Secure Physical Computation using Disposable Circuits

Ben A. Fisch[1], Daniel Freund[2], and Moni Naor[⋆3]

[1] Columbia University benafisch@gmail.com
[2] Cornell University freund90@mac.com
[3] Weizmann Institute of Science moni.naor@weizmann.ac.il

Abstract. In a *secure physical computation*, a set of parties each have physical inputs and jointly compute a function of their inputs in a way that reveals no information to any party except for the output of the function. Recent work in CRYPTO'14 presented examples of physical *zero-knowledge* proofs of physical properties, a special case of secure physical two-party computation in which one party has a physical input and the second party verifies a boolean function of that input. While the work suggested a general framework for modeling and analyzing physical zero-knowledge protocols, it did not provide a general theory of how to prove any physical property with zero-knowledge. This paper takes an orthogonal approach using *disposable circuits* (DC)—cheap hardware tokens that can be completely destroyed after a computation—an extension of the familiar tamper-proof token model. In the DC model, we demonstrate that two parties can compute any function of their physical inputs in a way that leaks at most 1 bit of additional information to either party. Moreover, our result generalizes to any multi-party physical computation. Formally, our protocols achieve unconditional UC-security with input-dependent abort.

1 Introduction

In a *secure two-party computation* (2PC), parties A and B each have secret inputs x_A and x_B respectively, and they jointly compute a function $f(x_A, x_B)$ such that each party receives the output of the function, but no further information about the other party's secret input. A *secure multi-party computation* (MPC) extends 2PC to an arbitrary number of parties each with private inputs to a multi-party function. As early as the 1980s, various results demonstrated that any two-party or multi-party function can be securely computed under standard cryptographic assumptions [21, 9].

In the present work, we consider secure computation in a physical context where each party's input is a physical entity. For example, suppose parties A

[⋆] Incumbent of the Judith Kleeman Professorial Chair. Research supported in part by grants from the Israel Science Foundation, BSF and Israeli Ministry of Science and Technology and from the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation.

and B wish to securely compute a function of their DNA (e.g. whether they are both carriers of a certain recessive gene). Unless the parties trust each other to honestly supply their genetic information as digital inputs, they cannot use standard secure 2PC to solve this task. They also may not trust one another to directly measure each other's genetic information and potentially learn more than the necessary information. To solve this task, the parties require a *secure physical computation* that is guaranteed to return to both parties the correct function output of the physical inputs and simultaneously prevent either party from learning more than the function output.

A special case of physical secure computation is a *physical zero-knowledge proof*, where a Prover must prove to a Verifier that a physical input object satisfies a physical property without revealing any further information about the input. Informal examples of physical zero-knowledge have appeared throughout the literature [19, 20]. Recent work of [4] introduced formal definitions and an analysis framework for physical zero-knowledge protocols, and gave examples of protocols for comparing DNA profiles and neutron radiographs of objects. Earlier work of [7, 8] used physical zero-knowledge techniques to demonstrate that two nuclear warheads have the same design without revealing further information on the design. The necessity for such a protocol arises in nuclear disengagement, where one country must prove to the other that it is destroying authentic nuclear weapons without revealing sensitive information on that weapon's design. Previous techniques relied on *information barriers*, or trusted devices that would measure the input objects and display only a red (reject) or green (accept) light. However, in absence of trust, information barriers are problematic as they may reveal too much information to the inspecting party, or may display false information. The protocols in [4] and [7] avoid information barriers and instead use techniques that physically manipulate the inputs and enable the inspecting party to verify physical properties without ever recording any sensitive information. Such protocols seem to be inherently problem-specific, and there is no general zero-knowledge protocol presented for proving any arbitrary physical property of any input.

We revisit the idea of an information barrier using special devices we call *disposable circuits*: instead of avoiding ever recording sensitive information, we suggest using cheap computing devices such as smart cards that can be destroyed or have their memory completely wiped after the computation. Each party will create a disposable device that can investigate the physical inputs and perform the necessary computations. Since neither party can trust the other party's device to act as a true information barrier, we can only allow party A's device to directly supply output to party B and party B's device to directly supply output to party A. Each party must remain isolated from its device during the computation, and the opposing party should be able to destroy or memory-wipe the device after the computation is complete. On the other hand, each party can only trust the correctness of its own device's output. Thus, the two parties need a secure protocol to verify the authenticity of the outputs received.

Many previous works have explored the use of physical hardware and physical separation in cryptographic protocols. Ben-Or et. al. [1] introduced the *multi-prover interactive proof system* (MIP) model, and showed that with two physically isolated provers it is possible to achieve unconditionally secure ZK proofs for NP. In addition, they showed similar results for bit commitment and Rabin-OT. Goldreich and Ostrovsky [10] demonstrated uses of tamper-proof hardware for the purpose of software protection. Katz [15] initiated a long line of works that used tamper-proof hardware to achieve universally composable (UC) security in multi-party computation protocols. In particular, Goyal et. al. [12] showed that tamper-proof hardware could be used to achieve unconditional UC-secure non-interactive multi-party computation.

Our model of disposable circuits is an extension of the standard stateful tamper-proof token model, which was first formalized by Katz [15]. The tokens in our model have the additional capability to measure physical inputs. In particular, a party in our model has the ability to create a token that will directly probe a specific physical input held by another party. We assume that both parties can reference the same object by some uniquely identifiable physical information even if all further physical characteristics of that object are secret. For example, the physical input might be a box held by the second party whose contents are secret, and yet both parties can identify the box by its exterior characteristics and physical location. The issuer of a token could certify that the token is investigating a specified object simply through physical observation. Alternatively, the token itself could be programmed to recognize the pre-specified characteristics used to reference the object. The motivation for this functionality is to remove any need for parties to trust each other to supply correct physical inputs or honest descriptions of physical inputs to the tokens.

Another nuance in our model is the assumption that any user of a token can destroy or memory-wipe that token. Katz's original model did not make any assumptions on the destruction of tokens, but several subsequent works considered tokens with a self-destruction capability. For example, Goldwasser et. al. [11] introduced *one-time memory* (OTM) tokens that immediately self-destruct after delivering output. Goyal et. al. used OTM tokens as well in their construction of unconditionally secure MPC. However, in these works, the self-destruct behavior was only used to make the hardware token tamper-proof against its user, and not to prevent its creator from recovering information. In fact, an important property of the standard tamper-proof token model is that it does not require mutual trust—each party only needs to trust the physical assumptions about its own hardware. This distinguishes the tamper-proof token model from similarly powerful models that require the parties to agree upon a trusted setup, such as the *common reference string* (CRS) model. Our disposable circuit model similarly only requires the device creator to trust a device's full functionality and tamper-proof properties, but we additionally require the user to trust one physical assumption: that the device can be destroyed.

We show that in this disposable circuit model, a prover party can prove any physical property of an arbitrary physical input to a verifier party in a way that leaks at most a single bit about the input. The high-level idea is to allow the verifier to create a tamper-proof device that will investigate the object in isolation. To restrict the communication between the device and verifier, the prover party will mediate all communication between the device and the verifier. The device will conditionally reveal to the prover a secret string after checking the validity of the physical property. The prover could send this string back to the verifier as evidence of the property's validity, but would need to ensure that this string does not reveal any further information to the verifier. To accomplish this, the prover and verifier will execute a secure 2PC protocol that verifies whether both parties "know" the same secret. The device can always communicate an arbitrary bit to the verifier because it can cause the protocol to abort dependent on the physical input. Crucially, the device is destroyed or memory-wiped at the end of the protocol so that it cannot reveal any further information.

More generally, we show that it is possible for two parties to evaluate any function of their physical inputs in a way that leaks at most one bit of information about the inputs. Each party can issue a token to investigate the opposing party's physical input and output a *message authentication code* (MAC) signature on the input description to the opposing party. The two parties then invoke a secure 2PC functionality that verifies the MAC signatures and computes the desired function. Moreover, we show that this approach can be easily extended for physical multi-party computation (MPC). Each party creates multiple tokens, one for each of the other parties. Each token investigates a single party's input and outputs a MAC signature on the input description to that party. The parties each collect the MAC signatures they have received from the other parties' tokens, and input these signatures along with their input descriptions to an MPC functionality that verifies the signatures and computes the desired function.

Building on the result that MPC can be unconditionally realized in the tamper-proof token model [12], our protocols are unconditionally secure (i.e. they do not rely on computational assumptions). We will analyze the security of our protocols using the notion of *security with input-dependent abort* introduced in [14], which cleanly captures the idea that an adversary can learn at most 1 bit about another party's input by causing the protocol to abort conditioned on that party's input.

Lastly, we present in Section 5 a protocol called *isolated circuit secure communication* that allows the two isolated disposable circuits to communicate secretly with each other without leaking any information to A or B and discuss its potential applications. We conclude in Section 6 with a question left open by our work.

2 Preliminaries

In this section we provide further details and formal specifications of the underlying definitions and concepts we address in this paper. We provide formal specifications in terms of *ideal functionalities* (see Canetti [2]). The ideal functionality definition of a protocol or token describes its target behavior in an *ideal world* using a trusted entity.

Disposable circuits model. Our formalization of the disposable circuits (DC) model is based on the stateful tamper-proof token model of Katz [15]. Katz formally defines a "wrapper" functionality $\mathcal{F}_{WRAP}$ to model a real world system in which any party can construct a hardware device encapsulating an arbitrary software program and pass it to a user party, who may only interact with the embedded software in a black-box manner. Formally, $\mathcal{F}_{WRAP}$ takes two possible commands: a "creation" command, which initializes a new token T_ρ implementing a stateful interactive polynomial-time program ρ (e.g., an ITM), and an "execution" command, which causes ρ to run on a supplied input and return output. Additionally, the token T_ρ may be partially isolated so that it cannot communicate with its creator and only interacts with a specified user. This is captured in $\mathcal{F}_{WRAP}$ by having the creator party P specify a user party P', whereafter $\mathcal{F}_{WRAP}$ stores (P, P', ρ) and only accepts subsequent "execution" commands from P' to run T_ρ.

We define the ideal functionality $\mathcal{F}_{DC}$ (see Figure 1) as an extension of $\mathcal{F}_{WRAP}$. We model the ability of tokens to directly measure physical inputs in the environment following the paradigm in [4]. Every real world physical object x is assigned a unique identifier id_x, which captures the public meta information that the parties in the real world system use to reference the same object. Upon receiving the identifier id_x, $\mathcal{F}_{DC}$ queries an ideal world oracle for a physical measurement $\mathcal{M}$ with id_x, which returns the logical output of the real world physical measurement $\mathcal{M}(x)$. However, without loss of generality, we may assume that $\mathcal{M}$ is an identity function (outputting a canonical description of x and all its physical characteristics), and hence we eliminate $\mathcal{M}$ from the formal description of $\mathcal{F}_{DC}$ for simplicity. Instead, upon receiving id_x as input to a token implementing ρ, $\mathcal{F}_{DC}$ directly computes $\rho(\bar{x})$ where $\bar{x}$ is the canonical description of x. We also model the capability of the token creator to certify that the token will be used only to investigate a specific object by optionally including id_x in the "creation" command. Additionally, we add a "destroy" command to $\mathcal{F}_{DC}$ that deletes the tuple representing an active token and outputs $\perp$ to the issuer of the command. This models the real world ability of users to destroy or memory-wipe hardware tokens at any point in time as well as certify that the device has been successfully destroyed.

Secure physical property verification (SPV). SPV involves two parties, a prover P and a verifier V. The prover holds a physical input x, and will allow the verifier to verify a physical property π of x, i.e. certify that $\pi(x) = 1$. The ideal functionality for SPV is described below in Figure 2, and only slightly modifies the physical zero-knowledge ideal functionality $\mathcal{F}_{ZK}^{\Pi}$ definition described in [4].

Secure physical two-party computation (2PC). Secure physical 2PC involves two parties, A and B, who each hold physical inputs x and y respectively.

Functionality $\mathcal{F}_{DC}$

$\mathcal{F}_{DC}$ is parametrized by a polynomial $p(\cdot)$ and an implicit security parameter k.

"Creation" Upon receiving $(\mathsf{create}, sid, P, P', id_x, mid, \rho)$ from a party P where ρ is the description of a deterministic program (e.g., an interactive Turing machine), mid is a machine id, and id_x either uniquely references a physical object x or is set to $\perp$:

1. Send $(\mathsf{create}, sid, P, P', id_x, mid, \rho)$ to P'.
2. Store $(P, P', id_x, mid, \rho, \emptyset)$.

"Execution" Upon receiving $(\mathsf{run}, sid, P, mid, msg)$ from party P':

1. Find the unique tuple $(P, P', id_x, mid, \rho, state)$. If no such tuple is stored, then do nothing.
2. If $id_x \neq \perp$ and $msg \neq id_x$, then do nothing.
3. If $msg = id_x$, set $\mathsf{input} = \bar{x}$, the canonical description x.
4. If $id_x = \perp$, then set $\mathsf{input} = msg$.
5. Run $\rho(\mathsf{input}, state)$ for at most $p(k)$ steps. Let $(out, state')$ denote the result. If ρ does not terminate in $p(k)$ steps, then set $out = \perp$ and $state' = state$.
6. Send (sid, P, mid, out) to P', store $(P, P', id_x, mid, \rho, state')$ and erase $(P, P', id_x, mid, \rho, state)$.

"Destroy" Upon receiving $(\mathsf{destroy}, sid, mid, P)$ from party P', erase any tuple of the form $(P, P', *, mid, *, *)$ and return $\perp$ to P'.

Fig. 1. Ideal world disposable circuit functionality

A function of x and y has two components: a pair of physical measurements $(\mathcal{M}_1, \mathcal{M}_2)$ and a mathematical function $f(\mathcal{M}_1(x), \mathcal{M}_2(y))$. However, in defining the ideal functionality $\mathcal{F}_{\mathsf{Physical}2PC}$, we may assume without loss of generality that $\mathcal{M}_1$ and $\mathcal{M}_2$ are "identity" measurements that each output a sufficiently detailed description of their physical inputs and relevant physical properties. Thus, our definition instead allows $\mathcal{F}_{\mathsf{Physical}2PC}$ to query an oracle $\mathcal{O}$ that outputs canonical descriptions of physical objects. The ideal functionality is formally defined in Figure 3.

Secure physical multi-party computation (MPC). Secure physical MPC naturally extends the definition of physical 2PC to N parties with N physical inputs $x_1, ..., x_N$. The ideal functionality $\mathcal{F}_{\mathsf{Physical}MPC}$ receives an input identifier id_{x_i} from each ith party, it derives the canonical description $\mathcal{O}(id_{x_i}) = \bar{x}_i$ for each i, computes a function $f(\bar{x}_1, ..., \bar{x}_N)$, and outputs this value to all parties.

Unconditional one-time MAC. We denote by (MAC, VF) an unconditionally secure one-time message authentication code scheme. A message m is

Functionality $\mathcal{F}_{SPV}^{\Pi}$

The functionality is parametrized by a physical property Π. We model the physical measurement $\mathcal{M}^{\Pi}$ for verifying Π as an ideal functionality $\mathcal{F}_{\mathcal{M}}^{\Pi}$. We assume any physical input x has a unique identifier id_x in the ideal world.

- Upon receiving $(id_x, \mathsf{Prover}, \mathsf{Verifier})$ from the party Prover, $\mathcal{F}_{SPV}^{\Pi}$ queries $\mathcal{F}_{\mathcal{M}}^{\Pi}$ to compute $\Pi(x)$, and sends $(\Pi(x), id_x, \Pi)$ to Verifier. Finally, if Verifier receives $\Pi(x) = 1$, it outputs ACCEPT. Otherwise Verifier outputs REJECT.

Fig. 2. Ideal world Secure Property Verification

Functionality $\mathcal{F}_{\mathsf{Physical}2PC}$

Upon initiation, the functionality is supplied with a function $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ and access to an oracle $\mathcal{O}$, which on input id_x identifying a physical object x outputs a canonical description including physical properties of x.

- Upon receiving id_x from the party A and id_y from the party B, query $\mathcal{O}$ and record $\bar{x} = \mathcal{O}(id_x)$ and $\bar{y} = \mathcal{O}(id_y)$. Output $f(\bar{x}, \bar{y})$ to both parties A and B.

Fig. 3. Ideal world physical 2PC

signed with a key k as $(m, MAC_k(m))$. The verification algorithm VF computes $VF_k(m, \sigma) = 1$ if and only if σ is a correct signature of m with key k. An example of an unconditional one-time MAC is the map $x \mapsto a \cdot x + b$ over a finite field, where the key $k = (a, b)$ is used to sign m as $a \cdot m + b$.

Proving security of protocols. Our security analysis uses the UC-framework [3]. We compare the execution of a *real* protocol ρ (defined in a given computational environment) with a static malicious adversary $\mathcal{A}$ corrupting a subset of the parties to an *ideal* process where the parties only interact with the ideal functionality $\mathcal{F}_\rho$ for ρ and an ideal world adversary $\mathcal{S}$, also called the simulator. When $\mathcal{A}$ (resp. $\mathcal{S}$) corrupts a party, it learns that party's entire state, and takes control of its communication. We define an environment $\mathcal{Z}$ that sets the parties' inputs in both executions and observes their outputs. Additionally, $\mathcal{Z}$ may communicate freely with the adversary $\mathcal{A}$ in the real protocol and $\mathcal{S}$ in the ideal protocol. Let $\mathrm{REAL}_{\rho,\mathcal{A},\mathcal{Z}}$ denote the output of $\mathcal{Z}$ after interacting with the real protocol and let $\mathrm{IDEAL}_{\mathcal{F}_\rho,\mathcal{A},\mathcal{Z}}$ denote its output after interacting with the ideal process.

Definition 1. *(Realizing an ideal functionality) A protocol ρ* ***securely realizes*** *the ideal functionality $\mathcal{F}_\rho$ if for any real world adversary $\mathcal{A}$ there exists an ideal adversary $\mathcal{S}$ such that* $\mathrm{REAL}_{\rho,\mathcal{A},\mathcal{Z}} \sim \mathrm{IDEAL}_{\mathcal{F}_\rho,\mathcal{S},\mathcal{Z}}$ *for every environment $\mathcal{Z}$.*

We will prove that our protocols satisfy a slightly relaxed definition of security called *security with input-dependent abort* [14], which allows an adversary to learn at most 1 bit of information by causing the protocol to abort dependent on another party's inputs. Formally, given any multi-party functionality $\mathcal{F}$, we define the functionality $\mathcal{F}^{\mathsf{IDA}}$ as follows. $\mathcal{F}^{\mathsf{IDA}}$ receives inputs from the parties and runs an internal copy of $\mathcal{F}$. In addition, it receives the description of a predicate ϕ from the adversary. Before delivering any output from $\mathcal{F}$, it evaluates ϕ with all the inputs. If ϕ outputs 1, then $\mathcal{F}^{\mathsf{IDA}}$ replaces the outputs to the honest parties with abort. Otherwise, $\mathcal{F}^{\mathsf{IDA}}$ delivers the correct outputs from $\mathcal{F}$ to both parties. We make only one modification to this definition to account for functionalities with physical inputs: if id_x denotes a physical input identifier, then ϕ is evaluated on the canonical description $\bar{x}$ of the object identified by id_x (along with the other inputs).

Definition 2. *(Security with input-dependent abort) A protocol ρ implementing the ideal functionality $\mathcal{F}_\rho$ is* ***secure with input-dependent abort*** *if ρ securely realizes $\mathcal{F}_\rho^{\mathsf{IDA}}$.*

3 Secure Property Verification

In this section, we present a solution for SPV that is secure with input-dependent abort. The general setup is as follows. The input to the protocol is the physical object x and a physical property Π. We express Π as a boolean function so that $\Pi(x) \in \{0, 1\}$. The verifier, denoted V, programs a DC token C_V to examine x, and compute $\Pi(x)$. In the formal description, C_V is replaced with the ideal functionality $\mathcal{F}_{DC}$, as described in Section 2. The token C_V must then communicate $\Pi(x)$ to V via an indirect channel through the prover, denoted P, who is mediating the communication. The prover's task is to limit this communication to a single bit.

It is straightforward for C_V to authenticate any message it sends to the verifier via the prover by signing the message using a MAC and a secret key it shares with V. In fact, since the message only consists of a single bit, it suffices for C_V to either send the secret key itself or nothing. More precisely, let C_V and V share a secret string sk of length λ (a security parameter). After investigating x, C_V sends sk to P if and only if $\Pi(x) = 1$. P must demonstrate to V that it "knows" sk as "evidence" that $\Pi(x) = 1$. P could simply send sk to V, but would need to ensure that V cannot learn any further information from sk. Namely, P should be convinced that V already "knows" sk as well.

Solutions from bit commitment. A natural way to convince P that V already knows sk is to use a commitment protocol in which V sends P a commitment to sk that C_V can later de-commit (without involving or even communicating with V). We discuss briefly a few solutions for this task that we can easily derive from existing protocols in the literature. One suitable protocol is the "commit and reveal" protocol of [1] for two isolated provers in the MIP model, where V acts as one prover and C_V as the second. The UC-secure commitment protocol of Moran and Segev [18] is also applicable, but their protocol uses V to send both the commitment and de-commitment while C_V is used for a validity check. It is easy to modify the protocol so that the token C_V de-commits the secret instead of V, although to maintain UC-security we would need to involve a third isolated token for the validity check (the simulator can extract the committed value by rewinding this token whereas it cannot rewind V). Another option is to adapt the standard construction of bit commitment from Rabin-OT [16]. In the generalized version of this construction using error-correcting codes (e.g., [6, 5]), the sender computes a randomized encoding $\mathsf{enc}(s)$ of the input string s to be committed, and obliviously transmits to the receiver a subset of the bits of $\mathsf{enc}(s)$ via Rabin-OT. To de-commit, the sender reveals s, and the receiver is able to check that $\mathsf{enc}(s)$ is consistent with the previously transmitted bits. In our setting, P would simply request from V to see a random subset of the bits of $\mathsf{enc}(s)$, and C_V would de-commit by sending s.

All of the above solutions would be unconditionally secure in our model, and we imagine it would not be hard to obtain UC-security as well. However, it is simpler to present and prove our feasibility result by making black-box use of previously defined functionalities that are known to be unconditionally UC-secure in the tamper-proof token model (and the disposable-circuit model by extension). The above solutions could not use the standard ideal commitment functionality $\mathcal{F}_{COM}$ in a black-box way because $\mathcal{F}_{COM}$ does not separate the party that sends the commitment from the party (or token) that reveals the commitment, which seems crucial in our application.

Solution from 2PC. In an effort to present and prove our result in the simplest possible way, we offer a solution using general 2PC, which can be realized with unconditional UC-security in the tamper-proof token model [12]. V and P will run a secure 2PC functionality that takes inputs x and y from each party, and either outputs 1 if $x = y$ and otherwise outputs abort. If both parties are honest, they will both supply the input sk. If the honest V receives the output 1, then it is convinced that P knows sk, which constitutes a proof that C_V verified $\Pi(x) = 1$ (up to the negligible probability that P guessed sk). More formally, given any UC-secure protocol implementing this 2PC functionality, if the protocol execution outputs 1 when V supplies sk, then there exists an efficient simulator that can extract sk from P. Even if V is dishonest, it can learn at most 1 bit from the protocol by causing the 2PC subprotocol to either abort or succeed dependent on the input that the honest P supplies. We formally describe the protocol in Figure 4 and prove that it securely realizes SPV with input-dependent abort.

Claim 1: *Protocol 1 securely realizes $\mathcal{F}^{\Pi}_{SPV}$ with input-dependent abort (unconditionally).*

Proof. Let π denote the real protocol according to the description of Protocol 1. We define the ideal world simulator $\mathcal{S}$ given any real world adversary $\mathcal{A}$ and

Protocol 1 - SPV

Setup: Let λ denote a security parameter. The Verifier chooses a random password string sk in $\{0,1\}^\lambda$. Let $\bar{x}$ denote a canonical description of the physical input x that the prover supplies. Let Π be a boolean function so that $\Pi(\bar{x}) \in \{0,1\}$. Recall that $\mathcal{F}_{DC}$ is able to derive $\bar{x}$ given id_x, a unique identifier for the object. The prover claims that $\Pi(\bar{x}) = 1$.

1. **(V initiates a token)** Let ρ define a program that on input $\bar{x}$ outputs sk if $\Pi(\bar{x}) = 1$ and outputs $\perp$ otherwise. V sends $(\mathsf{create}, sid, V, P, id_x, mid, \rho)$ to $\mathcal{F}_{DC}$.
2. **(P queries the token)** P submits $(\mathsf{run}, sid, V, mid, id_x)$ to $\mathcal{F}_{DC}$ and receives in response $\rho(\bar{x})$.
3. **(P destroys the token)** P submits $(\mathsf{destroy}, sid, mid, V)$ to $\mathcal{F}_{DC}$ and waits to receive $\perp$ in response.
4. **(2PC equality check)** P inputs $\rho(\bar{x})$ and V inputs sk to a protocol that unconditionally UC-realizes the following ideal functionality:
   - EqualityChecker: On input (msg, sid, P, V) from party P and (msg', sid, V, P) from party V, output 1 to both parties if $msg = msg'$, and otherwise output abort.
5. If V receives the output 1 from EqualityChecker, then it outputs Accept. If either party receives abort, then it outputs abort.

Fig. 4. SPV from secure 2PC.

consider separately the cases where $\mathcal{A}$ corrupts Prover and $\mathcal{A}$ corrupts Verifier. We may assume without loss of generality that $\mathcal{A}$ is a proxy for $\mathcal{Z}$. In other words, $\mathcal{Z}$ controls $\mathcal{A}$'s outgoing communication and $\mathcal{A}$ forwards all its incoming messages back to $\mathcal{Z}$.

$\mathcal{A}$ corrupts V. $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$. $\mathcal{S}$ intercepts the command $(\mathsf{create}, sid, V, P, id_x, mid, \rho)$ that $\mathcal{A}$ sends to $\mathcal{F}_{DC}$ as well as the value s that $\mathcal{A}$ submits to EqualityChecker. $\mathcal{S}$ defines the unary predicate ϕ_s on strings $m \in \{0,1\}^*$ so that $\phi_s(m) = 0$ when $\rho(m) = s$ and $\phi_s(m) = 1$ when $\rho(m) \neq s$. $\mathcal{S}$ then submits this predicate ϕ_s to $\mathcal{F}_{SPV}^{\mathsf{IDA}}$. Recall that $\mathcal{F}_{SPV}^{\mathsf{IDA}}$ will proceed identically to $\mathcal{F}_{SPV}$ on input id_x unless $\phi_s(\bar{x}) = 1$ (i.e. $\rho(\bar{x}) \neq s$). Thus, $\mathcal{S}$ will receive the output $(\Pi(x), id_x, 1)$ from $\mathcal{F}_{SPV}^{\mathsf{IDA}}$ if and only if $\mathcal{A}$ would receive the output 1 from EqualityChecker. In this case, $\mathcal{S}$ forwards 1 to $\mathcal{Z}$ and sets V's output to Accept. Likewise, $\mathcal{S}$ will receive the output abort if and only if $\mathcal{A}$ would receive abort from EqualityChecker. In this case, $\mathcal{S}$ sends abort to $\mathcal{Z}$ and sets V's output to abort as well. In the former case, the honest P has no output in either the ideal or real processes. In the latter case, the honest party P also outputs abort in both processes. Thus, $\mathcal{Z}$ has identical views in both processes.

$\mathcal{A}$ corrupts P. $\mathcal{S}$ runs a simulated $\mathcal{F}_{DC}$ with a simulated $\mathcal{A}$ just as V would, except that it chooses its own random password string sk' and defines ρ' to output sk' when $\Pi(x) = 1$. $\mathcal{S}$ records the value out that $\mathcal{A}$ receives back from $\mathcal{F}_{DC}$, and the value s' that the simulated $\mathcal{A}$ would submit to $\mathsf{EqualityChecker}$. If either $\mathsf{out} = \bot$ or $\mathsf{out} \neq s'$, then $\mathcal{S}$ sends the constant predicate $\phi = 1$ to $\mathcal{F}_{SPV}^{\mathsf{IDA}}$, causing both P and V to output abort, and sends abort directly to $\mathcal{Z}$ as well. Otherwise, if $\mathsf{out} = sk' = s'$, then $\mathcal{S}$ sends $(\Pi(x), id_x, 1)$ to $\mathcal{F}_{SPV}^{\mathsf{IDA}}$, causing V to output Accept.

In the case that the simulated $\mathcal{A}$ receives $\mathsf{out} = sk'$, the real $\mathcal{A}$ would receive sk from $\mathcal{F}_{DC}$. Let s denote the value that the real $\mathcal{A}$ would send as input to the $\mathsf{EqualityChecker}$. Since sk and sk' are identically distributed random strings, it follows that (sk, s) and (sk', s') must be identically distributed. Conditioned on $\mathsf{out} = sk'$, the probability that $s' \neq sk'$ is identical to the probability that $s \neq sk$, which both cause P and V to output abort in the real and ideal processes respectively. In this case, the view of $\mathcal{Z}$ is distributed identically in the real and ideal processes.

In the case that the simulated $\mathcal{A}$ receives $\mathsf{out} = \bot$, the real $\mathcal{A}$ also receives $\bot$. Here, $\mathcal{S}$ immediately causes an abort. The only way that $\mathcal{Z}$ might see different values in the real process is if $\mathcal{A}$ avoids an abort by guessing sk. This occurs with probability negligible in λ over V's random choice of sk.

We conclude that in both cases, $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \sim \mathrm{IDEAL}_{\mathcal{F}_{SPV}^{\mathsf{IDA}},\mathcal{S},\mathcal{Z}}$.

4 Secure Physical Two-Party Computation

We now have two parties, Party A and Party B, who respectively hold physical inputs x_A and x_B, and wish to evaluate a function of their inputs. Party A creates a DC token C_A to investigate x_B and party B creates a DC token C_B to investigate x_A. We assume that the function can be computed by measuring x_A and x_B separately in isolation, and then evaluating a logical function f of the measured values.[4] The parties exchange circuits so that C_A is isolated with x_B and only a communication channel to B while C_B is isolated with x_A and only a communication channel to A. Figure 5 depicts a schematic of the setup after the circuits are exchanged.

A high level description of the protocol is as follows. Each token computes a canonical description of the input it investigates, and outputs this description signed with an unconditional one time MAC to the opposing party. Let k_A and k_B denote secret keys that A shares with C_A and B shares with C_B respectively. C_A (investigating x_B) derives the description $\bar{x}_B$ and sends $(\bar{x}_B, MAC_{k_A}(\bar{x}_B))$ to B. C_B (investigating x_A) derives the description $\bar{x}_A$ and sends $(\bar{x}_A, MAC_{k_B}(\bar{x}_A))$ to A. Parties A and B also derive the canonical descriptions of their own inputs to check that the descriptions they receive from the tokens are correct. Next, A and B execute a secure 2PC functionality that verifies the MAC signatures and computes the function f. A supplies the inputs $(\bar{x}_A, MAC_{k_B}(\bar{x}_A), k_A)$ and B supplies the inputs $(\bar{x}_B, MAC_{k_A}(\bar{x}_B), k_B)$. The 2PC functionality outputs abort to both parties if either MAC verification fails, and otherwise outputs $f(\bar{x}_A, \bar{x}_B)$ to both parties. We describe the details of the protocol formally in Figure 6.

[4] We do not posit that this covers all physical functions (e.g, in the quantum domain).

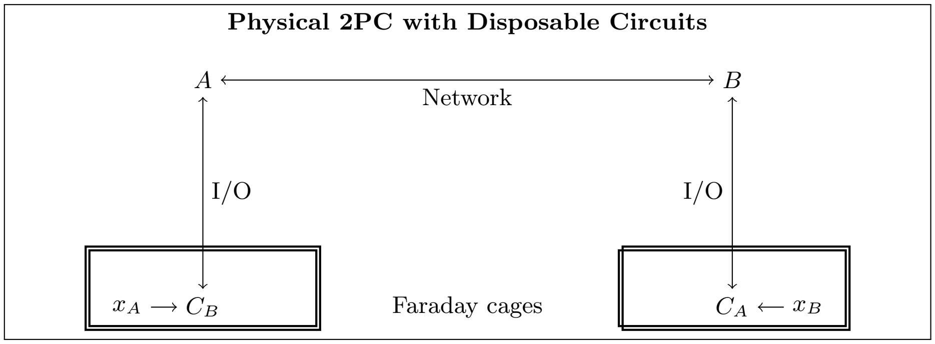
Physical 2PC with Disposable Circuits
A
Network
B
I/O
I/O
$x_A \longrightarrow C_B$
Faraday cages
$C_A \longleftarrow x_B$

Fig. 5. Schematic of the physical setup for secure physical 2PC using disposable circuits (Protocol 2).

Alternatively, we could use C_A and C_B to locally compute the outputs first, send the outputs signed with a MAC to the opposing party. Then A and B could execute a 2PC functionality that both verifies the signatures and checks that the outputs are equal, outputting 1 if these checks pass. This should be a more efficient method in general since f could be an arbitrarily complex function and costly to evaluate inside the 2PC functionality. This modified protocol uses the same general technique as the *dual-execution garbled circuits* (DualEx) protocol for 2PC [17, 13], although the goal of DualEx was to achieve a tradeoff between security and efficiency in standard 2PC. Nonetheless, we find the former method simpler to present and prove secure.

Claim 2: *Protocol 2 securely realizes $\mathcal{F}_{\mathsf{Physical2PC}}$ with input-dependent abort.*

Proof. Let π denote the real protocol according to the description of Protocol 1. We define the ideal world simulator $\mathcal{S}$ given any real world adversary $\mathcal{A}$. It suffices to analyze the case where $\mathcal{A}$ corrupts A, as the case in which $\mathcal{A}$ corrupts B is symmetrical. As before, we may assume without loss of generality that $\mathcal{A}$ is a proxy for $\mathcal{Z}$.

$\mathcal{S}$ runs a simulated copy of $\mathcal{A}$. $\mathcal{S}$ intercepts id_{x_B} and $\rho_{\mathcal{A}}$ from $\mathcal{A}$'s create message to $\mathcal{F}_{DC}$. In addition, $\mathcal{S}$ runs a simulated $\mathcal{F}_{DC}$. $\mathcal{S}$ chooses a random secret key r, defines $\rho_{\mathcal{S}}(y) = (y, MAC_r(y))$, and sends $(\mathsf{create}, sid_{\mathcal{S}}, \mathcal{S}, A, id_{x_B}, mid_{\mathcal{S}}, \rho_{\mathcal{S}})$ to $\mathcal{F}_{DC}$. Next, $\mathcal{S}$ intercepts id_{x_A} from $\mathcal{A}$'s run command to $\mathcal{F}_{DC}$. $\mathcal{S}$ sends $(\mathsf{run}, sid_{\mathcal{S}}, \mathcal{S}, mid_{\mathcal{S}}, id_{x_A})$ to the simulated $\mathcal{F}_{DC}$, and forwards the resulting output $(\bar{x}_A, MAC_r(\bar{x}_A))$ to $\mathcal{A}$ and to $\mathcal{Z}$. $\mathcal{S}$ intercepts the input (m', σ', k') that $\mathcal{A}$ submits to $\mathsf{VFChecker}$. If $VF_r(m', \sigma') \neq 1$ then $\mathcal{S}$ sets $\phi = 1$. Otherwise, $\mathcal{S}$ sets the predicate ϕ such that $\phi(x, y) = 1$ if and only if $VF_{k'}(\rho_{\mathcal{A}}(y)) \neq 1$. Finally, $\mathcal{S}$ sends id_{x_A} and ϕ to $\mathcal{F}^{IDA}_{\mathsf{Physical2PC}}$. $\mathcal{S}$ forwards the output it receives, whether abort or $f(\bar{x}_A, \bar{x}_B)$, to $\mathcal{Z}$ and sets A's output accordingly.

In the real protocol, $\mathcal{Z}$ sees $(\bar{x}_A, MAC_{k_B}(\bar{x}_A))$ and in the ideal world it sees $(\bar{x}_A, MAC_r(\bar{x}_A))$ where k_B is the random value chosen by A and r is the random

Protocol 2 - Physical 2PC

Setup: Let λ denote a security parameter. Let (MAC, VF) denote an unconditional one-time message authentication scheme. Party A has physical input x_A identified by id_{x_A} and party B has physical input x_B identified by id_{x_B}. Let $\bar{x}_A$ denote a canonical description of x_A and $\bar{x}_B$ a canonical description of x_B. Parties A and B share a function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$.

1. **(A initiates a token)** Choose a random string $k_A \in \{0,1\}^\lambda$. Let ρ_A define a program that on input y outputs (y, σ_A) where $\sigma_A = MAC_{k_A}(y)$. A sends $(\mathsf{create}, sid_A, A, B, id_{x_B}, mid_A, \rho_A)$ to $\mathcal{F}_{DC}$.
2. **(B initiates a token)** Choose a random string $k_B \in \{0,1\}^\lambda$. Let ρ_B define a program that on input y outputs (y, σ_B) where $\sigma_B = MAC_{k_B}(y)$. B sends $(\mathsf{create}, sid_B, B, A, id_{x_A}, mid_B, \rho_B)$ to $\mathcal{F}_{DC}$.
3. **(A queries B's token)** A submits $(\mathsf{run}, sid_B, B, mid_B, id_{x_A})$ to $\mathcal{F}_{DC}$, receives in response $(\mathsf{out}_B, \sigma_B)$, and checks that $\mathsf{out}_B = \bar{x}_A$.
4. **(B queries A's token)** A submits $(\mathsf{run}, sid_A, A, mid_A, id_{x_B})$ to $\mathcal{F}_{DC}$, receives in response $(\mathsf{out}_A, \sigma_A)$, and checks that $\mathsf{out}_A = \bar{x}_B$.
5. **(The parties destroy the tokens)** A submits $(\mathsf{destroy}, sid_B, mid_B, B)$ to $\mathcal{F}_{DC}$ and waits to receive $\perp$ in response. B submits $(\mathsf{destroy}, sid_A, mid_A, A)$ to $\mathcal{F}_{DC}$ and waits to receive $\perp$ in response.
6. **(2PC check signatures and compute f)** A submits inputs $(\bar{x}_A, \sigma_B, k_A)$ and B submits inputs $(\bar{x}_B, \sigma_A, k_B)$ to a protocol that unconditionally UC-realizes the following ideal functionality:
   - VFChecker: On input (m, σ, k_A) from party A and (m', σ', k_B) from party B, check that $VF_{k_A}(m', \sigma') = 1$ and check that $VF_{k_B}(m, \sigma) = 1$. If these checks pass, compute and output $f(m, m')$ to both parties. Otherwise, output abort.
7. If VFChecker outputs abort, then both A and B output abort. Otherwise, VFChecker outputs $f(\bar{x}_A, \bar{x}_B)$ to both A and B, who each locally output this value as well.

Fig. 6. Secure physical 2PC leaking at most 1 bit.

value chosen by $\mathcal{S}$. These values are identically distributed. Thus, the probability $\mathcal{A}$ responds in the real world with (m, σ, k) such that the verification of the honest party's MAC fails, i.e. $VF_{k_B}(m, \sigma) \neq 1$, is identical to the probability that the simulated $\mathcal{A}$ responds with (m', σ', k') such that $VF_r(m', \sigma') \neq 1$. Likewise, the outcome of the verifying the adversary's MAC in the real world, i.e. $VF_{k'}(\rho_{\mathcal{A}}(\bar{x}_B))$, is identically distributed to $VF_k(\rho_{\mathcal{A}}(\bar{x}_B))$. Thus, the probability that one of the verifications fails in the real world is identical to the probability that one of the verifications fails in the simulation. This event causes the same outcome in the real process and ideal process, namely abort.

On the other hand, when both MAC verifications pass, in the ideal world the output is always the true output $f(\bar{x}_A, \bar{x}_B)$, whereas in the real world the output is $f(m, \bar{x}_B)$ possibly for $m \neq \bar{x}_A$. Nonetheless, when $m \neq \bar{x}_A$, the probability that verifying the honest party's MAC passes, i.e. $VF_{k_B}(m,\sigma) = VF_{k_B}(m, MAC_{k_A}(\bar{x}_A)) = 1$, is negligible by reduction to the security of (MAC, VF).

We conclude that $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \sim \mathrm{IDEAL}_{\mathcal{F}^{\mathsf{IDA}}_{\mathsf{Physical2}PC},\mathcal{S},\mathcal{Z}}$

5 Secure Physical Multi-Party Computation

Our protocol for secure physical 2PC can be generalized for n parties computing a multi-party functionality of their n physical inputs $x_1, ..., x_n$. We will sketch the protocol without going into low-level detail, as the extension is quite natural. In the first phase of the protocol, each party P_i creates $n-1$ tokens $\{T^i_j\}_{j \neq i}$, and transfers the token T^i_j to party P_j. The token T^i_j investigates the input x_j and outputs to P_j the canonical description $\bar{x}_j$ of x_j and a MAC signature using P_i's key k_i: $(\bar{x}_j, MAC_{k_i}(\bar{x}_j))$. At the end of the first phase, each party P_i has received $n-1$ MAC signed descriptions of its input x_i from the other parties' tokens. Each P_i checks that the $n-1$ descriptions it has received are all the same, and also match the correct canonical description $\bar{x}_i$ that it has derived on its own. In the second phase, each party P_i inputs $(\bar{x}_i, k_i, \{MAC_{k_j}(\bar{x}_i)\}_{i \neq j})$ to a secure MPC protocol implementing a functionality that verifies all the MAC signatures and either outputs abort (when a signature is invalid) or otherwise returns $f(\bar{x}_1, ..., \bar{x}_n)$ to each party. The resulting protocol is secure with input-dependent abort, as an adversary and its tokens can at best cause the MPC protocol to either succeed or fail depending on the inputs they investigate and the adversary's input. The MPC protocol can be unconditionally UC-securely realized in the DC model (as an extension of the tamper-proof token model) following the result of Goyal et. al. [12].

6 Isolated Circuit Secure Communication

In this section we show a simple protocol we call *isolated circuit secure communication* (ICSC) that enables the two isolated circuits C_A and C_B in the setup depicted in Figure 5 to send messages to each other in a way that keeps the messages hidden from both A and B. The protocol is *malleable*, meaning that A and B can sabotage any message sent between the circuits without detection. However, all messages that A and B see during the protocol reveal no information (are uniformly distributed) even if A colludes maliciously with C_A or B colludes maliciously with C_B.

Security (sketch). We do not precisely formalize the security of the ICSC protocol (e.g., in the UC-framework), but we sketch here the security intuition. First, it is easy to see the the output of the protocol to C_A and C_B respectively is correct when all parties are honest. Second, the views of A and B are indistinguishable from random. Consider the view of an adversary $\mathcal{A}$ corrupting A (the case for B is symmetric). In the first round, $\mathcal{A}$ receives $k^0_B \oplus m$ for some m and in the second round it receives $k^1_B \oplus m'$ for some m'. Given that k^0_B and k^1_B

Protocol 3 - ICSC

C_A and A share secret key k_A^0 and k_A^1; C_B and B share a secret keys k_B^0 and k_B^1.

1. *C_A sends message m_0 to C_B*:
   - C_A to B: send $u_0 = k_A^0 \oplus m_0$
   - B to A: send $v_0 = k_B^0 \oplus u_0$
   - A to C_B: send $w_0 = k_A^0 \oplus v_0 = k_B^0 \oplus m_0$
   - C_B: output $m_0 = k_B^0 \oplus w_0$
2. *C_B sends message m_1 to C_A*:
   - C_B to A: send $v_1 = k_B^1 \oplus m_1$
   - A to B: send $u_1 = k_A^1 \oplus v_1$
   - B to C_A: send $w_1 = k_B^1 \oplus u_1 = k_A^1 \oplus m_1$
   - C_A: output $m_1 = k_A^1 \oplus w_1$

Multiple rounds: Using a pseudorandom function or any KDF, on round i party A uses a key pair $\mathrm{KDF}(k_A^0; k_A^1; i)$, and party B uses a key pair $\mathrm{KDF}(k_B^0; k_B^1; i)$.

Fig. 7. Isolated circuits C_A and C_B exchange secret messages indirectly through A and B.

are independent uniformly distributed random strings, the simulator can replace both of $\mathcal{A}$'s messages with independent random strings. The argument can be extended to prove computational security in the multi-round setting via a standard hybrid argument, replacing the PRF generated keys with uniform random strings.

Applications. ISCS enables two tokens in isolation to execute arbitrary two-party interactive (digital) functionalities without leaking information to the two parties involved (other than 1 bit by aborting). In theory, this is no more powerful than the protocol we presented for physical two-party computation. However, we can imagine ISCS could have practical import. For example, we could use ISCS to efficiently compile any interactive protocol that leaks up to k bits into one that leaks at most 1 bit of the initial k bits. The isolated tokens could first run the interactive protocol to learn the output and at most k additional bits. Next, the tokens would communicate their outputs to the two parties using the DualEx paradigm (they send their MAC signed output to the opposing party, and the parties run secure 2PC to verify the MACs and equality of the outputs).

7 Further directions

Open question: *Is there a protocol for physical SPV or 2PC/MPC in the DC model that is fully secure against malicious adversaries, or that provably does not leak any information?*

Acknowledgements

We would like to thank the anonymous reviewers of this paper for their many helpful comments and suggestions.

References

1. M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *20th ACM STOC*, pages 113–131. ACM Press, May 1988.
2. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
3. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE, Oct. 2001. Full version at Cryptology ePrint Archive, Report 2001/055, http://eprint.iacr.org/2001/055.
4. B. Fisch, D. Freund, and M. Naor. Physical zero-knowledge proofs of physical properties. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014*, volume 8617, pages 313–336. Springer, Aug. 17–21, 2014.
5. T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. MiniLEGO: Efficient secure two-party computation from general assumptions. In *Eurocrypt 2013*, LNCS, pages 537–556. Springer, 2013.
6. J. A. Garay, Y. Ishai, R. Kumaresan, and H. Wee. On the complexity of UC commitments. In P. Q. Nguyen and E. Oswald, editors, *Eurocrypt 2014*, volume 8441 of *LNCS*, pages 677–694. Springer, May 11–15, 2014.
7. A. Glaser, B. Barak, and R. J. Goldston. A new approach to nuclear warhead verification using a zero-knowledge protocol. Presented at 53rd Annual INMM (Institute of Nuclear Materials Management) meeting, 2012.
8. A. Glaser, B. Barak, and R. J. Goldston. A zero-knowledge protocol for nuclear warhead verification. *Nature*, 510:497–502, 2014.
9. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In A. Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
10. O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.
11. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. One-time programs. In D. Wagner, editor, *Crypto 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Aug. 17–21, 2008.
12. V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In *2010*, volume 5978 of *LNCS*, pages 308–326. Springer, 2010.
13. Y. Huang, D. Evans, and J. Katz. Quid-Pro-Quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE Symposium on Security and Privacy*, pages 272–284, 2012.
14. Y. Ishai, E. Kushilevitz, R. Ostrovsky, M. Prabhakaran, and A. Sahai. Efficient non-interactive secure computation. In K. G. Paterson, editor, *Eurocrypt 2011*, volume 6632 of *LNCS*, pages 406–425, Tallinn, Estonia, May 15–19, 2011. Springer.
15. J. Katz. Universally composable multi-party computation using tamper-proof hardware. In M. Naor, editor, *Eurocrypt 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, May 20–24, 2007.

16. J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.
17. P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *(PKC 2006)*, volume 3958 of *LNCS*, pages 458–473. Springer, Apr. 2006.
18. T. Moran and G. Segev. David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In N. P. Smart, editor, *Eurocrypt 2008*, volume 4965 of *LNCS*, pages 527–544. Springer, Apr. 13–17, 2008.
19. M. Naor, Y. Naor, and O. Reingold. Applied kid cryptography or how to convince your children you are not cheating. *Journal of Craptology*, 0(1), April 1999.
20. J.-J. Quisquater, L. Guillou, M. Annick, and T. Berson. How to explain zero-knowledge protocols to your children. In *Proceedings on Advances in Cryptology*, CRYPTO '89, pages 628–631, New York, NY, USA, 1989. Springer-Verlag New York, Inc.
21. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.