in this paper hold for both variants of the security definition.

this κ to the adversary. (Note that a coin toss does not guarantee secrecy of any kind.)

If the functionality now sent κ directly and without delay to the parties, this behaviour would not be implementable by any protocol (this would basically mean that the protocol output is immediately available, even without interaction). So the functionality lets the adversary decide when to deliver κ to each party. Note however, that the adversary may not in any way influence the κ that is delivered.

A more detailed description follows:

Ideal functionality CT_n (n-bit Coin Toss)

1. Wait until there have been "`init`" inputs from P_1 and P_2. Ignore messages from the adversary, but immediately inform the adversary about the `init`.
2. Select $\kappa \in \{0,1\}^n$ uniformly and send κ to the adversary. From now on:
   – on the first (and only the first) "`deliver to 1`" message from the adversary, send κ to P_1,
   – on the first (and only the first) "`deliver to 2`" message from the adversary, send κ to P_2.

Using CT_n, we can also formally express what we mean by *extending* a coin toss. Namely:

Definition 1. *Let $n = n(k)$ and $m = m(k)$ be positive, polynomially bounded and computable functions such that $m(k) < n(k)$ for all k. Then a protocol is a* universally composable $(m \to n)$-coin toss extension protocol *if it securely and non-trivially implements CT_n by having access only to CT_m. This security can be computational, statistical or perfect.*

By a "non-trivial" implementation we mean a protocol that, with overwhelming probability, guarantees outputs if no party is corrupted and all messages are delivered. (Alternatively, one may also consider protocols that provide output with overwhelming probability.) This requirement is useful since without it, a trivial protocol that does not generate any output formally implements every functionality. (Cf. [10] and [2, Section 5.1] for more discussion and formal definitions of "non-triviality.")

On unlimited simulators. Following [3], we have modelled statistical and perfect stand-alone and UC security using unlimited simulators. Another approach is to require the simulators to be polynomial in the running-time of the adversary. All our results apply also to that case: For the impossibility results, this is straightforward, since the security notion gets stricter when the simulators become more restricted. The only possibility result for statistical/perfect security is given in Theorem 5. There, the simulator we construct is in fact polynomial in the runtime of the adversary.

In the following sections, we investigate the existence of such coin toss extension protocols, depending on the desired security level (i.e., computational / statistical / perfect security) and the parameters n and m.

3 The Computational Case

3.1 Universal Composability

In the following, we need the assumption of enhanced trapdoor permutations with dense public descriptions (called ETD henceforth). Roughly, these are trapdoor permutations with the additional properties that (i) one can choose the public key in an oblivious fashion, i.e., even given the coin tosses we used it is infeasible to invert the function, and (ii) the public keys are computationally indistinguishable from random strings. We also need the notion of exponentially-hard ETD, which are secure even against subexponential-time adversaries. For detailed definitions, cf. the full version [14].

Lemma 1. *There is a constant $d \in \mathbb{N}$ s.t. the following holds:*

Assume that ETD exist, s.t. the size of the circuits describing the ETD is bounded by $s(k)$ for security parameter k.[7]

Then there is a protocol π using a uniform common reference string (CRS) of length $s(k)^d$, s.t. π securely UC-realises a bit commitment that can be used polynomially many times.

A protocol for realising bit commitment using a CRS has been given in [10]. To show this lemma, we only need to review their construction to see, that a CRS of length s^d is indeed sufficient. For details, see the full version [14].

Lemma 2. *Let $s(k)$ be a polynomially bounded function, that is computable in time polynomial in k.*

Assume one of the following holds:

- *ETD exist and s is a polynomially-large function.*
- *Exponentially-hard ETD exist and s is a superlogarithmic function.*

Then there also exist a constant $e \in \mathbb{N}$ independent of s and ETD, s.t. the size of the circuits describing the ETD is bounded by $s(k)^e$ for security parameter k.

This is shown by scaling the security parameter of the original ETD. The proof is given in the full version [14].

Theorem 1. *Let $n = n(k)$ and $m = m(k)$ be polynomially bounded and efficiently computable functions. Assume one of the following conditions holds:*

- *m is polynomially-large and ETD exist, or*
- *m is super*poly*logarithmic and exponentially-hard ETD exist.*

Then there is a polynomial-time computationally universally composable protocol π for $(m \to n)$-coin toss extension.

[7] By the size of the circuits we means the total size of the circuits describing both the key generation and the domain sampling algorithm. Note that then trivially also the size of the resulting keys and the amount of randomness used by the domain sampling algorithm are bounded by $s(k)$.

Proof. Let d be as in Lemma 1. Let further e be as in Lemma 2. If m is polynomially-large or superpolylogarithmic, then $s := m^{1/(de)}$ is polynomially-large or superlogarithmic, resp. So, by Lemma 2 there are ETD, s.t. the size of the circuits describing the ETD is bounded by $s^e = m^{1/e}$. Then, by Lemma 1 there is a UC-secure protocol for implementing n bit commitments using an $(m^{1/d})^d = m$-bit CRS.

It is straightforward to see that using n UC-bit-commitments one can UC-securely implement an n-bit coin-toss using the protocol from [5]. Furthermore, an m-bit CRS can be trivially implemented using an m-bit coin-toss. Using the Composition Theorem we can put the above constructions together and get a protocol that UC-realises an n-bit coin-toss using an m-bit coin-toss. □

Note that given stronger, but possibly unrealistic assumptions, the lower bound for m in Theorem 1 can be decreased. If we assume that for any superlogarithmic m, there are ETD s.t. the size of their circuits is bounded by $m^{1/d}$ (where d is the constant from Lemma 1), we get coin-toss extension even for superlogarithmic m (using the same proof as for Theorem 1, except that instead of Lemma 2 we use the stronger assumption).

However, we cannot expect an even better lower bound for m, as the following theorem shows:

Theorem 2. *Let $n = n(k)$ and $m = m(k)$ be functions with $n(k) > m(k) \geq 0$ for all k, and assume that m is not superlogarithmic (i.e., 2^{-m} is non-negligible). Then there is no non-trivial polynomial-time computationally universally composable protocol for $(m \to n)$-coin toss extension.*

Proof (sketch). Assume for contradiction that protocol π, with parties P_1 and P_2 using CT_m, implements CT_n (with m, n as in the theorem statement). Let $\mathcal{A}_1$ be an adversary on π that, taking the role of a corrupted party P_1, simply reroutes all communication of P_1 (with either P_2 or CT_m) to the protocol environment $\mathcal{Z}_1$ and thus lets $\mathcal{Z}_1$ take part as P_1 in the real protocol.

Imagine a protocol environment $\mathcal{Z}_1$, running with π and $\mathcal{A}_1$ as above, that keeps and internal simulation $\overline{P_1}$ of P_1 and lets this simulation take part in the protocol (through $\mathcal{A}_1$). After a protocol run, $\mathcal{Z}_1$ inspects the output $\overline{\kappa_1}$ of $\overline{P_1}$ and compares it to the output κ_2 of the uncorrupted P_2.

In a real protocol run with π, $\mathcal{A}_1$, and $\mathcal{Z}_1$, we will have $\overline{\kappa_1} = \kappa_2$ with overwhelming probability since π non-trivially implements CT_n, and CT_n guarantees common outputs. So a simulator $\mathcal{S}_1$, running in the ideal model with CT_n and $\mathcal{Z}_1$, must be able to achieve that the ideal output κ_2 (that is ideally chosen by CT_n and cannot be influenced by $\mathcal{S}_1$) is identical to what the simulation $\overline{P_1}$ of P_1 inside $\mathcal{Z}_1$ outputs. In that sense, $\mathcal{S}_1$ must be able to "convince" $\overline{P_1}$ to also output κ_2. To this end, $\mathcal{S}_1$ may—and must—fake a complete real protocol communication as $\mathcal{A}_1$ would deliver it to $\mathcal{Z}_1$ (and thus, to $\overline{P_1}$).

However, then we can construct another protocol environment $\mathcal{Z}_2$ that expects to take the role of party P_2 in a real protocol run (just like $\mathcal{Z}_1$ expected to take the role of P_1). To this end, an adversary $\mathcal{A}_2$ on π with corrupted P_2 is employed that forwards all communication of P_2 with either P_1 or CT_n to $\mathcal{Z}_2$.

Internally, $\mathcal{Z}_2$ now simulates $\mathcal{S}_1$ (and not P_2!) from above and an instance $\overline{\mathsf{CT}_n}$ of the trusted host $\mathsf{CT_n}$. Recall that $\mathcal{S}_1$, given a target string κ by $\mathsf{CT_n}$, mimics an uncorrupted P_2 along with an instance of CT_m. In that situation, $\mathcal{S}_1$ can convince an honest P_1 with overwhelming probability to eventually output κ.

Chances are 2^{-m} that the CT_m-instance made up by $\mathcal{S}_1$ outputs the same seed as the real CT_m in a run of $\mathcal{Z}_2$ with π and $\mathcal{A}_2$. So with probability at least $2^{-m} - \mu$ for negligible μ, in such a run, $\mathcal{Z}_2$ observes a P_1-output κ that is identical to the output of the internally simulated $\overline{\mathsf{CT}_n}$. But then, by assumption about the security of π, there is also a simulator $\mathcal{S}_2$ for $\mathcal{A}_2$ and $\mathcal{Z}_2$ that provides $\mathcal{Z}_2$ with an indistinguishable view. In particular, in an ideal run with $\mathcal{S}_2$ and CT_n, $\mathcal{Z}_2$ observes equal outputs from CT_n and $\overline{\mathsf{CT}_n}$ with probability at least $2^{-m} - \mu'$ for negligible μ'. This is a contradiction, as both outputs are uniformly and independently chosen n-bit strings, and $n \geq m + 1$. □

4 Statistical and Perfect Cases

4.1 Stand-alone simulatability

We start off with a negative result:

Theorem 3. *Let $m < n$ be functions in the security parameter k. If m is not superlogarithmic, there is no two-party n-bit coin-toss protocol π (not even an inefficient one) that uses an m-bit coin-toss and has the following properties:*
- Non-triviality. *If no party is corrupted, the probability that the parties give different, invalid or no output is negligible (by invalid output we mean output not in $\{0,1\}^n$).*
- Security. *For any (possibly unbounded) adversary corrupting one of the parties there is a negligible function μ, s.t. for every security parameter k and every $c \in \{0,1\}^n$, the probability for protocol output c is at most $2^{-n} + \mu(k)$.*

If we require perfect non-triviality (the probability for different or no outputs is 0) and perfect security (the probability for a given output c is at most 2^{-n}), such a protocol π does not exist, even if m is superlogarithmic.

Proof (sketch). It is sufficient to consider the case $n = m + 1$.

Without loss of generality, we can assume that the available m-bit coin toss is only used at the end of the protocol. Similarly, we can assume that in the honest case, the parties never output distinct values. A detailed proof for these statements can be found in the full proof.

To show the theorem, we first consider "complete transcripts" of the protocol. By a complete transcript we mean all messages sent during the run of a protocol, excluding the value of the m-bit coin-toss. We distinguish three sets of complete transcripts: the set $\mathfrak{A}$ of transcripts having non-zero probability for the protocol output 0^n, the set $\mathfrak{B}$ of transcripts having zero probability of output 0^n and zero probability that the protocol gives no output, and the set $\mathfrak{C}$ of transcripts having non-zero probability of giving no output. Note that, since for a complete transcript, the protocol output only depends on the m-bit coin-toss, any of the above non-zero probabilities is at least 2^{-m}.

For any partial transcript p (i.e., a situation *during* the run of the protocol), we define three values α, β, γ. The value α denotes the probability with which a corrupted Alice can enforce a transcript in $\mathfrak{A}$ starting from p, the value β denotes the probability with which a corrupted Bob can enforce a transcript in $\mathfrak{B}$, and the value γ denotes the probability that the complete protocol transcript will lie in $\mathfrak{C}$ if no-one is corrupted. We show inductively that for any partial transcript p, $(1-\alpha)(1-\beta) \le \gamma$. In particular, this holds for the beginning of the protocol. For simplicity, we assume that 2^{-m} is not only non-negligible, but noticeable (in the full proof, the general case is considered). Since a transcript in $\mathfrak{C}$ gives no output with probability at least 2^{-m}, the probability that the protocol generates no output (in the uncorrupted case) is at least $2^{-m}\gamma$. By the non-triviality condition, this probability is negligible, so γ must be negligible, too. So $(1-\alpha)(1-\beta)$ is negligible, too. Therefore $\max\{1-\alpha, 1-\beta\}$ must be negligible. For now, we assume that $1-\alpha$ is negligible or $1-\beta$ is negligible (for the general case, see the full proof).

If $1-\alpha$ is negligible, the probability for output $\mathtt{0}^n$ is at least $2^{-m}\alpha$. Since α is overwhelming and 2^{-m} noticeable, this is greater than $2^{-n} = \frac{1}{2}2^{-m}$ by a noticeable amount which contradicts the security property.

If $1-\beta$ is negligible, we consider the maximum probability a corrupted Bob can achieve that the protocol output is not $\mathtt{0}^n$. By the security property, this probability should be at most $(2^n-1)2^{-n}$ plus a negligible amount, which is not overwhelming. However, since every transcript in $\mathfrak{B}$ gives such an output with probability 1, the probability of such is β, which is overwhelming, in contradiction of the security property.

The perfect case is proven similarly. □

The full proof is given in the full version [14].

Corollary 1. *By a* non-trivial *coin-toss protocol we mean a protocol s.t. (in the uncorrupted case) the probability that the parties give no or different output is negligible. By a* perfectly non-trivial *coin-toss protocol where this probability is zero.*

Let m be not superlogarithmic *and $n > m$. Then there is no non-trivial protocol realising n-bit coin-toss using an m-bit coin-toss in the sense of* statistical *stand-alone simulatability.*

Let m be any *function (possibly superlogarithmic) and $n > m$. Then there is no perfectly non-trivial protocol realising n-bit coin-toss using an m-bit coin-toss in the sense of* perfect *stand-alone simulatability.*

Proof. A statistically secure protocol would have the security property from Theorem 3 and thus, if non-trivial, contradict Theorem 3. Analogously for perfect security. □

However, not all is lost:

Now we will prove that there exists a protocol for coin toss extension from m to n bit which is statistically stand-alone simulatably secure. The basic idea is to have the parties P_1 and P_2 contribute random strings to generate one string with sufficiently large min-entropy (the min-entropy of a random variable X is defined as $\min_x - \log \mathsf{Pr}[X = x]$). The randomness from this string is then extracted using a randomness extractor. Interestingly the amount of perfect randomness (i.e., the size of the m-bit coin-toss) one needs to invest is smaller than the amount extracted. This makes coin toss extension possible.

To obtain the coin toss extension we need a result about randomness extractors able to extract one bit of randomness while leaving the seed reusable like a catalyst.

Lemma 3. *For every m there exists a function $h_m : \{0,1\}^m \times \{0,1\}^{m-1} \to \{0,1\}, (s,x) \mapsto r$ such that for a uniformly distributed s and for an x with a min-entropy of at least t the statistical distance of $s \| h_m(s,x)$ and the uniform distribution on $\{0,1\}^{m+1}$ is at most $2^{-t/2}/\sqrt{2}$.*

Proof. Let $h_m(s,x) := \langle s_1 \ldots s_{m-1}, x\rangle \oplus s_m$. Here $\langle \cdot, \cdot \rangle$ denotes the inner product and $\oplus$ the addition over GF(2). It is easy to verify that $h_m(s,\cdot)$ constitutes a family of universal hash functions [11], where s is the index selecting from that family. Therefore the Leftover Hash Lemma [15, 17] guarantees that the statistical distance between $s \| h_m(s,x)$ and the uniform distribution on $\{0,1\}^{m+1}$ is bounded by $\frac{1}{2}\sqrt{2 \cdot 2^{-t}} = 2^{-t/2}/\sqrt{2}$. □

With this function h_m a simple protocol is possible which extends $m(k)$ coin tosses to $m(k) + 1$ if the function $m(k)$ is superlogarithmic.

Theorem 4. *Let $m(k)$ be a superlogarithmic function, then there exists a constant round statistically stand-alone simulatable protocol that realises an $(m+1)$-bit coin-toss using an m-bit coin-toss.*

Proof. Let h_m be as in Lemma 3. Then the following protocol realises a coin toss extension by one bit. Assume $m := m(k)$ where k is the security parameter.

1. P_1 uniformly chooses $a \in \{0,1\}^{\lfloor \frac{m-1}{2} \rfloor}$ and sends a to P_2
2. P_2 uniformly chooses $b \in \{0,1\}^{\lceil \frac{m-1}{2} \rceil}$ and sends b to P_1
3. If one party fails to send a string of appropriate length or aborts then this string is assumed by the other party to be an all-zero string of the appropriate length
4. P_1 and P_2 invoke the m-bit coin toss functionality and obtain a uniformly distributed $s \in \{0,1\}^m$. If one party P_i fails to invoke the coin toss functionality or aborts, then the other party chooses s at random
5. Both P_1 and P_2 compute $s \| h_m(s, a\|b)$ and output this string.

Similar to construction 7.4.7 in [13] the protocol is constructed in a way that the adversary is not able to abort the protocol (not even by not terminating). Hence we can safely assume that the adversary will send some message of the correct length and will invoke the coin toss functionality. We assume the adversary to corrupt P_2, corruption of P_1 is handled analogously. Further we assume the random tape of $\mathcal{A}$ to be fixed in the following. Due to these assumptions there exists a function $f_{\mathcal{A}} : \{0,1\}^{\lfloor m/2 \rfloor} \to \{0,1\}^{\lceil m/2 \rceil}$ for each real adversary $\mathcal{A}$ such that the message b sent in step 2 of the protocol equals $f_{\mathcal{A}}(a)$. There is no loss in generality if we assume the view of the parties to consists of just a, b, s and the protocol output to be $s \| h_m(s, a \| b)$.

Now for a specific adversary $\mathcal{A}$ with fixed random tape the output distribution of the real protocol (i.e., view and output) is completely described by the following experiment: choose $a \overset{R}{\in} \{0,1\}^{\lfloor m/2 \rfloor}$, let $b \leftarrow f_{\mathcal{A}}(a)$, choose $s \overset{R}{\in} \{0,1\}^{m(k)}$, let $r \leftarrow s \| h_m(s, a \| b)$ and return $((a, b, s), r)$.

We now describe the simulator. To distinguish the the random variables in the ideal model from their real counterparts, we decorate them with a $\sim$, e.g., $\tilde{a}, \tilde{b}, \tilde{s}$. The simulator in the ideal model obtains a string $\tilde{r} \overset{R}{\in} \{0,1\}^{m+1}$ from the ideal n-bit coin-toss functionality and sets $\tilde{s} = r_1 \dots r_m$. Then the simulator chooses $\tilde{a} \overset{R}{\in} \{0,1\}^{\lfloor \frac{m-1}{2} \rfloor}$ and computes $\tilde{b} = f_{\mathcal{A}}(\tilde{a})$ by giving $\tilde{a}$ to a simulated copy of the real adversary. If $h_m(\tilde{s}, \tilde{a} \| \tilde{b}) = \tilde{r}_{m+1}$ then the simulator gives $\tilde{s}$ to the simulated real adversary expecting the coin toss. Then the simulator outputs the view $(\tilde{a}, \tilde{b}, \tilde{s})$. If however, $h_m(\tilde{s}, \tilde{a} \| \tilde{b}) \neq \tilde{r}_{m+1}$ then the simulator *rewinds* the adversary, i.e., the simulator chooses a fresh $\tilde{a} \overset{R}{\in} \{0,1\}^{\lfloor \frac{m-1}{2} \rfloor}$ and again computes $\tilde{b} = f_{\mathcal{A}}(a)$. If now $h_m(\tilde{s}, \tilde{a} \| \tilde{b}) = \tilde{r}_{m+1}$ the simulator outputs $(\tilde{a}, \tilde{b}, \tilde{s})$. If again $h_m(\tilde{s}, \tilde{a} \| \tilde{b}) \neq \tilde{r}_{m+1}$ then the simulator rewinds the adversary again. If after k invocations of the adversary no triple $(\tilde{a}, \tilde{b}, \tilde{s})$ was output, the simulator aborts and outputs *fail*.

To show that the simulator is correct, we have to show that the following to distributions are statistically indistinguishable: $((a, b, s), r)$ as defined in the real model, and $((\tilde{a}, \tilde{b}, \tilde{s}), \tilde{r})$.

By construction of the simulator, it is obvious that the two distributions are identical under the condition that $r_m = 0$, $\tilde{r}_m = 0$ and that the simulator does not fail. The same holds given $r_m = 1$, $\tilde{r}_m = 1$ and that the simulator does not fail. Therefore it is sufficient to show two things: (i) the statistical distance between r and the uniform distribution on n bits is negligible, and (ii) the probability that that the simulator fails is negligible. Property (i) is shown using the properties of the randomness extractor h_m. Since a is chosen at random, the min-entropy of a is at least $\lfloor \frac{m-1}{2} \rfloor \geq \frac{m}{2} - 1$, so the min-entropy of $a \| b$ is also at least $\frac{m}{2} - 1$. Since s is uniformly distributed, it follows by Lemma 3 that the statistical distance between $r = s \| h_m(s, a \| b)$ is bounded by $2^{-m/4-1/2}/\sqrt{2} = (2^{-m})^{1/4}/2$. Since for superlogarithmic m it is 2^{-m} negligible, this statistical distance is negligible.

Property (ii) is then easily shown: From (i) we see, that after each invocation of the adversary the distribution of $h_m(\tilde{s}, \tilde{a} \| \tilde{b})$ is negligibly far from uniform. So the probability that $h_m(\tilde{s}, \tilde{a} \| \tilde{b}) \neq \tilde{r}_m$ is at most negligibly higher than $\frac{1}{2}$. Since the $h_m(\tilde{s}, \tilde{a} \| \tilde{b})$ in the different invokations of the adversary are independent, the probability that $h_m(\tilde{s}, \tilde{a} \| \tilde{b}) \neq \tilde{r}_m$ after each activation is neglibigly far from 2^{-k}. So the simulator fails only with negligible probability.

It follows that the real and the ideal protocol execution are indistinguishable, and the protocol stand-alone simulatably implements an $(m+1)$-bit coin-toss. □

The idea of the one bit extension protocol can be extended by using an extractor which extracts a larger amount of randomness (while not necessarily treating the seed like a catalyst). This yields constant round coin toss extension protocols. However, the simulator needed for such a protocol does not seem to be efficient, even if the real adversary is. To get a protocol that also fulfils both the property of computational stand-alone simulatabiliy and of statistical stand-alone simulatabiliy, we need a simulator that is efficient if the adversary is.

Below we give such a coin toss extension protocol for superlogarithmic $m(k)$ which is statistically secure *and* computationaly secure, i.e., the simulator for polynomial adversaries is polynomially bounded, too. The basic idea here is to extract one bit at a time in polynomially many rounds.

Theorem 5. *Let $m(k)$ be superlogarithmic, and $p(k)$ be a positive polynomially-bounded function, then there exists a statistically and computationally stand-alone simulatable protocol that realises an $(m+p)$-bit coin-toss using an m-bit coin-toss.*

Proof. Let h_m be as in Lemma 3. Then the following protocol realises a coin toss extension by $p(k)$ bits.

1. `for` $i = 1$ `to` $p(k)$ `do`
   (a) P_1 uniformly chooses $a_i \in \{0,1\}^{\lfloor \frac{m-1}{2} \rfloor}$ and sends a_i to P_2
   (b) P_2 uniformly chooses $b_i \in \{0,1\}^{\lceil \frac{m-1}{2} \rceil}$ and sends b_i to P_1
   (c) If one party fails to send a string of appropriate length or aborts then this string is assumed by the other party to be an all-zero string of the appropriate length
2. P_1 and P_2 invoke the m-bit coin toss functionality and obtain a uniformly distributed $s \in \{0,1\}^m$. If one party P_i fails to invoke the coin toss functionality or aborts, then the other party chooses s at random
3. P_1 and P_2 compute $s \| h_m(s, a_1 \| b_1) \| \dots \| h_m(s, a_{p(k)} \| b_{p(k)})$ and output this string.

We only roughly sketch the differences to the proof of Theorem 4. For each protocol round the simulator follows the strategy described in the proof of Theorem 4 (i.e., the simulator rewinds the adversary by *one* round, if the coin-toss produced is not the correct one.) Then using standard hybrid techniques it can be shown that this simulator indeed gives an indistinguishable ideal protocol run. Here it is only noteworthy that we use the fact that $s \| h_m(s, a_1 \| b_1) \| \dots \| h_m(s, a_{p(k)} \| b_{p(k)})$ is statistically indistinguishable from the uniform distribution on $m + p$ bits. However, this follows directly from Lemma 3 and the fact that each $a_i \| b_i$ has min-entropy at least $\lfloor \frac{m-1}{2} \rfloor$ even given the values of all $a_\mu \| b_\mu$ for $\mu < i$. □

4.2 Universal Composability (statistical/perfect case)

In the case of statistical security, adversary and protocol environment are allowed to be computationally unbounded. In that case, we show that there is no simulatably secure coin toss extension protocol that runs in a polynomial number of rounds. This is forced by requiring the parties to halt after a polynomial number of activations. However, note that we do not impose any restrictions on the amount of computational work these parties perform in one of those activations.

The proof of this statement is done by contradiction. Furthermore, the proof is split up into an auxiliary lemma and the actual proof. In the auxiliary lemma, we show that without loss of generality, a protocol for statistically universally composable coin toss extension has a certain outer form. Then we show that any such protocol (of this particular outer form) is insecure.

For the following statements, we always assume that $m = m(k), n = n(k)$ are arbitrary functions, only satisfying $0 \leq m(k) < n(k)$ for all k. We also restrict to protocols that proceed in a polynomial number of rounds. That is, by a "protocol" we mean in the following one in which each party halts after at most $p(k)$ activations, where $p(k)$ is a polynomial which depends only on the protocol. (As stated above, the parties are still unbounded in each activation.) We start with a helping lemma whose proof is available in the full version [14].

Lemma 4. *If there is a statistically universally composable protocol for* $(m \to n)$*-coin toss extension, then there is also one in which each party*

- *has only one connection to the other party and one connection to* CT_m*,*
- *in each activation sends either an "*`init`*" message to* CT_m *or some message to the other party,*
- *sends in each protocol run at most one message to* CT_m*, and this is always an "*`init`*" message,*
- *the internal state of each of the two parties consists only of the view that this party has experienced so far, and*
- *after* P_i *sends "*`init`*" to* CT_m*, it does not further communicate with* P_{3-i} *(for* $i = 1, 2$ *and in case of no corruptions).*

We proceed with

Lemma 5. *There is no statistically universally composable protocol for* $(m \to n)$*-coin toss extension which meets the requirements from Lemma 4.*

Proof. Assume for contradiction that π, using CT_m, is a statistically universally composable implementation of CT_n, and also satisfies the requirements from Lemma 4.

Assume a fixed environment $\mathcal{Z}_0$ that gives both parties "`init`" input and then waits for both parties to output a coin toss result. Consider an adversary $\mathcal{A}_0$ that delivers all messages between the parties immediately. The resulting setting D_0 is depicted in Figure 1.

Denote the protocol communication in a run of D_0, i.e., the ordered list of messages sent between P_1 and P_2, by *com*. Denote by κ_1 and κ_2 the final outputs of the parties. For $M \subseteq \{0,1\}^n$ and a possible protocol communication prefix $\overline{c}$, let $\mathsf{E}(M, \overline{c})$ be the probability that the protocol outputs are identical and in M, provided that the protocol communication starts with $\overline{c}$, i.e.,

$$\mathsf{E}(M, \overline{c}) := \mathsf{Pr}[\kappa_1 = \kappa_2 \in M \mid \overline{c} \leq \mathit{com}]\,,$$

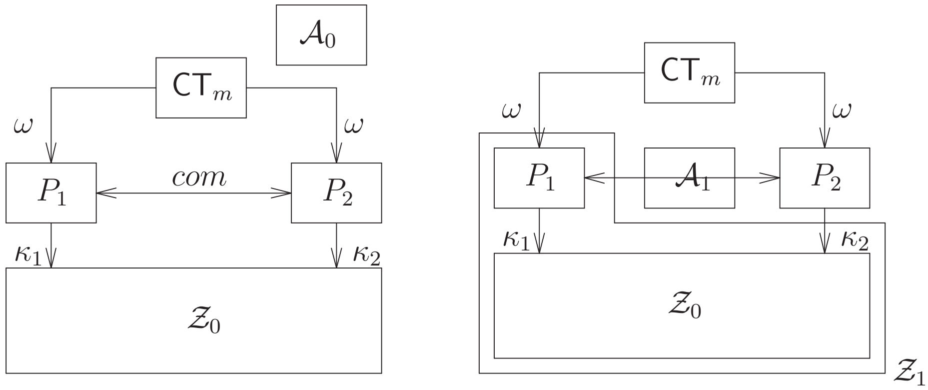

Fig. 1. *Left:* The initial setting D_0 for the statistical case. (Some connections which are not important for our proof have been omitted.) *Right:* Setting D_1 with a corrupted P_1. Setting D_2 (with P_2 corrupted instead of P_1) is defined analogously.

where $x \leq y$ means that x is a prefix of y.

Note that the parties have, apart from their communication *com*, only the seed $\omega \in \{0,1\}^m$ provided by CT_m for computing their final output κ. So we may assume that there is a deterministic function f for which $\kappa_1 = \kappa_2 = f(com, \omega)$ with overwhelming probability.

For a fixed protocol communication $com = c$, consider the set

$$M_c \ := \ \{0,1\}^n \ \setminus \ \{ \ f(c,s) \ \mid \ s \in \{0,1\}^m \ \}$$

of "improbable outputs" after communication c. Then obviously $|M_c| \geq 2^n - 2^m \geq 2^{n-1}$. By definition of the ideal output (i.e., the output of CT_n in the ideal model), this implies that for sufficiently large security parameters k, the probability that $\kappa_1 = \kappa_2 \in M_c$ is at least $2/5$. (Here, any number strictly between 0 and $1/2$ would have done as well.) Otherwise, an environment could distinguish real and ideal model by testing for $\kappa_1 = \kappa_2 \in M_c$. Since $\mathsf{E}(M_c, \varepsilon)$ is exactly that probability, we have $\mathsf{E}(M_c, \varepsilon) \geq 2/5$ for sufficiently large k. Also, $\mathsf{E}(M_c, c)$ is negligible by definition, so M_c satisfies

$$\mathsf{E}(M_c, \varepsilon) - \mathsf{E}(M_c, c) \geq \frac{1}{3} \tag{1}$$

for sufficiently large k.

Since the protocol consists by assumption only of polynomially many rounds, c is a list of size at most $p(k)$ for a fixed polynomial p. This means that there is a prefix $\overline{c}$ of c and a single message m (either sent from P_1 to P_2 or vice versa) such that $\overline{c}m \leq c$ and

$$\mathsf{E}(M_c, \overline{c}) - \mathsf{E}(M_c, \overline{c}m) \geq \frac{1}{3p(k)} \tag{2}$$

for sufficiently large k. Intuitively, this means that at a certain point during the protocol run, a single message m had a significant impact on the probability that the protocol output is in M_c.

Note that such an m must be either sent by P_1 or P_2. So there is a $j \in \{1, 2\}$, such that for infinitely many k, party P_j sends such an m with probability at least $1/2$. We describe a modification D_j of setting D_0. In setting D_j, party P_j is corrupted and simulated (honestly) inside $\mathcal{Z}_j$. Furthermore, adversary $\mathcal{A}_j$ simply relays all communication between this simulation inside $\mathcal{Z}_j$ and the uncorrupted party P_{3-j}. For supplying inputs to the simulation of P_j and to the uncorrupted P_{3-j}, a simulation of $\mathcal{Z}_0$ is employed inside $\mathcal{Z}_j$. The situation (for $j = 1$) is depicted in Figure 1.

Since D_j is basically only a re-grouping of D_0, the random variables com, ω, and κ_i are distributed exactly as in D_0, so we simply identify them. In particular, in D_j, for infinitely many k, there is with probability at least $1/2$ a prefix $\overline{c}$ and a message m *sent by* P_j of com that satisfy (2).

Now we slightly change the environment $\mathcal{Z}_j$ into an environment $\mathcal{Z}'_j$. Each time the simulated P_j sends a message m to P_{3-j}, $\mathcal{Z}'_j$ checks for *all* subsets M of $\{0, 1\}^n$ whether

$$\exists M \subseteq \{0,1\}^n : \quad \mathsf{E}(M, \overline{c}) - \mathsf{E}(M, \overline{c}m) \geq \frac{1}{3p(k)}, \tag{3}$$

where $\overline{c}$ denotes the communication between P_j and P_{3-j} so far.

If (3) holds at some point for the first time, then $\mathcal{Z}'_j$ tosses a coin b uniformly at random, and proceeds as follows: if $b = 0$, then $\mathcal{Z}'_j$ keeps going just as $\mathcal{Z}_j$ would have. In particular, $\mathcal{Z}'_j$ then lets P_j send m to P_{3-j}. However, if $b = 1$, then $\mathcal{Z}'_j$ rewinds the simulation of P_j to the point *before* that activation, and activates P_j again with fresh randomness, thereby letting P_j send a possibly different message m'. In the further proof, $\overline{c}$, m, and M refer to these values for which (3) holds.

In any case, after having tossed the coin b once, $\mathcal{Z}'_j$ remembers the set M from (3), and does not check (3) again. After the protocol finishes, $\mathcal{Z}_j$ outputs either $(\perp, \perp)$ (if (3) was never fulfilled), or (b, β) for the evaluation β of the predicate $[\kappa_1 = \kappa_2 \in M]$ (i.e., $\beta = 1$ iff the protocol gives output, the protocol outputs match and lie in M).

Now by our choice of j, $\mathsf{Pr}[b \neq \perp] \geq 1/2$ for infinitely many k.

Also, Lemma 4 guarantees that the internal state of the parties at the time of tossing b consists only of $\overline{c}$. So, when $\mathcal{Z}'_j$ has chosen $b = 1$, and rewound the simulated P_j, the probability that at the end of the protocol $\kappa_1 = \kappa_2 \in M$ is the same as the probability of that event in the setting D_j under the condition that the communication com begins with $\bar{c}$. This probability again is exactly $\mathsf{E}(M, \bar{c})$ by definition.

Similarly, when $\mathcal{Z}'_j$ has chosen $b = 0$, the probability that at the end of the protocol $\kappa_1 = \kappa_2 \in M$ is the same as the probability of that event in the setting D_j under the condition that the communication com begins with $\bar{c}m$, i.e. $\mathsf{E}(M, \bar{c}m)$.

Therefore just before $\mathcal{Z}'_j$ chooses b (i.e., when $\bar{c}$ and M are already determined), the probability that at the end we will have $\beta = 1 \wedge b = 1$ is $\frac{1}{2}\mathsf{E}(M, \bar{c})$ and the probability of $\beta = 1 \wedge b = 0$ is $\frac{1}{2}\mathsf{E}(M, \bar{c}m)$. Therefore the difference between these probabilities is at least $\frac{1}{2}\big(\mathsf{E}(M, \bar{c}) - \mathsf{E}(M, \bar{c}m)\big) \geq \frac{1}{3p(k)}$.

Since this bound on the difference of the probabilities always holds when $b \neq \perp$, by averaging we get

$$\mathsf{Pr}[\beta = 1 \wedge b = 1 \mid b \neq \perp] - \mathsf{Pr}[\beta = 1 \wedge b = 0 \mid b \neq \perp] \geq \frac{1}{3p(k)}$$

and using the fact that $\mathsf{Pr}[b \neq \perp] \geq \frac{1}{2}$ for infinitely many k we then have that

$$\mathsf{Pr}[\beta = 1 \wedge b = 1] - \mathsf{Pr}[\beta = 1 \wedge b = 0] \geq \frac{1}{6p(k)} \tag{4}$$

for infinitely many k when $\mathcal{Z}_j'$ runs with the real protocol as described above.

We show that no simulator $\mathcal{S}_j$ can achieve property (4) in the ideal model, where $\mathcal{Z}_j'$ runs with CT_n and $\mathcal{S}_j$. To distinguish random variables during a run of $\mathcal{Z}_j'$ in the ideal model from those in the real model, we add a tilde to a random variable in a run of $\mathcal{Z}_j'$ in the ideal model, e.g., $\tilde{b}$, $\tilde{\beta}$.

For any $\mathcal{S}_j$ achieving indistinguishability of real and ideal model, this can happen only with negligible probability, so we can assume without losing generality that $\mathcal{S}_j$ always delivers outputs.

By construction of $\tilde{b}$ and κ, the variable $\tilde{b}$ and the tuple $(\tilde{M}, \kappa)$ are independent given $\tilde{b} \neq \perp$. Hence, since $\tilde{\beta}$ is a function of $\tilde{M}$ and κ,

$$\mathsf{Pr}\left[(\tilde{b}, \tilde{\beta}) = (0,1)\right] = \mathsf{Pr}\left[(\tilde{b}, \tilde{\beta}) = (1,1)\right]. \tag{5}$$

So comparing (4) and and (5), $\mathcal{Z}_j'$'s output distribution differs non-negligibly in real and ideal model. So no simulator $\mathcal{S}_j$ can simulate attacks carried out by $\mathcal{Z}_j'$ and $\mathcal{A}_j$, which gives the desired contradication. □

Combining the above Lemmas 4 and 5 we therefore get:

Theorem 6. *There is no non-trivial statistically universally composable protocol for $(m \to n)$-coin toss extension that proceeds in a polynomial of rounds.*

The case of perfect security is shown analogously.

Acknowledgements. This work was partially supported by the projects ProSecCo (IST-2001-39227) and SECOQC of the European Commission. Part of this work was done while the first author was with the IAKS, Universität Karlsruhe. Further, we thank the anonymous referees for valuable comments.

References

1. Andris Ambanis, Harry Buhrman, Yevgeniy Dodis, and Heinz Röhrig. Multiparty quantum coin flipping. In *19th Annual IEEE Conference on Computational Complexity, Proceedings of CoCo 2002*, pages 250–259. IEEE Computer Society, 2004.
2. Michael Backes, Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. On fairness in simulatability-based cryptographic systems. In *3rd ACM Workshop on Formal Methods in Security Engineering, Proceedings of FMSE 2005*, pages 13–22. ACM Press, 2005.

3. Michael Backes, Birgit Pfitzmann, and Michael Waidner. Secure asynchronous reactive systems. IACR ePrint Archive, March 2004.
4. Mihir Bellare, Juan A. Garay, and Tal Rabin. Distributed pseudo-random bit generators – a new way to speed-up shared coin tossing. In *Fifteenth Annual ACM Symposium on Principles of Distributed Computing, Proceedings of PODC 1996*, pages 191–200. ACM Press, 1996.
5. Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *Advances in Cryptology, A report on CRYPTO 81*, number 82-04 in ECE Report, pages 11–15. University of California, Electrical and Computer Engineering, 1982.
6. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.
7. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.
8. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive, January 2005. Full and revised version of [7].
9. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology, Proceedings of CRYPTO 2001*, number 2139 in Lecture Notes in Computer Science, pages 19–40. Springer-Verlag, 2001.
10. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, pages 494–503. ACM Press, 2002. Extended abstract.
11. J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, April 1979.
12. Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Eighteenth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1986*, pages 364–369. ACM Press, 1986.
13. Oded Goldreich. *Foundations of Cryptography – Volume 2 (Basic Applications)*. Cambridge University Press, May 2004.
14. Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. On the (im-)possibility of extending coin toss. IACR ePrint Archive, 2006. Full version of this paper.
15. Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Twenty-First Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1989*, pages 12–24. ACM Press, 1989. Extended abstract.
16. Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy, Proceedings of SSP 2001*, pages 184–200. IEEE Computer Society, 2001.
17. Douglas R. Stinson. Universal hash families and the leftover hash lemma, and applications to cryptography and computing. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 42:3–31, 2002.