Physical Zero-Knowledge Proofs of Physical Properties

Ben 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. Is it possible to prove that two DNA-fingerprints match, or that they do not match, without revealing *any* further information about the fingerprints? Is it possible to prove that two objects have the same design without revealing the design itself? In the digital domain, *zero-knowledge* is an established concept where a prover convinces a verifier of a statement without revealing any information beyond the statement's validity. However, zero-knowledge is not as well-developed in the context of problems that are *inherently physical*. In this paper, we are interested in protocols that prove physical properties of physical objects without revealing further information. The literature lacks a unified formal framework for designing and analyzing such protocols. We suggest the first paradigm for formally defining, modeling, and analyzing *physical zero-knowledge* (PhysicalZK) protocols, using the Universal Composability framework. We also demonstrate applications of physical zero-knowledge to DNA profiling and neutron radiography. Finally, we explore *public observation proofs*, an analog of public-coin proofs in the context of PhysicalZK.

1 Introduction

Zero-knowledge proofs are protocols that prove an assertion without revealing any information beyond that assertion's validity. Zero-knowledge proofs were first introduced by Goldwasser, Micali, and Rackoff in 1985 [16]. The power of zero-knowledge proofs is quite remarkable: anything that can be proved efficiently can be proved with a zero-knowledge protocol, under the cryptographic assumption that one-way functions exist (see Goldreich [9]).

Zero-knowledge proofs have also been considered in a physical setting. A number of works have explored constructions of zero-knowledge protocols that can be physically implemented [26, 19, 24, 23]. One goal of those works was to design protocols with simple procedures and security arguments that the participating parties could easily understand. An added advantage of simple physical protocols is that humans can implement them without the aid of computers. Moran and Naor [24] give methods for polling people on sensitive issues using physical envelopes as an alternative to electronic polling, where humans might not trust computers to behave honestly. Many works have also addressed the incorporation of physical hardware into broader cryptographic schemes. In some cases, these hybrid protocols achieve efficiency or security gains that are unachievable in a standard computation model. Examples of physically realizable functionalities that have been suggested for aiding general cryptographic protocols include tamper-evidence [23], tamper-proof tokens [12, 7, 21, 22, 25, 20, 18], one-time programs [15], and physically uncloneable functions [3].

Previous literature on zero-knowledge in a physical setting addressed physical protocols for tasks that could otherwise be solved digitally. There is comparatively little formal work on protocols for inherently physical tasks that cannot be solved digitally. One example that has been studied rigorously is *distance bounding protocols*, introduced by Brands and Chaum in 1993 [2], in which a verifier party determines or verifies an upper bound on its physical distance to a prover party. In 2012, Glaser, Barak, and Goldston [8] suggested applying zero-knowledge concepts to the task of

⋆ Most of the work was done while the authors were visiting the Weizmann Institute of Science. Research in part supported by the Kupcinetz-Getz Summer Science School.
⋆⋆ Incumbent of the Judith Kleeman Professorial Chair.

proving that a nuclear weapon is authentic without revealing sensitive information about its actual design, a problem that arises in the context of nuclear disengagement treaties. They presented an ϵ-knowledge protocol for this task, but did not have a rigorous framework for formally defining and analyzing the protocol's ϵ-knowledge security.

Our contributions. We present the first formal treatment of *physical zero-knowledge* (PhysicalZK) proofs for inherently physical claims. In our setting, a prover convinces a verifier that an input object satisfies a given physical property. Our framework for designing and analyzing PhysicalZK protocols uses the *Universally Composable* (UC) security framework [4], popularly applied in analysis of hybrid protocols involving physical hardware.

Expanding on Glaser et al., we present the first PhysicalZK protocols for the nuclear verification problem, or the general task of verifying object neutron radiograph equality. We also demonstrate an application of PhysicalZK proofs to DNA profiling in which a prover (e.g. a suspect) convinces a verifier (e.g. the police) that its DNA profile does not match a target profile (e.g. obtained from a crime scene) without revealing to the verifier any further information about the profiles, and discuss a protocol for parental testing.

A further goal of our work is to initiate a rigorous study into the foundations of physical zero-knowledge. We point out both differences and similarities between physical and standard ZK where they arise. In particular, Section 3 compares the UC properties of physical vs. digital ZK, and Section 6 explores a physical analog of public coin proofs.

2 What is physical zero-knowledge (PhysicalZK)?

A standard zero-knowledge proof involves a binary relation R and an input x. A prover convinces a verifier that there exists a witness w such that $(x, w) \in R$. The verifier "learns nothing" from the protocol except the existence of w, and possibly the fact that the prover "knows" w. (See Goldreich [9] for formal definitions, classical theorems, and variants of zero-knowledge).

Previously, the term *physical zero-knowledge* was used for physically implemented ZK protocols, involving physical tools such as scissors, playing cards, envelopes, or pez dispensers. However, the underlying tasks in those protocols were still logical in nature (e.g. solving a Sudoku puzzle [19], finding Waldo [26]).

In our definition of physical zero-knowledge (PhysicalZK), a prover convinces a verifier that a physical input object has a physical property Π. The verifier should "learn nothing" except the validity of the statement "X satisfies Π." A physical measurement M verifies Π, possibly requiring the assistance of a measurement device D. Asymmetry between the prover and the verifier arises not from secret knowledge or computational power, but from *access permissions* to the object and measurement device. Since a verifier might forcefully break its restricted access, the threat model we consider only addresses adversaries that avoid being caught (similar to the *covert adversary* model [1]). Before proceeding, we give a few simple examples.

1. **Coke vs. Pepsi "blind test":** Alice demonstrates to Bob her ability to distinguish between the tastes of Coke and Pepsi using the classic blind test. However, the simplest test is not zero-knowledge. Bob might give Alice a cup of Sprite, and gain information from her response. One fix is to use indistinguishable coffee lids. Alice observes that Bob prepares cups of Coke and Pepsi. Bob then supplies Alice with the lids, Alice marks the inside of each lid with her secret signature, and covers the cups. After the blind test, Alice commits her response on a piece of paper. But before handing the paper to Bob, she will remove the lid, and check for her signature.

2. **Bins and Balls Equality:** Alice proves to Bob that two bins X and Y (of capacity n) contain the same number of balls. The following ϵ-knowledge protocol was given in [8]. Alice chooses $N > n$, and prepares two new pairs of bins, each of capacity $N + n$, labelled B_0 and B_1 respectively. Alice chooses two independent random values

r_0 and r_1 uniformly distributed in $[0, N)$. Concealing the bins from Bob, she adds r_0 balls to each bin in B_0, and r_1 to each bin in B_1. Bob randomly selects $i \in \{0, 1\}$, Alice hands Bob the pair B_i, and Bob checks that both bins in the pair have equal numbers of balls. Alice then pours the contents of X into one bin in the remaining pair, and the contents of Y into the other. Finally, Bob checks the final contents of the bins to verify that they contain equal numbers of balls. Alice's success of cheating is at most 1/2. (Appendix A contains a full analysis).

3. **Litmus test:** Alice proves to Bob that her solution is basic/acidic without revealing the actual pH. Blue litmus paper turns red in acidic solution, and red litmus paper turns blue in basic solution. First, Bob tests Alice's litmus paper in known basic/acidic solutions to check that it operates correctly. After the protocol is complete, the litmus paper must be completely destroyed (to prevent Bob from later examining traces of the solution remaining on the paper).

3 PhysicalZK in the UC security framework

The UC framework. The Universally Composable security framework (UC) of Canetti [4] defines two worlds: the "real" world in which the real protocol is executed, and the "ideal" world in which an ideal process is implemented with the help of a trusted third party. A protocol *environment machine* $\mathcal{Z}$ interacts with the protocols in both worlds, setting each party's inputs, and reading their outputs. Although $\mathcal{Z}$ does not see internal communication between parties, it communicates freely with an adversary $\mathcal{A}$. When $\mathcal{A}$ *corrupts* a party, it assumes the party's identity, and takes control of its communication. A real protocol *UC-emulates* an ideal process if for every real adversary $\mathcal{A}$ there exists an ideal world adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between its interactions with $\mathcal{A}$ in the real protocol and $\mathcal{S}$ in the ideal process. The *universal composition* theorem states that if π is a protocol involving sub-protocol calls to an ideal functionality $\mathcal{F}$, ρ is a protocol that UC-emulates $\mathcal{F}$, and $\pi^{\rho/\mathcal{F}}$ is the *hybrid protocol* obtained by replacing calls to $\mathcal{F}$ in π with calls to ρ, then $\pi^{\rho/\mathcal{F}}$ UC-emulates π.

Modeling physical protocols. We separate physical protocols into a *logical layer* and a *physical layer*. All the physical operations of the protocol belong to the physical layer. Every physical operation serves an ideal function, and can be modeled by an ideal process in an abstract computation model with interactive turing machines (ITMs). This translation is based on physical assumptions. The logical layer is the hybrid world protocol obtained by replacing all physical operations with oracle calls to their ideal functionalities.

For example, consider the operation of pouring x balls into a bin and sealing the bin. We can define an ideal functionality $\mathcal{T}$ and an ideal process for this operation as follows. $\mathcal{T}$ stores tuples of the form $(value, id, creator, holder, state)$. Upon receiving the two commands **Create**(x, id) and **Seal**(id) from party P_i, $\mathcal{T}$ stores the tuple $(x, id, P_i, P_i, sealed)$, and will deny requests to view the value x that come from any party other than P_i. However, any party P_j may send a special command **Force**(id) to $\mathcal{T}$, and $\mathcal{T}$ will respond by sending the entire tuple to P_j and broadcasting to all other participating parties that P_j issued the **Force** command. This emulates the real behavior of a party who forcefully breaks open the sealed bin without permission, and is labeled a cheater.

Rigorous analysis can be applied to the hybrid world logical layer. We can then interpret the universal composition property of our model as formally reducing security to the most basic physical assumptions necessary: if the hybrid world logical layer UC-emulates $\mathcal{F}$, then any real world physical protocol emulating the hybrid protocol also realizes $\mathcal{F}$.

UC physical commitments. Bit-commitment is impossible to UC-realize in the standard computation model without trusted setup assumptions [5]. However, physical assumptions change matters. Consider the following trivial protocol in which the parties continuously observe each other.

Alice commits to her bit by placing it in a sealed container, and de-commits by opening the container. To prevent Bob from forcibly cheating, Alice could run the protocol behind a secure glass screen (see Section 6 on *public observation protocols*). There is also a more sophisticated UC secure bit-commitment protocol using tamper-evident envelopes [23], which does not require continuous observation.

Likewise, ZK is not UC-realizable without setup assumptions, but there are UC-secure ZK proofs for any NP relation given UC bit-commitment [5, 6]. Therefore, it is possible to implement UC-secure ZK protocols for any NP relation using UC physical commitments.

Ideal functionality $\mathcal{F}_{ZK}^{\Pi}$. The ideal functionality $\mathcal{F}_{ZK}^{\Pi}$ is described in Figure 1, running with parties Prover, Verifier, and an oracle $\mathcal{F}_{\mathcal{C}}$ that compiles the ideal functionalities for a collection $\mathcal{C}$ of physical operations in the real world.

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

Π is a unary predicate representing a physical property Π. If id_X uniquely identifies a physical set X, the statement $id_X \in \Pi$ translates the physical world statement "X satisfies property Π." $\mathcal{F}_{\mathcal{C}}$ includes an ideal functionality $\mathcal{F}_{M^{\Pi}}$ for the physical measurement operation M^{π} required to verify Π, which outputs $\Pi(id_X) \in \{0, 1\}$. The parameter $\mathsf{leak}(id_X)$ represents information that is leaked when Verifier forcefully cheats.

- Upon receiving $(id_X, pid_{\mathcal{C}}, \mathsf{Prover}, \mathsf{Verifier})$ from the party Prover, $\mathcal{F}_{ZK}^{\Pi}$ queries the $\mathcal{F}_{\mathcal{C}}$ specified by the process identifier $pid_{\mathcal{C}}$ to compute $\Pi(id_X)$, and sends $(\Pi(id_X), id_X, \Pi)$ to Verifier.
- Upon receiving the instruction cheat from Verifier, send $(id_X, \mathsf{leak}(id_X))$ to Verifier, and send $(\mathtt{Cheater}, \mathsf{Verifier})$ to Prover. If Prover sends cheat, send $(\mathtt{Cheater}, \mathsf{Prover})$ to Verifier. Upon receiving the instruction fail from either party, send `Failed` to both parties.

Fig. 1. Ideal world PhysicalZK

Let $\rho^{\mathcal{F}_{\mathcal{C}}/\mathcal{C}}$ denote the $\mathcal{F}_{\mathcal{C}}$-hybrid model translation of a physical protocol ρ with physical operation collection $\mathcal{C}$. A proof that $\rho^{\mathcal{F}_{\mathcal{C}}/\mathcal{C}}$ UC-emulates $\mathcal{F}_{ZK}^{\Pi}$ captures (up to physical assumptions) that ρ is secure against any adversary in the real physical world whose behavior is restricted to operations in $\mathcal{C}$.[4] A generic procedure for this analysis is outlined in Figure 2. Appendix A includes a full UC modeling and security proof for the Bins and Balls Equality protocol of [8]. Let $\text{HYBRID}_{\rho,\mathcal{F}_{\mathcal{C}},\mathcal{A},Z}$ and $\text{IDEAL}_{\mathcal{F}_{ZK}^{\Pi},\mathcal{F}_{\mathcal{C}},\mathcal{S},\mathcal{Z}}$ respectively denote the random variables describing the output of environment $\mathcal{Z}$ after interacting with $\mathcal{A}$ in $\rho^{\mathcal{F}_{\mathcal{C}}/\mathcal{C}}$ and $\mathcal{S}$ in the ideal process for $\mathcal{F}_{ZK}^{\Pi}$.

Definition 1. [5] *A physical protocol ρ is a physical zero-knowledge protocol for property Π with respect to the physical operation set $\mathcal{C}$ if for any $\mathcal{A}$ there exists $\mathcal{S}$ such that for all environments $\mathcal{Z}$ outputting a single bit:*

$$\text{HYBRID}_{\rho,\mathcal{F}_{\mathcal{C}},\mathcal{A},Z} \approx \text{IDEAL}_{\mathcal{F}_{ZK}^{\Pi},\mathcal{F}_{\mathcal{C}},\mathcal{S},\mathcal{Z}}$$

Main differences from standard $\mathcal{F}_{ZK}^{R}$. One difference is to allow the verifier to obtain leakage by overtly cheating; however, $\mathcal{F}_{ZK}^{R}$ could be extended similarly. A more fundamental difference is

[4] Ideally, $\mathcal{C}$ should define a sufficient set of operations such that any action outside this set will either be recognized as malicious or irrelevant to the protocol. This is not a formal mathematical notion, but a physical assumption.

[5] To differentiate *statistical*, *computational*, and *perfect* PhysicalZK, we can easily extend the definition to depend on the type of indistinguishability (statistical/computational/perfect) that the relation $\approx$ describes.

the way $\mathcal{F}_{ZK}^{\Pi}$ verifies Π. $\mathcal{F}_{ZK}^{R}$ requires the prover to submit a witness w along with the input x so that $\mathcal{F}_{ZK}^{R}$ may efficiently verify $(x, w) \in R$. $\mathcal{F}_{ZK}^{R}$ cannot find a witness w on its own since UC requires the trusted party to be computationally efficient. In contrast, $\mathcal{F}_{ZK}^{\Pi}$ verifies $id_X \in \Pi$ on its own, as it only needs the prover to transfer *access permissions*, not secret knowledge.

This difference has significant consequences. $\mathcal{F}_{ZK}^{R}$ cannot be realized in UC without trusted setup because the simulator must straight-line extract a witness from its interaction with the real prover, implying that the real verifier could do so as well. UC-emulation of $\mathcal{F}_{ZK}^{\Pi}$ does not require extraction. Standard ZK proofs in UC are *zero-knowledge proofs-of-knowledge* (ZKPoK), whereas $\mathcal{F}_{ZK}^{\Pi}$ is not. Thus, although UC protocols for $\mathcal{F}_{ZK}^{\Pi}$ may rely on physical assumptions, they do not fundamentally require trusted setup assumptions.

$\mathcal{F}_{\mathcal{C}}$-hybrid protocol UC-emulation of $\mathcal{F}_{ZK}^{\Pi}$

We can assume that $\mathcal{A}$ acts as a proxy for the environment $\mathcal{Z}$ [4].

- When $\mathcal{A}$ corrupts Verifier, $\mathcal{Z}$ either sees a successful run of $\rho^{\mathcal{F}_{\mathcal{C}}/\mathcal{C}}$, or receives $(0, id_X, \Pi)$. Since $\mathcal{S}$ only receives a receipt $(\Pi(id_X), id_X, \Pi)$, it must simulate the hybrid world proof, invoking an instance of $\mathcal{F}_{\mathcal{C}}$, and dummy parties P and V. $\mathcal{S}$ plays the role of prover (P), and uses $\mathcal{Z}$'s messages to play the verifier (V), whose messages $\mathcal{S}$ forwards to $\mathcal{Z}$. If failure occurs or cheating is detected in the simulation, $\mathcal{S}$ sends either fail or cheat to $\mathcal{F}_{ZK}^{\Pi}$.
- $\mathcal{S}$'s simulation must be straight-line (it cannot rewind $\mathcal{Z}$), but $\mathcal{S}$ can extract any hybrid world "physical commitments" that the dummy party V makes with $\mathcal{F}_{\mathcal{C}}$. ($\mathcal{S}$ omits from its messages to $\mathcal{Z}$ any notification of the command **Force** it internally uses with $\mathcal{F}_{\mathcal{C}}$ to force open the commitments).
- When $\mathcal{A}$ corrupts Prover, $\mathcal{S}$ simulates the hybrid world protocol with $\mathcal{Z}$, but now playing the verifier's role (no secret input needed). If the simulation succeeds, $\mathcal{S}$ forwards $\mathcal{Z}$'s input to $\mathcal{F}_{ZK}^{\Pi}$. If failure or cheating occurs, $\mathcal{S}$ sends either fail or cheat to $\mathcal{F}_{ZK}^{\Pi}$.

Fig. 2. The $\mathcal{F}_{\mathcal{C}}$-hybrid model security proof

4 Neutron radiography

Glaser, Barak, and Goldston [8] were the first to suggest applying zero-knowledge proofs to the problem of authenticating nuclear warheads without revealing sensitive information about their design. One approach to authentication is "template-matching." The inspecting party possesses a template warhead, presumably confirmed to be authentic. The opposing party must prove that each warhead brought to the dismantlement queue is identical (in design) to the template.

Neutron radiography can be used to compare objects. An object is bombarded with neutrons, and the intensity of neutron scattering is measured over a range of angles. Glaser et al. suggested using *passive bubble detectors*[6] to physically record the neutron counts at randomly selected angles. The task of comparing the physically recorded counts essentially reduces to Bins and Balls Equality. The GBG protocol for Bins and Balls Equality (see Section 2) only achieves ϵ-*knowledge* with $\epsilon = n/N$ (security is broken with $O(N)$ repetitions). We present a modified protocol that achieves perfect PhysicalZK.

Protocol 4.1 guarantees that the number of balls the verifier eventually counts is uniformly distributed in $[N, 2N)$. Instead of preparing bin pair j containing $r_j \in [0, N)$ balls, the prover prepares a quadruple j of bins: one pair of bins with $r_j \in [0, N)$ balls each, and a second with

[6] A passive bubble detector contains droplets of superheated liquid dispersed throughout a clear gel. When a neutron hits a droplet, it vaporizes the droplet producing a visible air bubble trapped in the gel.

$N + r_j$ balls each. If the number of balls in the prover's original bins is $x < N$, then exactly one of $N + r_j + x$ and $r_j + x$ lies in the interval $[N, 2N)$. Only this bin pair is retained and displayed.

Protocol 4.1: Bins and Balls Quadruples

Input: Two bins X and Y, which both contain x and y balls respectively. The maximum capacity of each bin is N.

1. Prover prepares and seals k "quadruples" of bins $Q_1, ..., Q_k$, where each "quadruple" Q_i consists of two pairs of bins, $pair_{i,0}$ and $pair_{i,1}$. Each bin has capacity at least $2N$. For all $1 \leq i \leq k$, Prover randomly selects uniformly distributed values $r_i \in [0, N)$, and prepares each Q_i such that each bin in $pair_{i,0}$ contains r_i balls, and each bin in $pair_{i,1}$ contains $r_i + N$ balls.
2. Verifier randomly selects j uniformly distributed in $[1, k]$, and requests to view all $Q_{i \neq j}$. Prover reveals all quadruples $Q_{i \neq j}$, and Verifier checks that these quadruples were initialized correctly.
3. Prover selects a final pair out of Q_j: if $x + r_j \geq N$, then Prover chooses $pair_{j,0}$, and if $x + r_j < N$, then Prover chooses $pair_{j,1}$. Prover destroys the other pair.
4. Verifier labels the bins in the remaining pair as "bin_x" and "bin_y" (he can do this randomly to add a 1/2 factor to the soundness error).
5. Prover pours the contents of X into bin_x, and the contents of Y into bin_y. Prover reveals the contents of bin_x and bin_y, and Verifier accepts the proof if and only if the two bins contain the same number of balls.

Soundness: The soundness error is at most $\frac{1}{2k}$. The verifier would accept a false claim (when $x \neq y$) only if it selects a quadruple $j \in [k]$ and labeling of the bins in the final pair so that $x + r_j = y + r'_j$, where r_j and r'_j are the initializations of the bins labelled bin_x and bin_y respectively. If more than one quadruple contains an incorrect initialization such that $r_j \neq r'_j$, then the verifier catches the prover. If one labeling results in $x + r_j = y + r'_j$, then the opposite labeling does not. Therefore, this event occurs with probability at most $\frac{1}{2k}$.

Perfect Zero-Knowledge: We show that the distribution of balls in the final pair of bins (bin_x and bin_y) is the uniform distribution over $[N, 2N)$. Fix an arbitrary input value $0 \leq a < N$ for the number of balls that bins X and Y each hold. Let Z denote the number of balls in bin_x and bin_y at the end of the protocol. $Z = r_j + a$ when $r_j + a \geq N$, and $Z = r_j + a + N$ when $r_j + a < N$. Consider $t \in [N, 2N)$. If $t \in [N, N + a)$, then $Pr[Z = t] = Pr[r_j = t - a] = 1/N$. Otherwise, if $t \in [N + a, 2N)$, then $Pr[Z = t] = Pr[r_j = t - a - N] = 1/N$.

The complete formal proof is very similar to the proof in Appendix A.2. Roughly, since the distribution in the final pair is uniform and independent of the input, the simulator can run the protocol on an empty input.

4.1 From bins and balls to neutron bombardment

We adapt Protocol 4.1 to the problem of proving object radiograph equivalence. Neutron detectors are placed at a finite number of angles around each object, and a neutron source is fired at both objects for the same duration of time. A measurement device is used to measure the counts of neutrons that each detector has physically recorded.

Measurement devices. The parties mutually possess a neutron source with a known flow rate, and physical neutron detectors. Each party has its own measurement device D for obtaining the physically recorded neutron count of any neutron detector. In the hybrid world, D is modeled as an ideal functionality $\mathcal{F}_D$. When given the input id_X corresponding to an object X, $\mathcal{F}_D$ records a measurement value, and outputs a function of the measured value.

Operation Init(d, r). This initializes the given neutron detector d to the integer value r. We assume that the prover and verifier can perform this operation without the other party knowing the value r. [7]

Prover types. We consider two types of provers. Prover Type I has prior knowledge of the exact neutron counts x_θ and y_θ at any angle θ that the verifier chooses to examine, and Prover Type II does not possess this knowledge.

Drawbacks. A Type I prover is required to know the values of x and y for any angle θ. A Type II prover is allowed to re-handle the detectors after the neutron collection, possibly giving her the opportunity to dishonestly meddle with the results. In Appendix B we include a different zero-knowledge protocol for ORE that avoids both of these issues. The protocol uses a measurement device that outputs neutron counts modulo N.

Protocol 4.2: Perfect PhysicalZK protocol for Object Radiograph Equivalence (ORE)

Input: Two objects X and Y with equal ORE's, denoted $X \sim Y$.

1. Prover: Prepare k "quadruples" $Q_1, ..., Q_k$ of neutron detectors (as in Protocol 4.1), selecting random values $r_i \in [0, N)$ for $1 \leq i \leq k$, and using the **Init**(d,r_i) operation on each.
2. Verifier: For each Q_i, $1 \leq i \leq k$, randomly select $b_i \leftarrow \{0, 1\}$.
   - If $b_i = 0$: examine the detectors in Q_i, and check that the neutron count initialization is valid. `Fail` if invalid.
   - If $b_i = 1$: run collection test on Q_i.

Collection Test on Q_i:

1. Verifier: For each pair of the quadruple Q_i, randomly choose one detector to label d_x^i, and label the other d_y^i. Select a random angle θ, and send this to the prover.
2. Prover: Run the neutron source on X and Y, collecting at the angle θ, using detectors labeled d_x^i for X and d_y^i for Y.
   - *Type I Prover*: Choose the unique detector pair that has a count in the range $[N, 2N)$, and discard the other pair. Hand this pair to the verifier.
   - *Type II Prover*: Examine the contents of both detector pairs, and proceed as a Type I prover.
3. Verifier: Check that the detectors received from the prover have equal neutron counts. `Fail` if the counts are not equal.

Tolerance δ. The verifier could accept if and only if $|x_\theta - y_\theta| < \delta$. Two changes are necessary. First, the prover should choose which pair to discard based on the lower of the two values x and y. Unfortunately, the difference $|x - y|$ is still revealed. Second, the verifier must ensure that $N \geq max_\theta\{|x_\theta - y_\theta| + \delta\}$. Otherwise the prover could fool the verifier into accepting that $|x_\theta - y_\theta| < \delta$ when $|x_\theta - y_\theta| > N - \delta$. Verifier can incorporate checking the size of N into the cut-and-choose protocol, but needs to know some loose upper bound on $x_\theta - y_\theta$.

Soundness and Completeness: Protocol 4.2 has perfect completeness and soundness error at most $(\frac{1+\beta}{2})^k$, where $\beta < 1$ is the probability that $x_\theta = y_\theta$ at a uniformly distributed angle θ (when $X \not\sim Y$). Suppose Prover cheats on c out of k detector quadruples. The probability that Verifier doesn't check any of the c bad quadruples is 2^{-c}. The probability Prover passes on all the $k - c$ good quadruples is $\frac{1}{2} + \frac{1}{2} \cdot \beta$ because it passes always if Verifier chooses to check and with probability β if Verifier chooses to run a collection test. By independence, the soundness error is

[7] The appropriate time to run the neutron source at a detector during initialization is calculated from the flow rate of the source. However, the initialization value should be hidden from the other party. The initialization can be done privately, or using a concealed on/off switch on the neutron source.

thus $(\frac{1}{2})^c \cdot (\frac{1}{2} + \frac{\beta}{2})^{k-c}$. Since $\beta \geq 0$, an optimal (cheating) strategy is to set $c = 0$, giving error at most $(\frac{1}{2} + \frac{\beta}{2})^k$.

Perfect Zero-Knowledge: Follows from the analysis of Protocol 4.1.

5 DNA profiling

In recent years, genetic privacy in DNA profiling has become the subject of wide debate. Privacy issues obstruct criminal investigations, deterring non-guilty suspects from otherwise providing DNA samples, and giving guilty suspects legitimate excuses to refuse testing. We present a zero-knowledge protocol through which a suspect can prove to the police that his DNA profile does not match a crime scene profile. We also sketch an adaptation of the BBQ (Protocol 4.1) primitive to DNA testing. One potential application is a zero-knowledge protocol for parental testing.

STR analysis. DNA profiling uses STR analysis. STR stands for "Short Tandem Repeats," which are short nucleotide sequences that repeat in tandem. In certain locations of the human genome, although all humans posses the same repeating sequence, the exact number of repeat units is highly variable from person to person. The variations of a gene or genetic locus in the human population are called *alleles*. Every individual has two alleles of each gene, one from each parent.

CODIS profiles. In the United States, all forensic laboratories share CODIS (the Combined DNA Index System), which uses 13 specific STR loci to identify individuals. A CODIS DNA profile vector consists of 13 pairs of STR sequence lengths, one pair for each loci.

DNA primers. A genetic profile is generated through STR analysis. PCR (the polymerase chain reaction) is run with oligonucleotide primers to isolate and amplify each STR repeat sequence. Primers determine the specific start and end nucleotides of the sequence to be amplified, and thus control the lengths of the flanking regions that are cut out along with the STR sequences (see Figure 3). We will use the notation $P_{i,j}$ to denote a primer pair that isolates the ith locus STR sequence, and produces a pair of fragments of sizes $m_{i,1} + j$ and $m_{i,2} + j$, when $m_{i,1}$ and $m_{i,2}$ are the sizes of the ith locus alleles.

Electrophoresis. In capillary electrophoresis, the most popular technique for DNA profiling, DNA fragments are fluorescently labelled during PCR, and passed through a capillary tube. Smaller fragments pass faster than larger ones. A laser detects the fragments as they pass by. The length of a fragment is deduced from the time the fragment takes to reach the laser.

δ-CE device. We imagine a slightly modified capillary electrophoresis apparatus in which the laser can only be operated for a limited time window δ, effectively limiting the range of DNA fragment sizes that will be detected.[8]

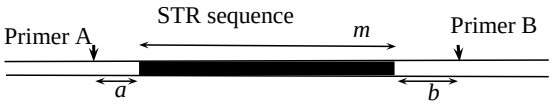

Fig. 3. STR sequence of length m cut out with flanking region $a + b$.

5.1 DNA Inequality

At a basic level, the *police* (POL) will give the *defender* (DEF) one of the two DNA samples at random, and DEF must correctly identify the sample received. In general, DEF is not the sus-

[8] Only the owner of the δ-CE can trust its operations. The party operating the δ-CE can ensure the limited-laser functionality by using a charged capacitor to power the laser so that the laser retains power for at most time δ. The operation can be repeated by recharging the capacitor.

pect, but a public defender forensic team representing the suspect. There are two main challenges in proving zero-knowledge. First, the protocol simulator needs to extract the verifier's challenges without rewinding. Second, we must prevent POL from substituting a third auxiliary DNA sample mid-protocol. (Similarly, in the ZK protocol for *graph non-isomorphism* (GNI), the verifier proves it knows an isomorphism between its challenge graph and one of the two input graphs [11]). Additionally, we rely on the physical assumption that two DNA samples from the same person are always indistinguishable, whereas samples from individuals with distinct profiles are always distinguishable.

Random seals. We require non-forgeable tamper-evident seals. We imagine that tamper-evident seals could be manufactured with a "random" pattern that is uniformly and independently distributed in a sufficiently large domain. A similar random pattern trick was suggested in [26].

Seal covers. Our protocol also requires seal covers that function as physical commitments. The covers should be designed to hide any identifiable markings on the tamper-evident seals, and it should be possible to open the seals without removing their covers.

Protocol 5.1: Zero-knowledge proof for DNA inequality

Preparation: Two test tubes are jointly prepared, one with DNA sample C, and one with S. DEF places identifiable tamper-evident seals on each test tube: one identifies S, and the other identifies C. POL covers the seals.

1. POL conceals the two test tubes, selects one randomly, and hands it back to DEF.
2. DEF checks that the seal has not yet been opened, and then opens the seal without removing its cover. DEF determines the profile of the sample in the test tube, and physically commits to the identity of the sample in the test tube.
3. POL "de-commits" to the challenge test tube by removing the cover on its seal.
4. DEF checks and identifies the uncovered seal to see that it wasn't replaced. If it is not the original seal, then DEF terminates the protocol. Otherwise, DEF opens its commitment from Step 2.
5. POL checks that DEF committed to the correct identity of the challenge sample.

Completeness and Soundness. As DEF is able to differentiate between two distinct DNA samples, the protocol has perfect completeness. If the DNA samples are the same, DEF guesses the challenge sample correctly with probability 1/2. (The error is exponentially reduced by repetition).

Hybrid model. The compiler holds tuples for each DNA sample input. The value attribute of each tuple is the DNA profile vector. The *random seal* operations queries an RO for a value r, tags r to the tuple, locks read/write access, and outputs r to the calling party. The *scramble* functionality swaps the *id* attributes of two tuples with probability 1/2. The environment machine initializes and locks tuples for each DNA input sample. This emulates the "joint preparation" for the physical reason that no information is revealed to either party until they use analysis tools, such as electrophoresis, to examine the products of the PCR preparation.

Zero-Knowledge. The ideal functionality DNA Inequality is denoted $\mathcal{F}_{ZK}^{DI}$. The job of the simulator $\mathcal{S}$ in the case that $\mathcal{A}$ corrupts the hybrid world prover (DEF) was handled generically in Section 3, Figure 2. The case that $\mathcal{A}$ corrupts the hybrid world verifier (POL) is more interesting. First, $\mathcal{S}$ learns from $\mathcal{F}_{ZK}^{DI}$ if DEF passes or fails the ideal execution on the environment Z's input. Next, $\mathcal{S}$ simulates the hybrid model protocol, playing the role of DEF while Z uses $\mathcal{A}$ to control POL. Recall that $\mathcal{S}$ can straight-line extract physical commitments in the hybrid world (see Figure 2). Thus, $\mathcal{S}$ always knows the identity of POL's challenge sample, or that POL is cheating. $\mathcal{S}$ sets DEF's commitment in Step 2 to the correct identity of the challenge if DEF should pass, and the incorrect if DEF should fail. $\mathcal{S}$ only de-commits in Step 4 if POL did not cheat. If POL did not cheat, both $\mathcal{S}$

and the hybrid world DEF supply identical responses to Z, namely the identity of POL's challenge. If POL did cheat on its challenge, then $\mathcal{S}$ terminates its simulation. The hybrid world DEF will also terminate unless it fails to catch POL cheating, which only occurs with negligible probability (POL must guess the secret RO tag in order to fool DEF).

Testing a village. There are cases where entire villages have been tested to see if the DNA profile of anyone in the population matches the crime scene DNA profile. Protocol 5.1 can be naturally extended for proving that a DNA profile does not exist in a population. Consider a population of 400 people. The verifier DEF receives 401 DNA samples, one from every individual in the population, plus a crime scene sample, all delivered in a set of 401 identical, covered, sealed, and randomly permuted test-tubes. DEF is required to find the crime scene sample C and hand it back to POL. If another individual has the same profile as C, then DEF fails with probability at least $\frac{1}{2}$.

5.2 Parental testing

To prove a parent-child relationship using DNA profiling, it is necessary to show that DNA samples from the parent and child share at least one allele in each STR locus. We construct a zero-knowledge protocol for this task using an analogous technique to the Bins and Balls Quadruples (BBQ) scheme from Section 4.

Recall that a *primer* $P_{i,j}$ is used to cut out the ith locus STR alleles with flanking regions of total length j. Performing STR analysis with randomized primers P_{i,r_i} for r_i uniformly distributed in $[0, N)$ is analogous to adding a random number of balls to a bin. The quadruples of bins in BBQ translate to quadruples of test tubes running reactions with randomized primers. However, a technical caveat arises: each ith locus actually contains a pair of alleles that will be amplified with the same P_{i,r_i}, producing a pair of fragment lengths whose joint distribution is *not* necessarily uniform!

In the special case of paternity testing, this issue can be easily avoided by choosing to compare STR regions on the Y chromosome, which is uniquely passed from father to son. In more general circumstances, the prover can choose to reveal only one allele from each locus (using δ-CE), which is sufficient for showing that the two DNA samples share at least one allele in each STR locus.

6 Public coin and public observation proofs

A *private coin protocol* is one in which the verifier's random bits ("coin flips") must be kept private during the protocol. In contrast, the verifier's messages in a *public coin protocol* only consist of the outcomes of its coin flips. Public-coin physical protocols are *publicly observable* in the sense that the verifier can sit behind a glass screen throughout the protocol's execution, sending messages to the prover, and observing the prover's physical operations. Thus, unlike general physical zero-knowledge protocols, public observation physical zero-knowledge protocols do not rely on tamper-evident functionalities or (as heavily) on a covert threat model. While all public-coin protocols are publicly observable, not all publicly observable protocols need to be public-coin. For instance, the protocol may involve private-coin computational subprotocols.

In this section, we present an example of a public observation protocol for a special case of *DNA Inequality* in which the suspect's DNA S should pass if in at least one of the 13 CODIS loci it has an allele that is not present in the crime scene DNA C (notated $S \not\subset C$). The protocol becomes a perfectly complete test for DNA Inequality when only homozygotic gene regions are compared (e.g. X or Y chromosomes in males). The construction of the protocol involves a reduction to standard cryptography, using bit-commitment and generic ZK proofs for NP statements. Our protocol takes

advantage of the fact that when $S \not\subset C$, the total number of distinct gene alleles in $S \cup C$ increases. Our protocol is closely related to the well-known public-coin protocol for GNI, making use of a classical set lower-bound protocol [17, 10].

In the digital setting, public-coin $\mathcal{ZK}$ equals private-coin $\mathcal{ZK}$ [27, 13, 14, 28]. In contrast, we don't know of a general method for converting any physical zero-knowledge protocol into a public observation zero-knowledge protocol. The known constructions of public-coin $\mathcal{ZK}$ proofs from private-coin $\mathcal{ZK}$ proofs involve simulating the private-coin verifier and applying universal hash functions to its messages. We do not know of any general analogous method for hashing physical messages. Furthermore, the public-coin verifier must be able to check set containment in the private-coin verifier's messages. In the physical setting, it is unclear whether the public-coin verifier can always assess the physical content of the private-coin verifier's messages, particularly when they involve physical concealment.f

Protocol 6.1: Public Observation ZK Protocol for DNA Inequality

- **Allele vector sets.** $V(C)$ is the set of *distinct* vectors of length 13 that can be formed by choosing one allele from each STR locus of C. $1 \leq |V(C)| \leq 2^{13}$. Similarly, we define $V(S \cup C)$, where each allele can be chosen from either S or C. If S introduces a new allele in at least one locus, then $|V(S \cup C)| \geq \frac{3}{2}|V(C)|$, and otherwise, $|V(S \cup C)| = |V(C)|$.
- **Equipment.** DEF has a *mod δ-CE device*, which is a δ-CE device that displays the lengths of DNA fragments modulo N, where N is a power of 2 greater than the longest possible fragment that will be measured. DEF also has access to a collection of DNA primers $\{P_{i,j}\}$.
- **Parameters ℓ and m.** Choose the smallest integers ℓ and m that satisfy the following conditions: $(\frac{3}{2})^\ell \geq 2m$ and $2^{m-\frac{1}{2}} \leq (\frac{3}{2})^\ell \cdot |V(C)|^\ell \leq 2^m$.[9]
- **Allele length vector set Z.** $Z \subset (\mathbb{Z}_N)^{13}$ is the set of possible vectors whose ith component is the length mod N of an allele taken from the ith locus of either S or C. It is the set of vectors of allele lengths mod N corresponding to $V(S \cup C)$.
- **Randomized allele length vector set $\mathbf{r}(Z^\ell)$.** Let $\mathbf{r}$ be a random uniformly distributed vector in $(\mathbb{Z}_N)^{13\ell}$. Define $\mathbf{r}(Z^\ell) = \{\mathbf{r} + \mathbf{z} \mid \mathbf{z} \in Z^\ell\}$ where addition is over $(\mathbb{Z}_N)^{13\ell}$.
- **Hash function family.** Set $k = log(N) \cdot 13\ell$, and choose a canonical encoding of $(\mathbb{Z}_N)^{13\ell}$ in $GF(2^k)$. We will use a family $\mathcal{H}_{k,m} := \{h_{a,b}\}$ of universal hash functions from $GF(2^k) \to GF(2^m)$ where $a, b \in GF(2^k)$, $a \neq 0$, and $h_{a,b}$ maps $x \mapsto ax + b$ and truncates the last $k - m$ bits.
- **Hash function shift.** For any vector $\mathbf{r} \in (\mathbb{Z}_N)^{13\ell}$ and $h \in \mathcal{H}_{k,m}$, define $\mathbf{r}(h) = h_{a,b-r\cdot a}$, where r is the encoding of $\mathbf{r}$ in $GF(2^k)$. Note that $\mathbf{r}(h)(\mathbf{r} + \mathbf{z}) = h(\mathbf{z})$.
- **Preparing $\mathbf{r}(Z^\ell)$.** DEF prepares ℓ sets of test tubes $T_1, ..., T_\ell$. Each T_i consists of test tubes $\{C_{i,j}\}_{j\in[13]}$ containing C and $\{S_{i,j}\}_{j\in[13]}$ containing S. For all (i, j), DEF selects an independent random value $r_{i,j}$ uniformly distributed in $[0, N)$, and then runs PCR on $C_{i,j}$ and $S_{i,j}$ with the primer pair $P_{j,r_{i,j}}$.[10] To display a vector $\mathbf{z} \in \mathbf{r}(Z^\ell)$ to POL, DEF chooses from every T_i the appropriate set of 13 test tubes containing the target fragments contained in $\mathbf{z}$, and then chooses time windows $t_{i,1}, ..., t_{i,13}$ to run the *mod δ*-CE device on each test tube in order to only detect the target fragments.

Perfect Completeness: We use the fact that for any set $\mathcal{A} \subseteq \{0, 1\}^m$ of size $|\mathcal{A}| \geq 2^{m-\frac{1}{2}}$, there there exists an (expected polynomial time computable) set of m hash functions $h_1, ..., h_m \in \mathcal{H}_{k,m}$

[9] Let $K = |V(C)|$. Note that $K = 2^r$ for $r \leq 13$. Choose the smallest integer x such that $(\frac{3}{2})^x \geq 2x \cdot log(\frac{3}{2}K) + 1$. One can verify that $x \leq 15$. Now find the smallest $\ell \geq x$ such that the fractional part of $\ell \cdot log(\frac{3}{2}K)$ is in $[\frac{1}{2}, 1) \cup \{0\}$. This will hold for either $x, x+1$, or $x+2$. Finally, set m to be the unique integer such that $m - \frac{1}{2} \leq \ell \cdot log(\frac{3}{2}K) \leq m$. Now ℓ and m satisfy $(\frac{3}{2})^\ell \geq 2\ell \cdot log(\frac{3}{2}K) + 1 \geq 2m$ and $2^{m-\frac{1}{2}} \leq (\frac{3}{2}K)^\ell \leq 2^m$.

[10] POL observes that the same primer is applied to $C_{i,j}$ and $S_{i,j}$.

Protocol 6.1 - Public observation ZK for DNA Inequality

POL observes *DEF* throughout the rounds. We assume that *DEF* cannot change the behavior of the *mod δ-CE device* while under observation. Let $(\texttt{com}, \texttt{dec})$ denote a commitment scheme.

1. DEF: Choose a random uniformly distributed $\mathbf{r} \in (\mathbb{Z}_N)^{13\ell}$ and prepare $\mathbf{r}(Z^\ell)$. Find a set of m hash functions $H = \{h_i\}_{i=1}^m \subset \mathcal{H}$ so that $\bigcup_{i=1}^m h_i(Z^\ell) = \{0,1\}^m$. Compute commitments to the hash functions $\mathbf{r}(h_i)$ for each i, denoted $Com_H = \{\texttt{com}(\mathbf{r}(h_1)), ..., \texttt{com}(\mathbf{r}(h_m))\}$. Send Com_H to POL.
2. POL: Pick a uniformly distributed $y \in \{0,1\}^m$.
3. DEF: Find an $h_i \in H$ and $\mathbf{z} \in Z^\ell$ such that $h_i(\mathbf{z}) = y$. Display the allele vector $\mathbf{v} = \mathbf{z} + \mathbf{r}$ from the set $\mathbf{r}(Z^\ell)$.
4. DEF and POL: Execute a UC-secure ZK proof of the NP statement "there exists $x \in Com_H$ such that $\texttt{dec}(x) = h$ and $h(\mathbf{v}) = y$."

such that $\cup_i h_i(\mathcal{A}) = \{0,1\}^m$. (We include a proof of this fact using the Probabilistic Method in the full version of this paper).When S has at least one distinct allele from C, then $|Z^\ell| = |V(S \cup C)^\ell| \geq (\frac{3}{2})^\ell \cdot |V(C)^\ell| \geq 2^{m-\frac{1}{2}}$. Given any $\mathbf{z}$ there is at least one h_i among the m preselected functions that satisfies $h_i(\mathbf{z}) = y$. Since $\mathbf{r}(h_i)(\mathbf{z} + \mathbf{r}) = h_i(\mathbf{z}) = y$, there exists $h \in Com_{\mathcal{H}}$ such that $h(\mathbf{v}) = y$.

Soundness error 1/2: When S does not contain any distinct alleles from C, then $|Z| = |V(C)|$. For any set of m hash functions $\{h_i\}_{i=1}^m$, the size of $\cup_{i=1}^m h_i(Z^\ell)$ is at most $m \cdot |V(C)|^\ell \leq m \cdot (\frac{3}{2})^{-\ell} \cdot 2^m \leq \frac{m}{2m} \cdot 2^m = 2^{m-1}$. Thus, the probability that a uniformly selected target $y \in \{0,1\}^m$ is in the image $\cup_i h_i(Z^\ell)$ is at most $\frac{1}{2}$. Note that this soundness error bound is independent of the prover's *mod δ-CE* device behavior, whether randomized or deterministic. The device output ensemble consists of at most $2^{m-1}/m$ distinct random variables over $\{0,1\}^m$, i.e. one variable X_z for each input configuration $z \in Z^\ell$. Given the uniformly selected target y, the probability that $h_i(X_z) = y$ for some i and z is (by a union bound) at most $1/2$.

Zero-Knowledge: We separately analyze the physical (DEF reveals $\mathbf{v}$ to POL) and computational (DEF proves there exists $x \in Com_{\mathcal{H}}$ such that $\texttt{dec}(x) = h$ and $h(\mathbf{v}) = y$) stages of the protocol. In the physical stage, POL only sees the indices of $\mathbf{v} = \mathbf{z} + \mathbf{r}$, and since $\mathbf{r}$ is uniformly distributed independent of $\mathbf{z}$, $\mathbf{v}$ is also uniformly distributed. In the computational stage, DEF and POL execute a ZK protocol that is UC-secure under either physical assumptions or computational setup assumptions. The composed protocol securely realizes PhysicalZK by the universal composition theorem.

7 Conclusion and Future Directions

The need for privacy pervades not only the world of digital information, but physical information as well. Privacy in nuclear disengagement treaties and DNA profiling are just two examples of pertinent real world problems requiring inherently physical rather than digital solutions, and motivate the importance of developing a better theoretical foundation for physical cryptography.

A starting point is the rigorous analysis of protocols. The approach presented in this work separates the logical and physical components of a protocol using the language of modern cryptography, formally reducing mathematical claims of security and correctness to the underlying physical assumptions theory cannot address. Beyond that, there are structural questions: are there ZK proofs for every physical property, or secure computation protocols for every physical task? We noted parallels between public observation physical ZK protocols and public coin digital ZK protocols. Can any physical ZK protocol be made into a publicly observable one?

In the physical world, opposite to the digital, general theories and impossibility results seem difficult or impossible to achieve with only the tools of mathematics. Nonetheless, investigating

general theories is an interesting direction for future work, perhaps beginning with restricted classes of physical operations. An orthogonal direction is to explore other models. In subsequent work, we show several techniques for solving generic physical tasks using a *disposable circuits model* in which digital information can be destroyed.

Acknowledgements

We are grateful to Tamir Biezuner, who helped us navigate the procedures involved in DNA profiling, and in particular, helped us adapt the *bins and balls quadruples* technique to DNA profiling using randomized primer selection. We also thank the anonymous reviewers for their helpful suggestions and comments.

References

1. Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In S. P. Vadhan, editor, *2007*, volume 4392 of *LNCS*, pages 137–156. Springer, Feb. 2007.
2. S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In T. Helleseth, editor, *Eurocrypt '93*, volume 765 of *LNCS*, pages 344–359. Springer, May 23–27, 1993.
3. C. Brzuska, M. Fischlin, H. Schröder, and S. Katzenbeisser. Physically uncloneable functions in the universal composition framework. In P. Rogaway, editor, *Crypto 2011*, volume 6841 of *LNCS*, pages 51–70. Springer, Aug. 14–18, 2011.
4. 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.
5. R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *Crypto 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Aug. 19–23, 2001.
6. R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In E. Biham, editor, *Eurocrypt 2003*, volume 2656 of *LNCS*, pages 68–86. Springer, May 4–8, 2003.
7. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In M. Naor, editor, *2004*, volume 2951 of *LNCS*, pages 258–277. Springer, Feb. 2004.
8. A. Glaser, B. Barak, and R. J. Goldston. A new approach to nuclear warhead verification using a zero-knowledge protocol. In *In 53rd Annual INMM meeting. Institute of Nuclear Materials Management*, 2012.
9. O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
10. O. Goldreich, Y. Mansour, and M. Sipser. Interactive proof systems: Provers that never fail and random selection. In *28th IEEE Symposium on Foundations of Computer Science*, pages 449–461, 1987.
11. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.
12. O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
13. O. Goldreich, A. Sahai, and S. Vadhan. Honest verifier statistical zero-knowledge equals general statistical zero-knowledge. In *30th Annual ACM Symposium on Theory of Computing*, pages 399–408, May 1998.
14. O. Goldreich and S. Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of szk. In *14th Annual IEEE Conference on Computational Complexity*, pages 54–73, May 1999.
15. 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.
16. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
17. S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, pages 59–68, 1986.
18. 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.
19. R. Gradwohl, M. Naor, B. Pinkas, and G. N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. *Theory Comput. Syst.*, 44(2):245–268, 2009.

20. C. Hazay and Y. Lindell. Constructions of truly practical secure protocols using standardsmartcards. In *ACM CCS '08*, pages 491–500. ACM Press, 2008.
21. D. Hofheinz, J. Müller-Quade, and D. Unruh. Universally composable zero-knowledge arguments and commitments from signature cards. In *5th Central European Conference on Cryptology MoraviaCrypt 2005*, 2005.
22. 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.
23. T. Moran and M. Naor. Basing cryptographic protocols on tamper-evident seals. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 285–297. Springer, July 2005.
24. T. Moran and M. Naor. Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In S. Vaudenay, editor, *Eurocrypt 2006*, volume 4004 of *LNCS*, pages 88–108. Springer, May 28 – June 1, 2006.
25. 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.
26. 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.
27. T. Okamoto. On relationships between statistical zero-knowledge proofs. In *28th ACM STOC*, pages 649–658. ACM Press, May 1996.
28. S. P. Vadhan. An unconditional study of computational zero knowledge. In *SIAM Journal on Computing*, pages 176–185, 2004.

A Full Security Proof Example: GBG ϵ-Knowledge

The basic Bins and Balls Equality (BBE) protocol of [8] is described in Section 2. The protocol does not actually achieve PhysicalZK, but it does achieve the closely related notion of ϵ-knowledge. Here $\epsilon = n/N$, where n is the capacity of the input bins, and N is the range of the random number of balls added.

Definition 2. *A protocol ρ UC ϵ-emulates an ideal functionality $\mathcal{T}$ if for any $\mathcal{A}$ there exists $\mathcal{S}$ such that for all environments $\mathcal{Z}$ outputting a single bit: $\Delta(\text{REAL}_{\rho,\mathcal{A},Z}, \text{IDEAL}_{\mathcal{T},\mathcal{S},\mathcal{Z}}) \leq \epsilon$, where Δ denotes statistical variation distance. Equivalently, $|Pr[\text{REAL}_{\rho,\mathcal{A},Z} = 1] - Pr[\text{IDEAL}_{\mathcal{T},\mathcal{S},\mathcal{Z}} = 1]| \leq \epsilon$*

A physical protocol π is a physical ϵ-knowledge protocol for property Π with respect to the physical operation set $\mathcal{C}$ if its $\mathcal{F}_{\mathcal{C}}$-hybrid translation ρ UC ϵ-emulates $\mathcal{F}_{ZK}^{\Pi}$.

Hybrid world modeling of the GBG protocol. The operations of the hybrid model compiler $\mathcal{F}_{\mathsf{BB}}$ are listed in Figure 4. A bin in the hybrid world is represented by a tuple of the form $(id, value, holder, state)$ stored by $\mathcal{F}_{\mathsf{BB}}$. Every operation listed models a real world operation used in the GBG protocol. In the real world protocol, parties can monitor each other by seeing who is holding or operating on a bin. To model this in the hybrid world, $\mathcal{F}_{\mathsf{BB}}$ allows any party to request the $(id, holder, state)$ of any tuple. Additionally, $\mathcal{F}_{\mathsf{BB}}$ notifies all parties of any operation executed, and its status (accept or reject). The hybrid world GBG protocol is described in Figure 5.

Ideal functionality modeling for BBE Recall the definition of the general PhysicalZK ideal functionality $\mathcal{F}_{ZK}^{\Pi}$ in Figure 1 of Section 3. The corresponding ideal functionality of a PhysicalZK protocol for BBE is $\mathcal{F}_{ZK}^{\mathsf{BBE}}$, where BBE denotes the physical property of bins containing equal numbers of balls. The GBG protocol, however, has a non-negligible soundness error of 1/2. While normally the error would be exponentially reduced by repetition, this protocol cannot be repeated on the same physical input. The input is consumed when new balls are added to the input bins. Thus, we will use a modified functionality $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ that allows the prover to cheat with probability at most δ (Figure 6).

$\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ runs with a party Prover, a party $\mathsf{Verifier}$, and an instance of $\mathcal{F}_{\mathsf{BB}}$ specified by a process identifier pid_{BB}. If $(id_X, x, holder, state)$ and $(id_Y, y, holder, state)$ are two tuples stored by $\mathcal{F}_{\mathsf{BB}}$, the relation $(id_X, id_Y) \in \mathsf{BBE}$ holds if and only if $x = y$.

Hybrid model compiler $\mathcal{F}_{\mathsf{BB}}$

For the following descriptions, each command is sent from party P_i to $\mathcal{F}_{\mathsf{BB}}$.

- **Create** ($value$): *Initializes a new bin with value number of balls.* Generate a unique id for the bin, store the tuple $(id, value, P_i, \text{“}open\text{”})$, and send the receipt $(id, value, P_i, \text{“}open\text{”})$ to P_i.
- **Seal** (id): *Seals a bin.* Check in the tuple $(id, value, holder, state)$ that $holder = P_i$ and $state = \text{“}open\text{”}$. If yes, updates the tuple to $(id, value, P_i, \text{“}sealed\text{”})$. Otherwise reject.
- **Break** (id): *Unseals a bin.* Check in the tuple $(id, value, holder, state)$ that $holder = P_i$. If yes, updates the tuple to $(id, value, P_i, \text{“}open\text{”})$. Otherwise reject.
- **Combine** (id_1, id_2): *Combines the contents of two bins.* Check that $holder_1 = holder_2 = P_i$, $state_1 = state_2 = \text{“}open\text{”}$. If yes, then update the tuples to $(id_1, 0, P_i, \text{“}open\text{”})$ and $(id_2, value_1 + value_2, P_i, \text{“}open\text{”})$. Otherwise reject.
- **Open** (id): *Opens and returns the bin contents count.* Check in $(id, value, holder, state)$ that $state = \text{“}open\text{”}$ and $holder = P_i$. If yes, send $(id, value)$ to P_i. Otherwise reject.
- **Send** (m, P_j): *Relays message to P_j.* Relay the message m to P_j. In the physical setting this could be implemented using a number of equivalent forms of communication (speech, writing, etc.)
- **Transfer** (id, P_j): *Transfers bin possession to P_j.* Check in $(id, value, holder, state)$ that $holder = P_i$. If yes, update the tuple to $(id, value, P_j, state)$.
- **Force** (id): *Forcefully opens a bin.* Send the entire tuple $(id, value, holder, state)$ to P_i.

Fig. 4. The $\mathcal{F}_{\mathsf{BB}}$-hybrid model ideal functionalities for BB

Hybrid protocol experiment. $\mathcal{Z}$ activates $\mathcal{F}_{\mathsf{BB}}$ and sets the input to π, the $\mathcal{F}_{\mathsf{BB}}$-hybrid protocol, by initializing the tuples $(id_X, x, \mathsf{Prover}, \text{“}sealed\text{”})$ and $(id_Y, y, \mathsf{Prover}, \text{“}sealed\text{”})$. Next, $\mathcal{Z}$ activates Prover and Verifier, sending them the process identifier pid_{BB} for $\mathcal{F}_{\mathsf{BB}}$. Prover and Verifier execute π as described in Figure 5. $\mathcal{A}$ corrupts one, both, or none of the parties. $\mathcal{A}$ controls the communication of any party it corrupts, but only acts as a proxy for $\mathcal{Z}$. After π has completed, each party sends its output to $\mathcal{Z}$, who's final view of the experiment consists of its communication tape with $\mathcal{A}$ and the outputs of Prover and $\mathsf{Verifier}$. Finally, $\mathcal{Z}$ outputs a single decision bit $\text{HYBRID}_{\pi,\mathcal{F}_{\mathsf{BB}},\mathcal{A},Z}$.

Ideal process experiment. $\mathcal{Z}$ activates $\mathcal{F}_{\mathsf{BB}}$ with process identifier pid_{BB} and initializes the tuples $(id_X, x, \mathsf{Prover}, \text{“}sealed\text{”})$ and $(id_Y, y, \mathsf{Prover}, \text{“}sealed\text{”})$. $\mathcal{Z}$ sends $(id_X, id_Y, pid_{\mathsf{BB}})$ to both Prover and Verifier. Prover will activate $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$, whose PID we denote as $\perp$. First, Prover transfers input tuple access to $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$, executing $\textbf{Transfer}(id_X, \perp)$ and $\textbf{Transfer}(id_Y, \perp)$. Next, it sends pid_{BB} and the input $(id_X, id_Y, pid_{\mathsf{BB}}, \mathsf{Prover}, \mathsf{Verifier})$ to $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$, which in turn proceeds according to Figure 6. As in [4], $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ will also mediate the ideal adversary corruption mechanism. $\mathcal{S}$ corrupts a party P by sending the command (**corrupt** P) to $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$. Since $\mathcal{S}$ cannot corrupt P before $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ is activated, $\mathcal{S}$ cannot modify the environment's input before $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ receives it. Thus, while $\mathcal{S}$ may modify outputs to its corrupted parties, it cannot compromise the ideal process's output to an uncorrupted party. Prover and Verifier both forward the outputs they receive to $\mathcal{Z}$, and $\mathcal{Z}$ outputs a single decision bit $\text{IDEAL}_{\mathcal{F}_{ZK}^{\mathsf{BBE}},\mathcal{F}_{\mathsf{BB}},\mathcal{S},Z}$.

Lemma 1. *The $\mathcal{F}_{\mathsf{BB}}$-hybrid protocol π of Figure 5 UC ϵ-emulates the ideal functionality $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ with $\delta = 1/2$ and $\epsilon = n/N$. In other words, for any $\mathcal{A}$ there exists $\mathcal{S}$ such that for any $\mathcal{Z}$:*

$$|Pr[\text{HYBRID}_{\pi,\mathcal{F}_{\mathsf{BB}},\mathcal{A},Z} = 1] - Pr[\text{IDEAL}_{\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}},\mathcal{F}_{\mathsf{BB}},\mathcal{S},Z} = 1]| \leq \epsilon$$

Proof. We use the following simple fact. Given any distinguisher algorithm D and random variables X and Y with finite range Ω:

$$|Pr[D(X) = 1] - Pr[D(Y) = 1]| \leq \Delta(X, Y)$$

$\mathcal{F}_{\mathsf{BB}}$-hybrid protocol π

Input: $\mathcal{F}_{\mathsf{BB}}$ has the input stored as $(id_X, x, \mathsf{Prover}, \text{“}sealed\text{”})$ and $(id_Y, y, \mathsf{Prover}, \text{“}sealed\text{”})$ where $0 \leq x, y \leq n$.

If $x \neq y$, Prover sends $(\texttt{Reject}, (id_X, id_Y), \mathsf{BBE})$ to Verifier. If $x = y$:

1. Prover: Randomly select r_0, r_1 in [0,N). **Create** (r_0) twice and **Create** (r_1) twice, receiving receipts from $\mathcal{F}_{\mathsf{BB}}$ with four unique id values $B_{0,0}, B_{0,1}, B_{1,0}$, and $B_{1,1}$. For each $B_{i,j}$, **Seal**$(B_{i,j})$ and **Send**$(B_{i,j}, \mathsf{Verifier})$.
2. Verifier: Select a random choice bit $\sigma \in \{0, 1\}$. **Send** $(\sigma, \mathsf{Prover})$.
3. Prover: **Break**$(B_{1-\sigma,0})$, and **Break**$(B_{1-\sigma,1})$. **Transfer** $(B_{1-\sigma,0}, \mathsf{Verifier})$ and **Transfer** $(B_{1-\sigma,1}, \mathsf{Verifier})$.
4. Verifier: **Open** $(B_{1-\sigma,0})$ and **Open** $(B_{1-\sigma,1})$, receiving from $\mathcal{F}_{\mathsf{BB}}$ two receipts $(B_{1-\sigma,0}, v)$ and $(B_{1-\sigma,1}, v')$. Check that $v = v'$. If $v \neq v'$, output $(\textbf{Cheater}, \mathsf{Prover})$.
5. Prover: **Combine** $(id_X, B_{\sigma,0})$, and **Combine**$(id_Y, B_{\sigma,1})$. **Transfer** $(B_{\sigma,0}, \mathsf{Verifier})$ and **Transfer** $(B_{\sigma,1}, \mathsf{Verifier})$.
6. Verifier: **Open** $(B_{\sigma,0})$ and **Open** $(B_{\sigma,1})$, receiving $(B_{\sigma,0}, v)$ and $(B_{\sigma,1}, v')$ from $\mathcal{F}_{\mathsf{BB}}$. Check that $v = v'$. If yes, output $(\texttt{Accept}, (id_X, id_Y), \mathsf{BBE})$. Otherwise, output $(\texttt{Reject}, (id_X, id_Y), \mathsf{BBE})$.

If the protocol prematurely fails due to an invalid message or operation, both parties output `Failed`. If Prover ever receives a receipt $(\textbf{Force}, \mathsf{Verifier}, \mathsf{accept})$ from $\mathcal{F}_{BB}$, it outputs $(\textbf{Cheater}, \mathsf{Verifier})$.

Fig. 5. The $\mathcal{F}_{\mathsf{BB}}$-hybrid model protocol for BBE

Functionality $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$

- Upon receiving $(id_X, id_Y, pid, \mathsf{Prover}, \mathsf{Verifier})$ from the party Prover, query the instance of $\mathcal{F}_{\mathsf{BB}}$ specified by pid to obtain the tuples $(id_X, x, \perp, \text{“}sealed\text{”})$ and $(id_Y, y, \perp, \text{“}sealed\text{”})$. If $x = y$, send $(\texttt{Accept}, (id_X, id_Y), \mathsf{BBE})$ to the party Verifier. If $x \neq y$, send $(\texttt{Reject}, (id_X, id_Y), \mathsf{BBE})$.
- Upon receiving the instruction $(\textbf{Cheat}, \mu)$ from Prover for $0 \leq \mu \leq \delta$, with probability μ send $(\texttt{Accept}, (id_X, id_Y), \mathsf{BBE})$ to Verifier, and otherwise send $(\textbf{Cheater}, \mathsf{Prover})$. Upon receiving the instruction **Cheat** from Verifier, send $(\textbf{Cheater}, \mathsf{Verifier})$ to Prover, and $(\textbf{Cheater}, x, y)$ to Verifier.
- If the queries to $\mathcal{F}_{\mathsf{BB}}$ are unsuccessful, the initial input is invalid, or upon receiving the instruction **Fail** from either party, send `Failed` to both parties.

Fig. 6. Ideal functionality for a ZK proof of BBE with soundness error δ

We will show that for all $\mathcal{A}$ there exists $\mathcal{S}$ such that for any environment $\mathcal{Z}$, the environment's respective views in the hybrid protocol experiment with $\mathcal{A}$ and the ideal process experiment with $\mathcal{S}$ are ϵ-close in statistical distance. We consider separately the four cases in which $\mathcal{A}$ corrupts the Prover, the Verifier, both parties, or neither parties. (We continue to write $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$, but it should be understood that $\delta = 1/2$).

$\mathcal{A}$ corrupts Prover. $\mathcal{S}$ obtains (id_X, id_Y, pid_{BB}) from $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$. $\mathcal{S}$ runs a separate instance of $\mathcal{F}_{\mathrm{BB}}$, and simulates the hybrid protocol π using empty entries for id_X and id_Y, and two dummy parties P (for Prover) and V (for Verifier). $\mathcal{S}$ plays P using the messages coming from $\mathcal{Z}$. $\mathcal{S}$ sends back to $\mathcal{Z}$ any receipts that P receives from $\mathcal{F}_{\mathrm{BB}}$. However, $\mathcal{S}$ does not send P's output from the simulation $\mathcal{Z}$. Instead, it sends one of the following messages to $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$:

- If either P or V output `Failed`, then $\mathcal{S}$ sends **Fail**.
- If P cheated in the initialization of only one pair, then $\mathcal{S}$ sends $(\textbf{Cheat}, 1/2)$.
- If P cheated in the initialization of both pairs, then $\mathcal{S}$ sends $(\textbf{Cheat}, 0)$.
- If none of the above apply, then $\mathcal{S}$ sends $(id_X, id_Y, pid_{\mathsf{BB}}, \mathsf{Prover}, \mathsf{Verifier})$.

Verifier writes the output received from $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$ to its local output tape. $\mathcal{S}$ receives Prover's output from $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$, and writes it to Prover's output tape. $\mathcal{Z}$'s view is identical to its view in the hybrid protocol experiment with $\mathcal{A}$ corrupting Prover.

$\mathcal{A}$ **corrupts Verifier.** $\mathcal{S}$ obtains $\mathcal{Z}$'s input (id_X, id_Y, pid_{BB}), and receives output from $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$: either $(\texttt{Accept}, (id_X, id_Y), \mathsf{BBE})$ in the case that $x = y$, or $(\texttt{Reject}, (id_X, id_Y), \mathsf{BBE})$ in the case that $x \neq y$. If `Reject`, then $\mathcal{S}$ writes the output to Verifier's output tape. If `Accept`, then $\mathcal{S}$ must simulate $\mathcal{A}$'s view of the hybrid protocol π.

$\mathcal{S}$ runs a separate instance of $\mathcal{F}_{\mathrm{BB}}$, creating empty entries for id_X and id_Y, and dummy parties P and V. $\mathcal{S}$ uses messages coming from $\mathcal{Z}$ to play V, and forwards any receipts that V receives to $\mathcal{Z}$. The only possible receipts V receives that could be *statistically different* in this simulated π and the hybrid experiment π are the receipts $(B_{\sigma,0}, v_\pi)$ and $(B_{\sigma,1}, v'_\pi)$ obtained in Step 6. Since the output is `Accept`, we know that $v_\pi = v'_\pi$. v_π is uniformly distributed in $[x, x+N)$, and the simulation output v_{sim} is uniformly distributed in $[0, N)$. The statistical difference is $\Delta(v_\pi, v_{sim}) = x/N \leq n/N = \epsilon$.

Next, $\mathcal{S}$ determines what message to send $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$. If P outputs **Cheater**, it sends **Cheat** to $\mathcal{F}_{ZK,\delta}^{\mathsf{BBE}}$, receives $(\textbf{Cheater}, x, y)$ as output, and writes to Verifier's output tape whatever $\mathcal{A}$ would. If P outputs `Failed`, $\mathcal{S}$ sends **Fail**, and writes `Failed` to Verifier's output tape. Otherwise, $\mathcal{S}$ does not send anything, and simply writes $(\texttt{Accept}, (id_X, id_Y), \mathsf{BBE})$ to Verifier's output tape.

The outputs are identical to the outputs in the hybrid protocol experiment given the same inputs. Therefore, the statistical difference between $\mathcal{Z}$'s views of the hybrid protocol experiment and ideal process experiment on the same inputs is precisely the statistical difference in its communication with $\mathcal{A}$ and $\mathcal{S}$ during π and the simulated π, which is at most $\epsilon = n/N$.

$\mathcal{A}$ **corrupts both or neither parties.** If both, then $\mathcal{S}$ also corrupts both. Since there are no secrets kept from $\mathcal{S}$, it can run the hybrid world experiment without help. If neither, $\mathcal{S}$ does nothing. For the same inputs, the outputs of Prover and Verifier are identical in the hybrid protocol experiment and the ideal process experiment when there is no corruption.

By Lemma 1, the GBG protocol is a physical ϵ-knowledge protocol for BBE with respect to the operation set BB.

B ORE with a mod-counter

We present an alternative zero-knowledge protocol for ORE using a *mod-counter*, a measurement device that outputs neutron counts modulo N. When N is greater than the maximum possible neutron count, the neutron counts are equal if and only if they are congruent modulo N.

Who brings the mod-counter $\mathcal{M}$? The verifier cannot trust a prover's device to output correct values. Likewise, the prover cannot trust the verifier, who might program the device to secretly store actual integer count values. Thus, our solution is to have the verifier *program check* the prover's mod-counter. We model $\mathcal{M}$ as an adaptive program computing a sequence of functions $\{f_i\}$ on a sequence of inputs d_i, where each f_i is a function of $d_1, ..., d_i$, and each d_i is a physical neutron detector. We assume that the prover cannot remotely change the device's state once the verifier takes possession of it.

1. **Prover's preparation:** The prover prepares and labels m detector pairs. In each pair $1 \leq i \leq m$, the prover initializes the count of both detectors, d_i^1 and d_i^2, to a random $r_i \in [0, N)$. (Only a dishonest prover may set d_i^1 to r_i^1 and d_i^2 to r_i^2 such that $r_i^1 \neq r_i^2$).
2. **Verifier's tests:** We assume the verifier has a device to obtain the neutron count of any detector. For each ith detector pair, the verifier performs *RandomCompute* with probability 1/5, and otherwise runs *RandomTest*:
   - *RandomCompute.* Choose random values $y_i^1, y_i^2 \in [0, N)$. Increase the count of d_i^1 by y_i^1 to $r_i^1 + y_i^1$, and increase the count of d_i^2 to $r_i^2 + y_i^2$. Select random θ_i, and run the neutron source, recording the scattering at θ_i from X using d_i^1 and from X' using d_i^2. Query $\mathcal{M}$ with d_i^1 and d_i^2. The test passes if and only if $\mathcal{M}(d_i^1) - y_i^1 = \mathcal{M}(d_i^2) - y_i^2$.

– *RandomTest.* Measure the detectors to uncover r_i^1 and r_i^2, and reject the protocol if $r_i^1 \neq r_i^2$. Select random $t_i^1, t_i^2 \in [0, 2N)$. Increase the count of d_i^1 to $r_i^1 + t_i^1$, and increase the count of d_i^2 to $r_i^2 + t_i^2$. Query $\mathcal{M}$ with d_i^1 and d_i^2. The test passes if and only if $\mathcal{M}(d_i^1) \equiv r_i^1 + t_i^1 \ (mod\ N)$ and $\mathcal{M}(d_i^2) \equiv r_i^2 + t_i^2 \ (mod\ N)$.

Completeness: When $x_{\theta_i} = x'_{\theta_i}$ for all i, then an honest prover will pass all rounds. Since $\mathcal{M}$ behaves correctly, it will pass every RandomTest. For all i, the prover sets $r_i^1 = r_i^2 = r_i$ so that in RandomCompute $\mathcal{M}(x_{\theta_i} + r_i + y_i^1) - y_i^1 = \mathcal{M}(x'_{\theta_i} + r_i + y_i^2) - y_i^2 \equiv x_{\theta_i} + r_i \ (mod\ N)$.

Zero-Knowledge: If the prover is honest, then on every round the initialization values are $r_i^1 = r_i^2 = r_i$ and the neutron counts collected from X and X' and angle θ_i are $x_{\theta_i} = x'_{\theta_i} = x_i$. Since r_i is uniformly distributed in $[0, N)$, $x_i + r_i \ (mod\ N)$ is also uniformly distributed in $[0, N)$, independent of x_i.

Soundness: The success probability of a cheating prover is bounded by $(\frac{4+\beta}{5})^m$ where $\beta < 1$ is an upper bound on the probability that $x_\theta = x'_\theta$ at a uniformly distributed angle θ when $X \not\sim X'$.

Proof: To simplify, assume $\mathcal{M}$'s input on each round is the *pair* of detectors. In reality, $\mathcal{M}$ is actually weaker, since it cannot examine the second detector before outputting a response for the first. We'll work with the hybrid model $\mathcal{M}$, which is an oracle taking integer pair inputs. Denote the output $\mathcal{M}(a, b) = (\mathcal{M}_1(a, b), \mathcal{M}_2(a, b))$. Further, assume that the prover knows the values of x_θ and x'_θ at every angle θ, and can predict the sequence of angles $\{\theta_i\}_{i=1}^{2m}$ that the verifier selects. (Soundness with a stronger prover holds against a weaker one). In calculating our upper bound we only consider deterministic $\mathcal{M}$ strategies since the verifier's strategy is independent and for any fixed protocol input there is a deterministic strategy for $\mathcal{M}$ that maximizes the soundness error.

Case 1: $x_{\theta_i} = x'_{\theta_i}$, occurs with probability β. The prover doesn't benefit from cheating. Case 2: $x_{\theta_i} \neq x'_{\theta_i}$, the prover's strategies are to either cheat on the initializations so that $x_{\theta_i} + r_i^1 = x'_{\theta_i} + r_i^2$, or instead to keep $r_i^1 = r_i^2$, and program $\mathcal{M}$ to cheat. In the latter case, the prover preselects the $k_i \leq N^2$ pairs for which $\mathcal{M}$ should pass RandomCompute. Setting $\mathcal{M}(a, b) = (a \ mod\ N, b + x_{\theta_i} - x'_{\theta_i} \ mod\ N)$ results in $k_i = N^2$.

If γ_i is the conditional probability that the prover passes RandomCompute on the *ith* round given that $x_{\theta_i} \neq x'_{\theta_i}$, then Lemma 2 implies the prover succeeds in round i with probability at most: $\beta + (1 - \beta)\left(\frac{1}{5}\gamma_i + \frac{4}{5}(1 - \frac{\gamma_i}{4})\right) = \beta + (1 - \beta)\frac{4}{5} = \frac{4+\beta}{5}$.

Lemma 2. *The prover's probability of passing RandomTest on the ith round given that $x_{\theta_i} \neq x'_{\theta_i}$ is at most $1 - \gamma_i/4$, and the optimal prover strategy achieves this probability exactly.*

Proof. Consider first the case where $r_i^1 = r_i^2 = r_i$. In this case, $\gamma_i = \frac{k_i}{N^2}$, where k_i is the number of pairs (y_i^1, y_i^2) in $[0, N) \times [0, N)$ for which $\mathcal{M}_1(x_{\theta_i} + r_i^1 + y_i^1) - y_i^1 = \mathcal{M}_2(x'_{\theta_i} + r_i^2 + y_i^2) - y_i^2$. For each of these k_i pairs, let $t_i^1 = x_{\theta_i} + y_i^1$ and $t_i^2 = x'_{\theta_i} + y_i^2$. Either $\mathcal{M}_1(t_i^1 + r_i) \neq t_i^1 \ (mod\ N)$ or $\mathcal{M}_2(t_i^2 + r_i) \neq t_i^2 \ (mod\ N)$, and so $\mathcal{M}$ would fail RandomTest if the verifier chooses the pair (t_i^1, t_i^2). Thus, there are at least k_i distinct pairs in $[0, 2N) \times [0, 2N)$, that cause $\mathcal{M}$ to fail RandomTest, and the prover's conditional probability of passing RandomTest is at most $1 - \frac{k_i}{(2N)^2} = 1 - \frac{\gamma_i}{4}$. The prover can program $\mathcal{M}$ to pass (i.e. behave normally) on all other $4N^2 - k_i$ pairs, so there is a prover strategy that passes RandomTest with probability exactly $1 - \frac{\gamma_i}{4}$. We will show that this strategy is optimal. Consider the second case where the prover cheats by setting r_i^1 and r_i^2 so that $x_{\theta_i} + r_i^1 = x'_{\theta_i} + r_i^2$. With this strategy, $\gamma_i = 1$, but since $r_i^1 \neq r_i^2$, the prover always fails RandomTest, failing the overall round with probability 1/2. This strategy is suboptimal because the previous strategy passes with probability 3/4 when setting $k_i = N^2$ so that $\gamma_i = 1$.