Provably Secure Substitution of Cryptographic Tools

Lea Kissner
leak@cs.cmu.edu

David Molnar
dmolnar@cs.berkeley.edu

Abstract

Many cryptographic protocols secure against malicious players use specially designed cryptographic tools. Essentially, these special tools function much like less-expensive tools, but give extra 'powers' to a reduction or simulation algorithm. Using these powers, cryptographers can construct a proof of security using standard techniques. However, these powers are not available to either the honest parties or the adversary. In a large class of protocols, by replacing the expensive, specially designed cryptographic tool with a corresponding less-expensive tool, we can improve the protocol's efficiency without changing the functionality available to either the adversary or the honest parties. The key motivating question we address in this paper is whether the new, 'substituted' protocol is still secure.

We introduce a framework for reasoning about this question. Our framework uses *translators*: special purpose oracles that map outputs of one cryptographic tool to corresponding outputs of a different tool. Translators are similar to, but generally weaker than, the "angels" of Prabhakaran and Sahai [29]. We introduce the notion of *substitution-friendly* protocols and show that such protocols remain secure after substitution in our framework. We also leverage existing proofs of security; there is no need to re-prove security from scratch. We demonstrate our framework with a non-interactive non-malleable bit commitment protocol.

Keywords: secure substitution, proof techniques, simulation proof, protocols

1 Introduction

In the past two decades, researchers have developed techniques to construct proofs of security for cryptographic protocols. These methods have greatly advanced our ability to make rigorous guarantees of security. Often, however, one must introduce additional overhead to the protocol or utilize additional assumptions in order to prove security using these techniques.

As an illustration, consider trapdoor commitment schemes. These are commitment schemes with a "public key"; knowledge of the corresponding private key allows one to open commitments to any value. In some protocols, the private key is available only to the simulator in the proof and used *only* in the proof of security. In this case, intuitively, the trapdoor commitment 'behaves like' a standard commitment scheme to both the honest and malicious players. Unfortunately, trapdoor commitments are far more expensive than the non-trapdoor variety.

Use of trapdoor commitments is just one example of a widespread pattern in proofs of security for cryptographic protocols. Protocols include a 'back door', often at great expense, necessary for the proof of security yet inaccessible during normal execution (e.g. [17, 2, 9, 18, 32, 15, 12, 5] and others). Indeed, if these back doors were accessible, then adversaries could use them to break the security of the protocol. Unfortunately, adding such a 'back door' may require an expensive cryptographic tool. However, it was an open question whether these expensive tools are needed for secure protocols and whether removing these back doors (and thus increasing efficiency) can be done without compromising the protocol's security.

In this paper, we introduce a new framework to analyze the security of such modified protocols, in which tools with back doors are replaced by 'workalike' tools without back doors. This substitution allows

us to greatly increase the efficiency of many protocols and often to remove extraneous assumptions and the necessity for trusted setup.

Our argument, unfortunately, is not as simple as taking the adversary against the substituted protocol and performing a standard reduction. An adversary can easily detect that the cryptographic tool he is using has changed! However, many protocols utilize an expensive cryptographic tool as a black box. We leverage this property to show that the execution of the protocol using this black-box cryptographic tool is, in a meaningful sense, equivalent to execution with a workalike tool. As the original protocol has already been proved secure against malicious players, we can thus conclude that the protocol utilizing a workalike tool is also secure.

Overall, we make the following contributions. In Section 4, we introduce a novel framework for proving security in which *translators* map outputs of one cryptographic tool to outputs of a different cryptographic tool, and define sufficient conditions for *secure substitution* of one cryptographic primitive for another in a protocol. We show in Section 5 that if these conditions are met, then many cryptographic security properties are preserved after substitution. Our security proofs allow us to re-use *existing* proofs of security for cryptographic protocols. To show the simplicity of applying secure substitution, we walk through a coin-flipping example in Section 7. Finally, in Section 8, we show how our techniques yield a non-interactive, non-malleable commitment scheme without a common reference string, which to the best of our knowledge has not been constructed in the standard model.

We stress that this work is intended as a preliminary step towards reconciling practical and provable protocols. In particular, we do not address the question of general composition of cryptographic protocols, in the sense of Canetti's universal composability framework. Nothing in our framework precludes composition, but we focus first on showing secure substitution for single protocols. A second limitation is that our current work takes translators as special entities with special powers used only for the purposes of the proof, in a manner similar to Prabhakaran and Sahai's "angels" [29]; in Section 2 we discuss this and other related work. In Section 6 we suggest possible ways to realize translators without special assumptions, but we have no such constructions at this time. Nevertheless, we show that one can use the translator framework to leverage existing proofs of security to obtain new proofs of security for more efficient, related protocols. Before this work there were no known analysis techniques for this class of protocols, and thus our work represents an important step towards improving the state of provable security for efficient protocols.

2 Related Work

Virtually all previous work on protocols provably secure against malicious players utilizes expensive cryptographic tools or additional assumptions necessary only for the proof of security. For example, in a simulation proof with a common reference string, the simulator can create the CRS so that he knows the secret trapdoor key. This secret key can be utilized to allow the simulator to 'cheat' by opening a trapdoor commitment to any value. During normal execution, however, the CRS is chosen uniformly at random. Therefore, in normal execution, no party knows the secret key; with overwhelming probability, neither adversaries nor honest parties can cheat when opening a trapdoor commitment.

Many examples exist of "special powers" available only to the simulator; we list only a few in this section. Blum, Feldman, and Micali create a non-interactive zero-knowledge proof using a CRS divided into elements of Z_n^*; the simulator can generate a CRS in such a way that it knows which elements are quadratic residues, but neither the adversary nor the honest player can determine this without violating the Quadratic Residuosity Assumption [8]. Feige, Lapidot, and Shamir use a witness-hiding proof of a disjunction that either a string x is in a language L or that part of a common reference string is pseudorandom [17]. A simulator can create a pseudorandom CRS, but normal players cannot. Damgard

showed how a simulator that knows a secret trapdoor commitment key can perform *concurrent* simulation by opening trapdoor commitments after seeing a verifier's challenges [13]. More recently, Barak's non-black-box techniques in zero-knowledge protocols involve proving, roughly, that either a statement is correct or that the prover knows the verifier's code [2]. A simulator can know the verifier's code, but normal players cannot.

In the area of commitment schemes, Di Crescenzo, Ishai, and Ostrovsky used an equivocal commitment with the secret key known to the simulator to construct a non-mallable commitment [15]. The efficiency of such a scheme was improved by Di Crescenzo, Katz, Ostrovsky, and Smith [12]. Canetti and Fischlin used a scheme with two different types of trapdoors to create a universally composable commitment [9]. Damgard and Nielsen improved on the efficiency of their construction [32]. Gennaro introduced "multi-trapdoor" commitments, in which each party of a protocol might have a different trapdoor key [18].

Los Angeles Network Security, Universal Composability, indifferentiability, and the Reactive Systems framework are techniques to prove the security of the composition of multiple cryptographic systems [30, 10, 28]. Our current work does not address such composition, although it is not ruled out by our approach; in future work, we intend to apply our techniques to composed settings. Instead our work focuses on formalizing an intuitive sense that two protocols which effectively differ very little during normal execution are 'equivalent' in practice. We do not construct protocol simulators to prove security, like those used in the above techniques. This allows us to analyze a large class of protocols for which we are aware of no previous security results.

In some sense, our translators resemble the "imaginary angels" in the Los Angeles Network-Aware Security framework of Prabhakaran and Sahai [30]. Prabhakaran and Sahai's superpolynomial angels find collisions for certain carefully-chosen instances of a collision-resistant hash function. Like angels, our translators are entites introduced for purposes of a security proof. Our translators, in contrast, do not explicitly "break" any security properties, nor must they necessarily be superpolynomial. Instead, they map outputs of one tool to outputs of a different tool.

3 Preliminaries

Allowed malicious adversaries. We use the notion of an *allowed adversary*; this means that we consider adversaries in a particular class of algorithms, such as all probabilistic polynomial time (PPT) algorithms. however, our results do not depend on the choice of the class. The standard definition of cryptographic security for multi-party computation (see, e.g., [19]) is based on a comparison between the ideal model and a trusted third party (TTP), where a malicious party may give arbitrary input to the TTP. This security definition is also limited to the case where at least one of the parties is honest.

A simulation proof is a common method of proving security under such a definition: the simulator G provides a concrete method of translating any strategy executed by Γ to a strategy in the TTP model. We illustrate such a proof in Figure 1.

Indistinguishability. Let D_1, D_2 be probability distributions over arbitrary domains. Denote the samples $x_1 \leftarrow D_1$, $x_2 \leftarrow D_2$. $x_1 \sim x_2$ if, for any allowed algorithm $\mathcal{D}$ (i.e. an allowed adversary) $\mathcal{D}$, $|\Pr[\mathcal{D}(x_1) = 1] - \Pr[\mathcal{D}(x_2) = 1]|$ is negligible [24]. If the allowed class of adversaries consists of probabilistic polynomial time adversaries, we say that D_1 and D_2 are *computationally indistinguishable* [24].

Indifferentiability. Let two tools $\mathcal{Z}$, $\mathcal{Y}$ have private interfaces (accessible to the honest players Υ) $\mathcal{Z}^1$, $\mathcal{Y}^1$ and public interfaces (accessible to the malicious players Γ) $\mathcal{Z}^2$, $\mathcal{Y}^2$, respectively. $\mathcal{Y}$ is indifferentiable from $\mathcal{Z}$, denoted $\mathcal{Y} \sqsubset \mathcal{Z}$, if there exists an algorithm $\mathcal{S}$ such

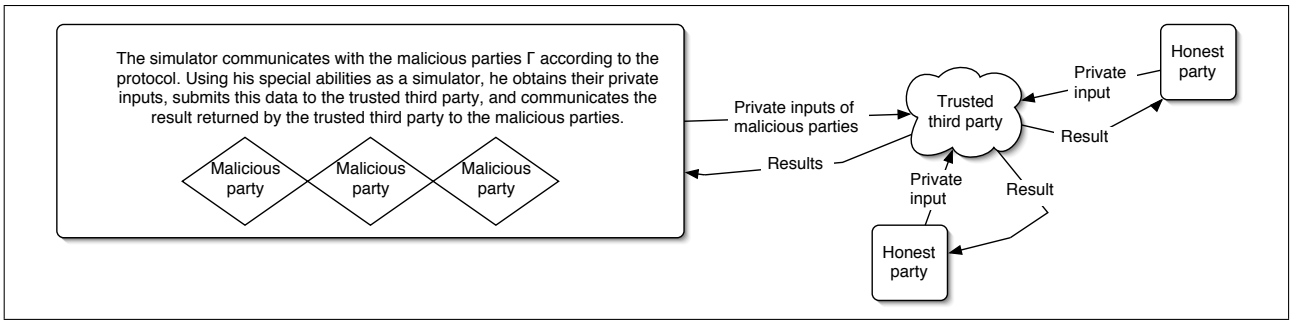
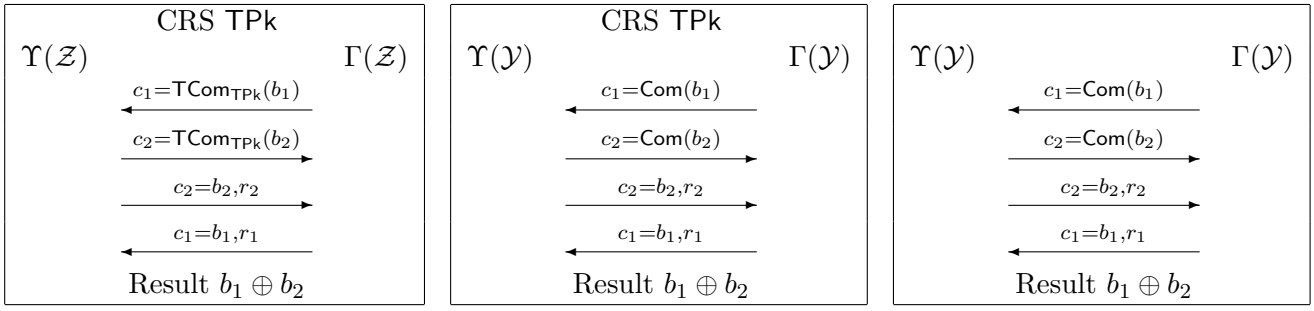


Figure 1: Basic outline of a standard simulation proof.



(a) Simulatable with CRS (b) Efficient with CRS (c) Efficient without CRS

Figure 2: Three variants of a coin-flipping protocol. In (a), a trapdoor commitment scheme TCom with a common reference string (CRS) allows a simulator to manipulate the result of the protocol. The protocol in (b) more efficiently duplicates this functionality for all real-world players using a more-efficient commitment scheme Com. In protocol (c), we have removed the common reference string required in protocol (a).

that for any allowed distinguisher $\mathcal{D}$, $\left|\Pr\left[\mathcal{D}^{\mathcal{Y}^1,\mathcal{Y}^2}=1\right]-\Pr\left[\mathcal{D}^{\mathcal{Z}^1,\mathcal{S}^{\mathcal{Z}^2}}=1\right]\right|$ (in the original notation, $\left|\Pr\left[\mathcal{D}(\mathcal{Y}^1,\mathcal{Y}^2)=1\right]-\Pr\left[\mathcal{D}(\mathcal{Z}^1,\mathcal{S}(\mathcal{Z}^2))=1\right]\right|$) is negligible [23].

4 Definitions

In this section, we formally define the novel terms used in this paper. For clarity, we show how each of these terms applies to the example protocols in Figure 2.

The protocol of Figure 2(a) utilizes TCom, while the protocols of Figure 2(b) and 2(c) utilize Com. Both TCom and Com are examples of *cryptographic tools*.

Definition 1. *A* **cryptographic tool** *is a Turing machine that has a defined set of interfaces, utilized in cryptographic protocols. Each interface operates as follows: when data is written to an input tape, the corresponding function (defined as part of the cryptographic tool) is computed on that input. The result of that computation is written to the corresponding output tape. The tool may also specify that only certain players may have access to an interface; this restriction is generally enforced by means of a secret key, such as in decryption.*

Both TCom and Com are concrete instantiations of a single *ideal cryptographic tool*; this tool has interfaces for commitment and decommitment. In addition, this ideal tool has a 'cheating' interface which

is only accessible to the simulator, as used in the TCom tool. Because this interface is not accessible to players in the real world, both TCom and Com effectively function like this ideal tool.

Definition 2. *An* **ideal tool** *I is a cryptographic tool, which may be accessed as an oracle by all players, with a defined set of interfaces $\mathcal{I}_1, \ldots, \mathcal{I}_F$.*

One way that a concrete tool can 'function like' an ideal tool is through *simulation security*. TCom is simulation-secure with respect to the ideal commitment tool.

Definition 3. *A tool $\mathcal{Z}$ is* **simulation-secure with respect to** *some ideal tool I if there exists a simulator that establishes that no adversary Γ can, with non-negligible probability, gain more information by interaction with the interfaces of $\mathcal{Z}$ than by use of I. That is, there exists a PPT algorithm $\mathcal{S}$ such that the distribution of transcriptions of the interaction of any allowed adversary Γ with $\mathcal{Z}$ is indistinguishable from the interaction of Γ with $\mathcal{S}(I)$.*

Another way that a concrete tool can 'function like' an ideal tool is by indifferentiability. Com is indifferentiable from a suitably-defined ideal commitment tool. Because both TCom and Com function like a suitably-defined ideal tool, Com is a workalike of TCom.

Definition 4. *Tool $\mathcal{Y}$ is a* **workalike** *of tool $\mathcal{Z}$ for ideal tool I if: 1) $\mathcal{Z}$ is simulation-secure with respect to I and 2) $\mathcal{Y} \sqsubset I$ for all players. For simplicity, we denote interfaces of $\mathcal{Y}$ and $\mathcal{Z}$ in the arbitrary order used to label the interfaces of I; that is, $\mathcal{Y}_i$, $\mathcal{Z}_i$, and $\mathcal{I}_i$ $(1 \leq i \leq F)$ are corresponding interfaces.*

Note that, for practical choices of Com, for any equivocal commitment scheme TCom, Com $\not\sqsubset$ TCom. For example, the Pederson commitment scheme [27] is differentiable from the Naor commitment scheme [25], as they utilize different domains for their commitments. We call these outputs that make the tools indifferentiable *handles*.

Definition 5. *Let $\mathcal{HO}$ be a minimal set of interfaces such that $\{\mathcal{I}_i\}_{i \notin \mathcal{HO}}$ is indistinguishable from $\{\mathcal{Z}\}_{i \notin \mathcal{HO}}$ to any allowed adversary. The output of any interface in $\mathcal{HO}$ is a* **handle**.

As we cannot compare handles output by different tools directly, we instead compare the inputs that were used to create those handles. We do not need to extract these inputs; it is sufficient for our proof to show that they are computationally indistinguishable. Equal-sized lists of handles created by Com and TCom from indistinguishable (to allowed adversaries) inputs are *translation-indistinguishable*.

Definition 6. *Let $\mathcal{H}_\mathcal{Y}, \mathcal{H}_\mathcal{Z}$ be equal-sized, ordered lists of handles. (associated with cryptographic tools $\mathcal{Y}$ and $\mathcal{Z}$, respectively). Let the interfaces of two tools $\mathcal{Z}$ and $\mathcal{Y}$ that accept $|\mathcal{H}_\mathcal{Y}| = |\mathcal{H}_\mathcal{Z}|$ handles as input be denoted $\mathcal{HI}$. We recursively define $\mathcal{H}_\mathcal{Y}$ and $\mathcal{H}_\mathcal{Z}$ to be* **translation-indistinguishable** *if*

- $\forall_{x \in \{0,1\}^*}\ \{\mathcal{Y}_i(\mathcal{H}_\mathcal{Y}, x)\}_{i \in \mathcal{HI}} \sim \{\mathcal{Z}_i(\mathcal{H}_\mathcal{Z}, x)\}_{i \in \mathcal{HI}}$
- *Let corresponding subsets $\mathcal{H}'_\mathcal{Y}, \mathcal{H}'_\mathcal{Z}$ be defined such that for some list L of set indices which specify subsets of size $|\mathcal{H}_\mathcal{Y}| - 1$, $L \in \binom{[|\mathcal{H}_\mathcal{Y}|]}{|\mathcal{H}_\mathcal{Y}|-1}$, $j \in L \Leftrightarrow (\mathcal{H}_\mathcal{Y})_j \in \mathcal{H}'_\mathcal{Y} \wedge (\mathcal{H}_\mathcal{Z})_j \in \mathcal{H}'_\mathcal{Z}$. All corresponding subsets $\mathcal{H}'_\mathcal{Y}, \mathcal{H}'_\mathcal{Z}$ must be translation-indistinguishable.*

Lists of handles created by Com and TCom from indistinguishable sets of inputs are *handleset-indistinguishable*; these input sets and lists may be of different sizes.

Definition 7. *Let $\mathcal{H}_1$, $\mathcal{H}_2$ be ordered lists of handles. Let $\mathcal{CD}((\mathcal{H}_1)_j)$ $(1 \leq j \leq |\mathcal{H}_1|)$ be the list of bit strings passed to a cryptographic tool to output handle $(\mathcal{H}_1)_j$. The lists of handles $\mathcal{H}_1$ and $\mathcal{H}_2$ are* **handleset indistinguishable**, *denoted $\mathcal{H}_1 \sim_\nu \mathcal{H}_2$, if $\{\mathcal{CD}((\mathcal{H}_1)_1), \ldots, \mathcal{CD}((\mathcal{H}_1)_{|\mathcal{H}_1|})\} \sim \{\mathcal{CD}((\mathcal{H}_2)_1), \ldots, \mathcal{CD}((\mathcal{H}_2)_{|\mathcal{H}_2|})\}$.*

In our proofs, we utilize a *translator*: a mapping between handles output by cryptographic tools. In our example, a translator from TCom to Com would output Com(3) on input TCom(3).

Definition 8. *Let cryptographic tool A be a workalike of tool B, or vice versa, with respect to ideal tool I. A* **translator** $\mathcal{T}$ *inputs handles from A, and outputs the corresponding handles (handles output by the corresponding functionality) from B; formally, $\mathcal{T} = \{\mathcal{T}_{\mathcal{H}_1}, \ldots, \mathcal{T}_{\mathcal{HO}_{|\mathcal{HO}|}}\}$, $\mathcal{T}_i : Range(A_i) \rightarrow Range(B_i)$ $(i \in \mathcal{HO})$.*

Such translation is correct in the presence of an allowed adversary Γ if, with overwhelming probability, for an $x \in \{0,1\}^*$ and $i \in \mathcal{HO}$ both chosen by Γ after interaction with $\mathcal{T}$, A, B:

- *Let $h_1 \leftarrow A_i(x)$, $h_2 \leftarrow B_i(x)$*
- *$\mathcal{T}(h_1) \sim h_2$*
- *$\mathcal{T}(h_2)$ and h_2 are translation-indistinguishable*

In most cases, we will utilize translators that are correct against a class of adversaries, e.g. all probabilistic polynomial time adversaries. In this case, we will drop the explicit dependence on Γ and simply refer to a correct translator $\mathcal{T}$*.*

Our main security result states that, for any cryptographically secure *replacement friendly protocol* using a cryptographic tool $\mathcal{Z}$, if the protocol is also secure when using an ideal tool I, then it is secure when using a workalike tool $\mathcal{Y}$. We give sufficient conditions for a replacement-friendly protocol:

Definition 9. *A* **replacement-friendly** *protocol using tool $\mathcal{Z}$ is one in which: 1) no player is required to compute a function of any handle from $\mathcal{Z}$, other than through black-box invocation of an interface of $\mathcal{Z}$. 2) There exists a PPT eavesdropper algorithm that can determine with overwhelming probability, for each handle the protocol requires a player to send, the interface i $(i \in \mathcal{HO})$ that an honest player would have used to construct that handle.*

Note that copying a handle does not disqualify a protocol from being replacement-friendly, but computing any non-identity function of a handle (such as equality) does. Thus, replacement-friendly protocols do not include certain pathological protocols, such as those of [11], that 'detect' that they are using a certain tool and purposely break. It does, however, include a large class of protocols, such as the example protocols in Figure 2.

5 Proof of Secure Replacement

Before sketching our proof, we prove several lemmas in Section 5.1. We then give a general proof sketch of secure translation, given abstract translators T' and T. In Section 6, we explore several scenarios in which we may construct such translators; these constructions prove security in each scenario.

5.1 Indistinguishability Lemma

Because of the excessive length of our general proof of security, we give only a proof sketch in this section, deferring the full proof to Appendix B. The operation of players in a protocol can be viewed as a series of function computations. As these functions fall into the 'allowed' range of adversary operation, the results cannot aid any allowed adversary in distinguishing previously indistinguishable data. We formalize this intuition in the following lemma.

Lemma 1. *For all functions f computable by an allowed adversary, $x \sim y \rightarrow f(x) \sim f(y)$.*

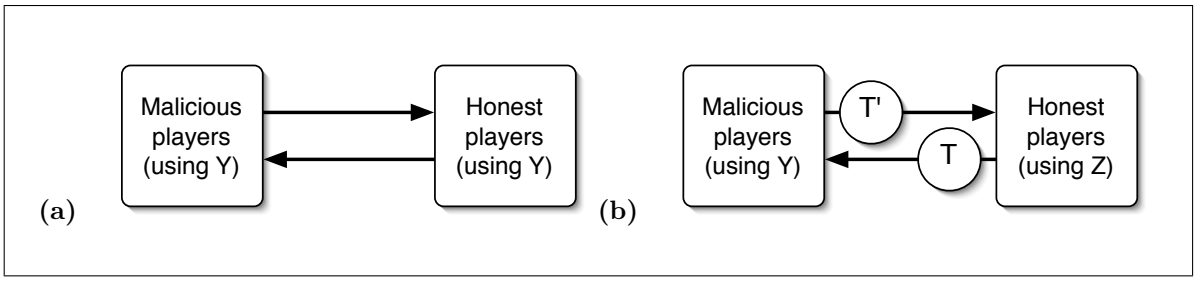


Figure 3: To prove security, we prove that Γ cannot distinguish between scenarios: (a) the normal operation of the protocol using $\mathcal{Y}$; (b) translations of handles to and from the honest players, where Υ utilizes $\mathcal{Z}$.

Proof. If $f(x)$ is distinguishable from $f(y)$ by an allowed adversary, then there exists an adversary who can distinguish x and y through calculation of f. By the definition of indistinguishability, there exists no such adversary. Thus, by contradiction, $f(x)$ is indistinguishable from $f(y)$. □

Corollary 2. *Lemma 1 holds for any number of function parameters ℓ. Formally, if each parameter $x_j \sim y_j$ $(1 \leq j \leq \ell)$, then for all $\{w_1, \ldots, w_\ell \mid w_j \in \{x_j, y_j\}\}$, $f(w_1, \ldots, w_\ell)$ are mutually computationally indistinguishable.*

Corollary 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be black-box subroutines such that $\forall_{x \in \{0,1\}^*} \mathcal{A}(x) \sim \mathcal{B}(x)$. Lemma 1 also holds for functions f which receive access to either $\mathcal{A}$ or $\mathcal{B}$. Formally, for all functions f, $x \sim y \rightarrow f^{\mathcal{A}}(x) \sim f^{\mathcal{B}}(x) \sim f^{\mathcal{A}}(y) \sim f^{\mathcal{B}}(y)$. Similarly to Lemma 2, this lemma may be extended to allow for any number of function parameters.*

5.2 Proof of Security Under Substitution

We now sketch the proof of a general theorem of security against allowed adversaries for substitution of the workalike cryptographic tool $\mathcal{Y}$ for the tool $\mathcal{Z}$. This theorem requires correct translators from $\mathcal{Y}$ to $\mathcal{Z}$ and vice versa. To ensure security, we prove an indistinguishability condition is preserved through execution of the generic replacement-friendly protocol $\mathcal{P}(\cdot)$, essentially that execution using $\mathcal{Y}$ and $\mathcal{Z}$ is indistinguishable to Γ. We denote as $\mathcal{P}(\mathcal{Z})$ (resp. $\mathcal{P}(\mathcal{Y})$) the protocol using the tool $\mathcal{Z}$ (resp. $\mathcal{Y}$).

Theorem 4. *Let $\mathcal{Y}$ be a workalike of $\mathcal{Z}$ with respect to ideal tool I. Let $\mathcal{P}(\mathcal{Z})$ be a replacement-friendly protocol. Let the translator T' ($T' = \{T'_{\mathcal{H}_1}, \ldots, T'_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T'_i : Range(\mathcal{Y}_i) \rightarrow Range(\mathcal{Z}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Y}$ to $\mathcal{Z}$. Let the translator T ($T = \{T_{\mathcal{H}_1}, \ldots, T_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T_i : Range(\mathcal{Z}_i) \rightarrow Range(\mathcal{Y}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Z}$ to $\mathcal{Y}$.*
Any allowed adversary Γ utilizing $\mathcal{P}(\mathcal{Y})$ cannot distinguish between a scenario in which he interacts with Υ utilizing $\mathcal{P}(\mathcal{Y})$, and a scenario in which he interacts (through T and T') with Υ utilizing $\mathcal{P}(\mathcal{Z})$, as shown in Figure 3.

Proof. We prove that any allowed adversary Γ cannot distinguish between scenarios (a) and (b) of Figure 3 by proving that the view of Γ is indistinguishable between these scenarios. To formalize the views of Γ and Υ, we give a general *functional decomposition* of a generic protocol in Figure 4. The operation of any protocol can be represented as a call-and-response style generic protocol between the adversaries Γ and honest players Υ. In this generic protocol, Γ is given the opportunity to send some data o_0 to Υ, who may respond with o'_1; Γ then responds with o_1. The protocol continues sequentially from there until the honest player makes the last move o'_{z-1} or the malicious player makes the last move o_z. Note that

Protocol Operation:

$$\begin{array}{lll} \Gamma & o_0 \rightarrow & \\ & \leftarrow o'_0 & \Upsilon \\ \Gamma & o_1 \rightarrow & \\ & \vdots & \\ & \leftarrow o'_{z-1} & \Upsilon \\ \Gamma & o_z \rightarrow & \end{array}$$

Scenario (a):

Malicious Players	**Honest Players ($\mathcal{Y}$)**
$o^a_0, s^a_0 \leftarrow \Gamma_0(\text{initial inputs})$	
	$o'^a_0, s'^a_0 \leftarrow \Upsilon^{\mathcal{Y}}_0(\text{initial inputs}, o^a_0)$
$o^a_1, s^a_1 \leftarrow \Gamma_1(s^a_0, o'^a_0)$	
$\vdots$	$\vdots$
	$o'^a_{z-1}, s'^a_{z-1} \leftarrow \Upsilon^{\mathcal{Y}}_{z-1}(s'^a_{z-2}, o^a_{z-1})$
$o^a_z, s^a_z \leftarrow \Gamma_z(s^a_{z-1}, o'^a_{z-1})$	

Scenario (b):

Malicious Players	**Honest Players ($\mathcal{Z}$)**
$o^c_0, s^c_0 \leftarrow \Gamma_0(\text{initial inputs})$	
	$o'^c_0, s'^c_0 \leftarrow \Upsilon^{\mathcal{Z}}_0(\text{initial inputs}, T'(o^c_0))$
$o^c_1, s^c_1 \leftarrow \Gamma_1(s^c_0, T(o'^c_0))$	
$\vdots$	$\vdots$
	$o'^c_{z-1}, s'^c_{z-1} \leftarrow \Upsilon^{\mathcal{Z}}_{z-1}(s'^c_{z-2}, T'(o^c_{z-1}))$
$o^c_z, s^c_z \leftarrow \Gamma_z(s^c_{z-1}, T(o'^c_{z-1}))$	

Figure 4: Let the Protocol Operation show the operation of a generic protocol. The operation of malicious players Γ in such a protocol may be represented as a series of functions $\Gamma_0, \ldots, \Gamma_z$. We may thus represent the operation of the malicious and honest players in scenarios (a) and (b) of Figure 3. We call this a *functional representation* of a protocol.

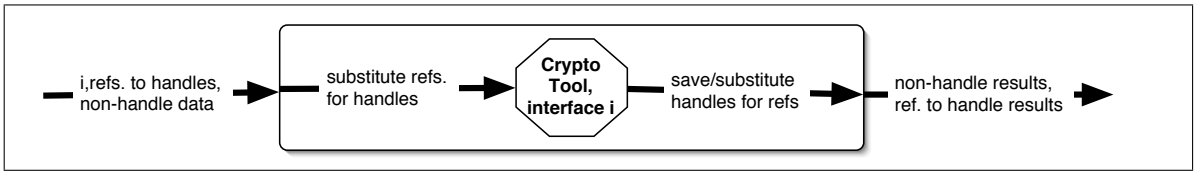


Figure 5: In our proof, Υ utilizes subroutines $\mathcal{A}$ or $\mathcal{B}$ to perform all calculation on handles. We illustrate their operation here: they take an interface identifier i, and arguments to that interface (but using handle references instead of handles); the subroutine then substitutes handles for handle references, applies the cryptographic tool's ith interface, and erases all intermediate results; the subroutine then saves any new handles created (substituting handle references) and returns the results.

any real protocol may use any combination of possible communication between groups of malicious and honest adversaries; we have given a completely generic representation of any protocol $\mathcal{P}$ for purposes of our proof.

In this generic protocol, we can break the computation performed by Γ in the steps $0, \ldots, z$ into a series of functions $\Gamma_0, \ldots, \Gamma_z$. Each of these functions at step ℓ takes as input the last state of Γ, denoted $s_{\ell-1}$, and the last move of Υ, denoted $o'_{\ell-1}$, and outputs new state s_ℓ and output o_ℓ to be sent to Υ. We may also perform a similar decomposition on the operation of Υ. We describe a full functional decomposition, for both scenarios (a) and (b) in Figure 4.

We can then examine the outputs created by Γ and Υ at each stage; if all data seen by Γ is indistinguishable between scenarios (a) and (b), then $\mathcal{P}(\mathcal{Y})$ is secure. We may prove this by induction. In order to complete our inductive proof, we also prove that all non-handle data in the state of Υ is indistinguishable between the two scenarios, and all that all handles calculated by Υ are handleset-indistinguishable between the two scenarios. Let $\nu(\cdot)$ output the set of handles included in its argument, and $\bar{\nu}(\cdot)$ output all non-handle data in its argument.

As the base case, note that Γ is calculating an identical function Γ_0 on identical data in scenarios (a) and (b). Thus, by Lemma 2, the outputs of this function are indistinguishable between these scenarios.

$$\begin{aligned} \text{initial inputs} &\sim \text{initial inputs} \\ \Gamma_0(\text{initial inputs}) &\sim \Gamma_0(\text{initial inputs}) \\ \{s_0^a, o_0^a\} &\sim \{s_0^b, o_0^b\} \\ \nu(o_0^a) &\sim_\nu \nu(o_0^b) \end{aligned}$$

To complete the inductive proof, we must prove that when either Υ or any allowed adversary Γ computes output as part of the protocol, any data visible to Γ is indistinguishable between the two scenarios. (Due to length considerations, we defer the detailed version of this proof to Appendix B.)

Γ steps. Consider some step ℓ of the protocol, in which Γ calculates Γ_ℓ. By the inductive hypothesis, we know that all data previously available to Γ is indistinguishable between scenarios (a) and (b). By Lemma 2, any function calculated as a function of that data outputs indistinguishable data as well. Thus, the outputs of Γ at any step ℓ are indistinguishable between scenarios (a) and (b).

$$\begin{aligned} \{s_\ell^a, o_\ell'^a\} &\sim \{s_\ell^b, T(o_\ell'^b)\} \\ \Gamma_{\ell+1}(s_\ell^a, o_\ell'^a) &\sim \Gamma_{\ell+1}(s_\ell^b, T(o_\ell'^b)) \\ \{s_{\ell+1}^a, o_{\ell+1}^a\} &\sim \{s_{\ell+1}^b, o_{\ell+1}^b\} \\ \nu(o_{\ell+1}^a) &\sim_\nu \nu(o_{\ell+1}^b) \end{aligned}$$

Υ steps. Consider some step ℓ of the protocol, in which Υ calculates Υ_ℓ. By the inductive hypothesis, we know that the non-handle data $\bar{\nu}(s_{\ell-1}), \bar{\nu}(o_\ell)$ is indistinguishable between scenarios (a) and (b), and that the handles $\nu(s_{\ell-1}), \nu(o_\ell)$ are handleset-indistinguishable between scenarios (a) and (b). Note that handleset-indistinguishability is preserved under translation.

In order to complete this section of the inductive proof, we must examine the class of functions that Υ may compute as part of a replacement-friendly protocol. The most important properties, at this juncture, are that: (1) $\mathcal{P}(\cdot)$ requires no player to compute a function of any handle; and (2) $\mathcal{P}(\mathcal{Z})$ (resp., $\mathcal{P}(\mathcal{Y})$) requires no non-black-box usage of $\mathcal{Z}$ (resp., $\mathcal{Y}$). We may observe from this that Υ does not require the handles themselves in order to calculate Υ_ℓ, if it can indirectly specify the handles needed for cryptographic computation.

We will therefore, without loss of generality, represent the execution of the function Υ_ℓ as a function f (identical in scenarios (a) and (b)) computed upon non-handle data, with access to the cryptographic tool via a subroutine. In order to specify a handle for cryptographic computation, Υ_ℓ will utilize a *reference* to the handle. These references are simply non-handle data utilized to specify a handle, such as numbers assigned in the order the handles were calculated. We illustrate the operation of this subroutine in Figure 5.

Note that the non-handle and handle portions of the output of Υ_ℓ can be calculated without access to any handles, only references. Intuitively, as this non-handle data is indistinguishable between scenarios (a) and (b), the non-handle output is indistinguishable between (a) and (b), and the handles computed are handleset-indistinguishable. By suitable application of Lemma 3, we can show that all handles output by Υ_ℓ are handleset-indistinguishable between scenarios (a) and (b), and that that all non-handle data is indistinguishable between scenarios (a) and (b). By the definition of correct translation, the handles seen by Γ in the next step of the protocol are indistinguishable between scenarios (a) and (b). Thus, the inductive hypothesis holds for this step of the protocol as well. □

Corollary 5. *Let $\mathcal{Y}$ be a workalike of $\mathcal{Z}$ with respect to ideal tool I. Let $\mathcal{P}(\mathcal{Z})$ be a replacement-friendly protocol. Let the translator T' ($T' = \{T'_{\mathcal{H}_1}, \ldots, T'_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T'_i : Range(\mathcal{Y}_i) \rightarrow Range(\mathcal{Z}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Y}$ to $\mathcal{Z}$. Let the translator T ($T = \{T_{\mathcal{H}_1}, \ldots, T_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T_i : Range(\mathcal{Z}_i) \rightarrow Range(\mathcal{Y}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Z}$ to $\mathcal{Y}$.*
If $\mathcal{P}(\mathcal{Z})$ and $\mathcal{P}(I)$ are cryptographically secure against any allowed adversary Γ, then $\mathcal{P}(\mathcal{Y})$ is also cryptographically secure against Γ.

6 Toward Real-World Translators

So far in this work, we have considered translators as special oracles used only as part of the proof of security. A key open question is whether translators can be instantiated for pairs of interesting cryptographic tools in the 'real world'. We point out two possible directions. First, we might make a non-black-box assumption about a cryptographic tool sufficient to construct a translator. While non-black-box assumptions are controversial, they have been studied in several contexts [14, 20, 21, 6, 4]. Note Υ may cooperate with the translator, making a non-black-box assumption necessary only for the tool $\mathcal{Y}$.

Second, proxy re-encryption and re-signatures are two exisiting cryptographic primitives that exhibit 'translator-like' behavior: a semi-trusted proxy translates from one secret key to another [7, 16, 1]. Further, we show in Appendix C how to translate between two Pedersen commitments [27] with different public parameters. While this does not allow us to appreciably reduce the complexity of a protocol, it does show that some kind of translation is possible using only standard cryptographic assumptions. Techniques from these examples may lead to concrete constructions of translators for use in our framework.

7 Substitution in the Coin-Flipping Example

To show the simplicity of applying our techniques, we formalize the proof of security for the coin flipping example of Figure 2 before proceeding to our main application result in Section 8. To apply our security result, we must: (1) identify a suitable ideal tool; (2) show that the protocol is replacement-friendly; (3) show that $\mathcal{Y}$ is a workalike of $\mathcal{Z}$.

We first define the ideal tool I for a trapdoor commitment scheme as a cryptographic tool with the interfaces ($\mathsf{ICom}, \mathsf{IDCom}, \mathsf{IFakeCom}, \mathsf{IFakeDCom}$) shown in Figure 6. The protocol $\mathcal{P}(\mathrm{I})$ satisfies the properties of a mutually independent announcement, as defined by Liskov et al. [22], i.e. the commitments

Tool state: a table $T = \{0,1\}^k \times \{0,1\}^k$.
ICom(m): Pick $r_m \leftarrow \{0,1\}^k$. Insert (r_m, m) into T. Return r_m.
IDCom(r,m): If there exists $(r, m) \in T$, then return true else return false.
$\mathsf{IFakeCom}$(): Pick $r_f \leftarrow \{0,1\}^k$. Insert (r_f, fake) into T. Return r_f.
$\mathsf{IFakeDCom}$(r,m): If there exists $(r, \mathsf{fake}) \in T$, then insert (r, m) into T and return r, else return $\perp$.

Figure 6: The ideal tool I for a trapdoor commitment scheme.

of both players are guaranteed to be uncorrelated at the time of announcing the decommitments. For completeness, we include the relevant definitions in Appendix E. $\mathcal{P}(\mathcal{Z})$ is a replacement-friendly protocol with respect to I, because the commitment scheme is utilized as a black box.

By the definition of a secure trapdoor commitment scheme, we see that such a tool $\mathcal{Z}$ is simulation-secure with respect to the ideal tool I. Also, a standard commitment scheme $\mathcal{Y}$ with appropriate domain and range is indifferentiable from the ideal tool I; we prove this in Appendix E.1.

Lemma 6. *Let $\mathcal{Y}$ be the commitment scheme $(\mathsf{Com}, \mathsf{DCom})$. Let U_k denote the uniform distribution over $\{0,1\}^k$. Suppose $\forall_{x \in \{0,1\}^*} \mathsf{Com}(x) \sim U_k$. Suppose the domain of DCom is $\{0,1\}^k$. Then $\mathcal{Y} \sqsubset \mathcal{I}$.*

We can then conclude that $\mathcal{Y} = (\mathsf{Com}, \mathsf{DCom})$ is a workalike of $\mathcal{Z} = (\mathsf{TCGen}, \mathsf{TCom}, \mathsf{TDCom}, \mathsf{TFakeCom}, \mathsf{TFakeDCom})$ with respect to the ideal tool I. Thus, if we can translate handles of $\mathcal{Z}$ to $\mathcal{Y}$ (and vice versa), we have satisfied all conditions of Theorem 5, and can apply it to prove the security of $\mathcal{P}(\mathcal{Y})$ in our translator framework. We note that the resulting protocol does not use the CRS. As a result, we can remove the CRS without penalty; a proof appears in Appendix C.3.

Theorem 7. *Let the translator T' correctly translate handles of $\mathcal{Y}$ to handles of $\mathcal{Z}$. Let the translator T correctly translate handles of $\mathcal{Z}$ to handles of $\mathcal{Y}$. The protocol $\mathcal{P}(\mathcal{Y})$ is simulatable and provides mutually independent announcements against all allowed adversaries.*

Proof. In this section, we have proved that all conditions for applying our technique of secure substitution apply: $\mathcal{P}(\mathcal{Z})$ is a replacement-friendly protocol that is simulatable and provides mutually independent announcements against all allowed adversaries, $\mathcal{Z}$ is simulation-secure with respect to I, $\mathcal{P}$(I) is secure against all allowed adversaries, and $\mathcal{Y} \sqsubset$ I for all players. Thus, by application of Theorem 5, $\mathcal{P}(\mathcal{Y})$ is also simulatable and provides mutually indpendent announcements against all allowed adversaries. □

8 Non-Interactive, Non-Malleable Commitment Without CRS

As an example application of our techniques for secure substitution of cryptographic tools, we construct a novel non-interactive, non-malleable commitment scheme without CRS from a scheme that utilizes a CRS. Currently known protocols for this task require either a CRS or a constant number of rounds of communication [3, 26].

Intuitively, a commitment is *non-malleable* if no adversary can, after seeing a commitment, construct a commitment that depends in a 'non-trivial' way on the committed value. For example, if an adversary can construct a commitment to $1-b$ given a commitment to b, the scheme is malleable. Non-malleability is a stronger property than the classical hiding property of a commitment scheme because an adversary may see the decommitment key for a target value before making its own decommitment.

Definition 10. *Let $\mathcal{D}$ be an efficiently sampleable distribution and let $\mathcal{R}$ be a PPT-computable relation. Consider the two security experiments of Figure 7. A commitment scheme is* **non-malleable with**

Experiment $\mathbf{Exp}_{a-priori}(\Gamma', \mathcal{R})$
1. $s \leftarrow_R \mathcal{D}$
2. $t \leftarrow \Gamma'(\mathcal{D})$

$\mathbf{Adv}_{a-priori}(\Gamma') = \Pr[\mathcal{R}(s,t) = 1]$

Experiment $\mathbf{Exp}_{a-post}(\Gamma, \mathcal{R})$
1. $\sigma \leftarrow \{0,1\}^{n^a}$
2. $s \rightarrow_R \mathcal{D}$
3. $(com_1, dec_1) \leftarrow \mathsf{Com}(\sigma, s)$
4. $com_2 \leftarrow \Gamma(\sigma, com_1, dec_1)$
5. $dec_2 \leftarrow \Gamma(\sigma, com_1, com_2, dec_1)$

$\mathbf{Adv}_{a-post}(\Gamma) = \Pr[\mathsf{DCom}(\sigma, com_1, dec_1) = s \wedge$
$\mathsf{DCom}(\sigma, com_2, dec_2) = t \wedge com_1 \neq com_2 \wedge \mathcal{R}(s,t) = 1]$

Figure 7: The "a priori" and "a posteriori" experiments for non-malleability with respect to opening of a commitment scheme $(\mathsf{Com}, \mathsf{DCom})$. Note that the common reference string is n^a bits long, for some constant a.

respect to opening *if for each PPT adversary* Γ*, there exists an adversary simulator* Γ' *such that* $\mathbf{Adv}_{a-post}(\Gamma, \mathcal{R}) - \mathbf{Adv}_{a-priori}(\Gamma', \mathcal{R})$ *is negligible.*

We begin with a protocol for non-interactive, non-malleable commitment due to Di Crescenzo et al. [15]. (For clarity, we have included this protocol in Appendix D.) This scheme makes use of both *equivocable* and standard commitments. An equivocable commitment is a special commitment scheme in the common reference string model in which a simulator can 'cheat' during the proof. The faking algorithm M outputs a commitment value com, two decommitment values dec_0, dec_1, and a random string σ'. When σ' is used as the common reference string, either decommitment dec_i may be used to decommit the value com to the bit i. Because a simulator in the common reference string model may set the CRS, the simulator can 'cheat'[1]. Note that this cheating functionality is unavailable during normal execution of the protocol. To apply our techniques, we must: (1) identify a suitable ideal tool, (2) verify that the protocol is replacement-friendly, (3) apply our security theorem, (4) remove the CRS as in Section 7.

At first glance, we might simply adapt the ideal tool for trapdoor commitments of Figure 6. However, the protocol requires players to compute a 'tag' value as a function of a commitment handle. Instead, we integrate this tag functionality to obtain the *equivocal commit-with-MAC* tool I in Figure 8. The protocol is replacement-friendly with regard to this ideal tool.

This choice of ideal tool introduces a complication, as we have changed the interfaces to the cryptographic tool. For players in the real world, the changes are simply in interface names. For the simulator, we have introduced an additional parameter to $\mathsf{IFakeCom}$: a 'hint' to enable correct translation. Luckily, the original protocol's simulator and proof of security [15] can be slightly changed to accommodate this. We defer discussion of this transformation to Appendix D.

We denote as $\mathcal{Z}$ the cryptographic tool composed of the $ax + b$ MAC together with the equivocable commitment, while the tool $\mathcal{Y}$ refers to the same MAC, but composed with a standard commitment. There is a well-defined mapping between commitments of $\mathcal{Z}$ and $\mathcal{Y}$; the simulator will only open a fake commitment according to the hint given to $\mathsf{IFakeCom}$, so these commitments may be treated just as ICom commitments. Note also that $\mathcal{Y}$ is a workalike of $\mathcal{Z}$ with regards to I; we may thus apply Theorem 5 to prove the non-malleable commitment protocol using $\mathcal{Y}$ is secure.

Theorem 8. *Let the translator* T' *correctly translate handles of* $\mathcal{Y}$ *to handles of* $\mathcal{Z}$*. Let the translator* T *correctly translate handles of* $\mathcal{Z}$ *to handles of* $\mathcal{Y}$*. Then the protocol* $\mathcal{P}(\mathcal{Y})$ *is a non-malleable bit*

[1] We note that this commitment protocol is not 'reusable', meaning that the CRS may be used for only one commitment. More recent work, e.g. [31], removes this limitation. Nevertheless, we believe this commitment is a useful example.

Tool state: a table $T = \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^k$.
$\mathsf{ICom}(m, K_{mac})$: Pick $r_m \leftarrow \{0,1\}^k$. Insert (r_m, K_{mac}, m) into T. Return r_m.
$\mathsf{IDCom}(r, K_{mac}, m)$: If there exists (r, K'_{mac}, m) or $(r, K'_{mac}, (\mathsf{fake}, m))$ in T such that $K_{mac} = K'_{mac}$,
then return true
else return false.
$\mathsf{IFakeCom}(K_{mac}, hint)$: Pick $r_f \leftarrow \{0,1\}^k$. Insert $(r_f, K_{mac}, (\mathsf{fake}, hint))$ into T. Return r_f.
$\mathsf{IFakeDCom}(r, K_{mac}, m)$: If there exists $(r, K_{mac}, (\mathsf{fake}, \cdot)) \in T$,
then remove $(r, K_{mac}, (\mathsf{fake}, \cdot))$, insert (r, K_{mac}, m) into T and return r,
else return $\perp$.

Figure 8: The ideal tool I for equivocable commit-with-MAC. Here *hint* is a single bit indicating to $\mathsf{IFakeCom}$ how the commitment might be opened.

commitment scheme.

We prove this theorem in Appendix D. As in Section 7, we can thus securely remove the CRS from the resulting protocol.

9 Conclusion and Future Work

We have taken a first step toward justifying the security of "practical" protocols by comparing them to protocols with a simulation proof of security. Several open problems present themselves. First, we believe our requirements for replacement-friendly protocols are overly restrictive and may be relaxed. Second, we would like to understand the interaction between translators and generic protocol composition. Finally, finding pairs of cryptographic tools that admit constructions of translators without additional assumptions would allow us to extend our results to the standard model, or alternatively we might hope to show that no such translators exist.

References

[1] G. Ateniese and S. Hohenberger. Proxy re-signatures: New definitions, algorithms, and applications. In *ACM CCS*, 2005.

[2] B. Barak. How to go beyond the black-box zk barrier. In *IEEE Symposium on Foundations of Computer Science*, 2001.

[3] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *Foundations of Computer Science (FOCS)*, 2002.

[4] Boaz Barak. *Non-Black-Box Techniques in Cryptography.* PhD thesis, Weizmann Institute of Science, January 2004.

[5] D. Beaver. Adaptive zero knowledge and computational equivocation. In *Symposium on the Theory of Computing STOC*, 1996.

[6] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.

[7] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of Eurocrypt*, volume 1403, pages 127–144, 1998.

[8] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Symposium on the Theory of Computing STOC*, 1988.

[9] R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO*, 2001.

[10] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`.

[11] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[12] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. In *EUROCRYPT*, 2001.

[13] I. Damgard. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, 2000. `http://www.daimi.au.dk/~ivan/papers/concurrent.ps`.

[14] Ivan Damgard. Towards practical public-key cryptosystems provably-secure against chosen-ciphertext attacks. In *Proceedings of CRYPTO*, 1991.

[15] G. DiCrescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Symposium on the Theory of Computing (STOC)*, 1998.

[16] Y. Dodis and A. Ivan. Proxy cryptography revisited. In *Proceedings of NDSS*, February 2003.

[17] U. Feige, D. Lapidor, and A. Shamir. Multiple non-interactive zero-knowledge proofs based on a single random string. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 308–317, 1990.

[18] R. Gennaro. Multi-trapdoor commitments and their applications. Cryptology ePrint Archive, Report 2003/214, 2003. `http://eprint.iacr.org/2003/214/`.

[19] Oded Goldreich. The foundations of cryptography – volume 2. http://www.wisdom.weizmann.ac.il/ oded/foc-vol2.html.

[20] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In *Proceedings of CRYPTO*, 1998.

[21] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. http://eprint.iacr.org/1999/009/, 1999. Final version of "On the Existence of 3-Round Zero-Knowledge Protocols".

[22] M. Liskov, A. Lysyanskaya, S. Micali, L. Reyzin, and A. Smith. Mutually independent commitments. In *ASIACRYPT*, 2001.

[23] Ueli Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Proc. of Theory of Cryptography Conference*, 2004.

[24] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[25] M. Naor. Bit commitment using pseudo-randomness. *J. Cryptology*, 4(2):151–158, 1991.

[26] R. Pass and A. Rosen. Concurrent non-malleable commitments. In *Foundations of Computer Science (FOCS)*, 2005.

[27] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Proceedings of CRYPTO*, volume 576, pages 129–140. Springer-Verlag, 1991.

[28] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2000.

[29] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 242–251, New York, NY, USA, 2004. ACM Press.

[30] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 242–251, New York, NY, USA, 2004. ACM Press.

[31] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *Symposium on the Theory of Computing (STOC)*, pages 426–437, 2003.

[32] I. Damgård and J. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO*, 2001.

Acknowledgments: We would like to thank Dawn Song for her valuable insights and advice. We thank Ivan Damgård, Nick Hopper, Shabsi Walfish, David Wagner, Hoeteck Wee, and Matt Lepinski for invaluable comments. We thank Chris Colohan for providing a laptop and for hospitality.

A Notation

- a, b - scenarios used in the general proof of security (see Figure 4)
- $[x]$ - the set $\{1, \ldots, n\}$
- $\mathrm{Dom}(f)$ - domain of function f
- $\mathrm{Range}(f)$ - range of function f
- $\mathrm{neg}(\cdot)$ - a negligible function
- $x \sim y$ - x is distributed indistinguishably from y, given some specified information
- $\binom{S}{x}$ - all subsets of S of size x
- $\mathcal{Y}$ - an efficient cryptographic tool, a workalike of $\mathcal{Z}$
- $\mathcal{Z}$ - an expensive cryptographic tool. $\mathcal{Y}$ is a workalike
- I - an ideal functionality for both $\mathcal{Y}$ and $\mathcal{Z}$
- $\mathcal{P}(\mathcal{Z})$ - a replacement-friendly protocol
- T' - translator from handles of $\mathcal{Y}$ to handles of $\mathcal{Z}$
- T - translator from handles of $\mathcal{Z}$ to handles of $\mathcal{Y}$
- z - an input to a cryptographic tool
- D_1, D_2 - probability distributions over some domain
- h - a handle
- $h_1 \sim_\nu h_2$ - h_1 is indistinguishably translation-indistinguishable from h_2 if the handles were constructed from indistinguishable data

- $\nu(\cdot)$ - the handle portion of the argument data
- $\bar{\nu}(\cdot)$ - the non-handle portion of the argument data
- $\mathcal{R}(\cdot)$ - non-handle references to the handles contained in the argument data, utilized in the proof of Theorem 5
- $\mathcal{CD}(h)$ - the data used to create handle h
- $\mathcal{A}$ - a generic oracle, indistinguishable from $\mathcal{B}$
- $\mathcal{B}$ - a generic oracle, indistinguishable from $\mathcal{A}$
- Γ - a malicious PPT adversary
- Υ - honest players
- X - a non-black-box extractor
- $\mathcal{D}$ - distinguisher for indifferentiability
- $\mathcal{S}$ - sanitizer for indifferentiability
- f - generic function
- $x, x_1, \ldots, x_\ell$ - generic function parameters
- $y, y_1, \ldots, y_\ell$ - generic function parameters, distributed computationally indistinguishably from $x, x_1, \ldots, x_\ell$
- $w_1, \ldots, w_\ell$ - generic function parameters, $w_j \in \{x_j, y_j\}$ $(1 \leq j \leq \ell)$
- r - random function parameter
- ℓ - number of function parameters to a generic function
- j - index over function parameters or sets
- i - index over interfaces
- ℓ - index over protocol steps
- F - number of interfaces of ideal tool I
- $\mathcal{H}$ - the set of interfaces that produce handles for some pair of workalikes $\mathcal{Y}$, $\mathcal{Z}$, with respect to ideal tool I; $\mathcal{H} \subseteq F$
- L - a set of indicies into another set
- z - number of steps in functional representation of protocol
- $o_0, o_z, o'_0, \ldots, o'_{z-1}$ - output of honest and malicious players in functional representation of protocol
- $s_0, \ldots, s_z, s'_0, \ldots, s'_{z-1}$ - internal state of honest and malicious players in functional representation of protocol
- $(\mathsf{Com}, \mathsf{DCom})$ - a standard commitment scheme
- $(\mathsf{TCGen}, \mathsf{TCom}, \mathsf{TDCom}, \mathsf{TFakeCom}, \mathsf{TFakeDCom})$ - a trapdoor commitment scheme
- $(\mathsf{TPk}, \mathsf{TSk})$ - public and secret key for trapdoor commitment scheme
- $(\mathsf{ICom}, \mathsf{IDCom}, \mathsf{IFakeCom}, \mathsf{IFakeDCom})$ - an ideal commitment scheme
- $\mathcal{D}$ - distribution given as part of definition of non-malleability
- $\mathcal{R}$ - probabilistic polynomial time "relation approximator"
- (A, B) - a standard commitment scheme used in the DIO protocol
- (C, D) - an equivocable commitment scheme used in the DIO protocol
- ACom - commitment value in DIO protocol
- CCom - commitment value in DIO protocol
- ADCom - decommitment value in DIO protocol
- CDCom - decommitment value in DIO protocol
- NMCom - output of DIO protocol
- NMDCom - output of DIO protocol
- QCom - part of output of Algorithm Q in DIO proof
- QDCom - part of output of Algorithm Q in DIO proof
- I_T - an ideal cryptographic tool for the ideal commitment scheme
- $< P, V > (x)$ - interaction of prover and verifier on input x in interactive proof

- (P_a, V_a) - prover and verifier in 3-round honest-verifier zero-knowledge protocol
- (P_b, V_b) - prover and verifier in Damgard modification of 3-round honest-verifier zero-knowledge protocol
- (P_c, V_c) - prover and verifier in Damgard modified protocol with trapdoor commitment replaced by standard commitment
- (P_d, V_d) - prover and verifier in Damgard modified protocol with trapdoor commitment replaced by standard commitment and common reference string removed

B Proof of Secure Replacement

In order to complete our proofs, we prove several lemmas in Appendix B.1. We then give a general proof of secure translation, given abstract translators T' and T.

B.1 Lemmas

Lemma 9. *For all functions f computable by an allowed adversary, $x \sim y \rightarrow f(x) \sim f(y)$.*

Proof. If $f(x)$ is distinguishable from $f(y)$ by an allowed adversary, then there exists an adversary who can distinguish x and y, through calculation of f. By the definition of indistinguishability, there exists no such adversary. Thus, by contradiction, $f(x)$ is indistinguishable from $f(y)$. □

Corollary 10. *Lemma 9 holds for any number of function parameters ℓ. Formally, if each parameter $x_j \sim y_j$ $(1 \leq j \leq \ell)$, then for all $\{w_1, \ldots, w_\ell \mid w_j \in \{x_j, y_j\}\}$, $f(w_1, \ldots, w_\ell)$ are mutually computationally indistinguishable.*

Corollary 11. *Let $\mathcal{A}$ and $\mathcal{B}$ be black-box subroutines such that $\forall_{x \in \{0,1\}^*} \mathcal{A}(x) \sim \mathcal{B}(x)$. Lemma 9 also holds for functions f which receive access to either $\mathcal{A}$ or $\mathcal{B}$. Formally, for all functions f, $x \sim y \rightarrow f^{\mathcal{A}}(x) \sim f^{\mathcal{B}}(x) \sim f^{\mathcal{A}}(y) \sim f^{\mathcal{B}}(y)$. Similarly to Lemma 10, this lemma may be extended to allow for any number of function parameters.*

B.2 General proof of security.

We now prove a general theorem of security against allowed adversaries for substitution of the workalike cryptographic tool $\mathcal{Y}$ for the tool $\mathcal{Z}$. This theorem requires generic correct translators from $\mathcal{Y}$ to $\mathcal{Z}$ and vice versa. To ensure security, we prove an indistinguishability condition is preserved through execution of the generic replacement-friendly protocol $\mathcal{P}(\cdot)$. We denote as $\mathcal{P}(\mathcal{Z})$ (resp. $\mathcal{P}(\mathcal{Y})$) the protocol using the tool $\mathcal{Z}$ (resp. $\mathcal{Y}$).

Theorem 12. *Let $\mathcal{Y}$ be a workalike of $\mathcal{Z}$ with respect to ideal tool I. Let $\mathcal{P}(\mathcal{Z})$ be a replacement-friendly protocol. Let the translator T' ($T' = \{T'_{\mathcal{H}_1}, \ldots, T'_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T'_i : Range(\mathcal{Y}_i) \rightarrow Range(\mathcal{Z}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Y}$ to $\mathcal{Z}$. Let the translator T ($T = \{T_{\mathcal{H}_1}, \ldots, T_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T_i : Range(\mathcal{Z}_i) \rightarrow Range(\mathcal{Y}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Z}$ to $\mathcal{Y}$.*
Any allowed adversary Γ utilizing $\mathcal{P}(\mathcal{Y})$ cannot distinguish between a scenario in which he interacts with Υ utilizing $\mathcal{P}(\mathcal{Y})$, and a scenario in which he interacts (through T and T') with Υ utilizing $\mathcal{P}(\mathcal{Z})$, as shown in Figure 9.

Proof. We prove that any allowed adversary Γ cannot distinguish between scenarios (a) and (b) of Figure 9 by proving that the view of Γ is indistinguishable between these scenarios. To formalize the views of Γ and Υ, we give a general *functional decomposition* of a generic protocol in Figure 10. The operation of

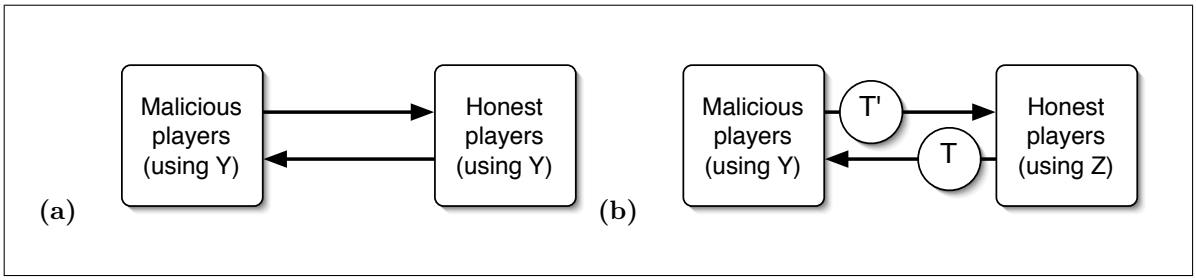


Figure 9: To prove security, we prove that Γ cannot distinguish between scenarios: (a) the normal operation of the protocol using $\mathcal{Y}$; (b) translations of handles to and from the honest players, where Υ utilizes $\mathcal{Z}$.

Protocol Operation:

$$
\begin{array}{lll}
\Gamma & o_0 \rightarrow & \\
 & \leftarrow o'_0 & \Upsilon \\
\Gamma & o_1 \rightarrow & \\
 & \vdots & \\
 & \leftarrow o'_{z-1} & \Upsilon \\
\Gamma & o_z \rightarrow &
\end{array}
$$

Scenario (a):

Malicious Players	**Honest Players ($\mathcal{Y}$)**
$o^a_0, s^a_0 \leftarrow \Gamma_0(\text{initial inputs})$	
	$o'^a_0, s'^a_0 \leftarrow \Upsilon^{\mathcal{Y}}_0(\text{initial inputs}, o^a_0)$
$o^a_1, s^a_1 \leftarrow \Gamma_1(s^a_0, o'^a_0)$	
$\vdots$	$\vdots$
	$o'^a_{z-1}, s'^a_{z-1} \leftarrow \Upsilon^{\mathcal{Y}}_{z-1}(s'^a_{z-2}, o^a_{z-1})$
$o^a_z, s^a_z \leftarrow \Gamma_z(s^a_{z-1}, o'^a_{z-1})$	

Scenario (b):

Malicious Players	**Honest Players ($\mathcal{Z}$)**
$o^c_0, s^c_0 \leftarrow \Gamma_0(\text{initial inputs})$	
	$o'^c_0, s'^c_0 \leftarrow \Upsilon^{\mathcal{Z}}_0(\text{initial inputs}, T'(o^c_0))$
$o^c_1, s^c_1 \leftarrow \Gamma_1(s^c_0, T(o'^c_0))$	
$\vdots$	$\vdots$
	$o'^c_{z-1}, s'^c_{z-1} \leftarrow \Upsilon^{\mathcal{Z}}_{z-1}(s'^c_{z-2}, T'(o^c_{z-1}))$
$o^c_z, s^c_z \leftarrow \Gamma_z(s^c_{z-1}, T(o'^c_{z-1}))$	

Figure 10: Let the Protocol Operation show the operation of a generic protocol. The operation of malicious players Γ in such a protocol may be represented as a series of functions $\Gamma_0, \ldots, \Gamma_z$. We may thus represent the operation of the malicious and honest players in scenarios (a) and (b) of Figure 9. We call this a *functional representation* of a protocol.

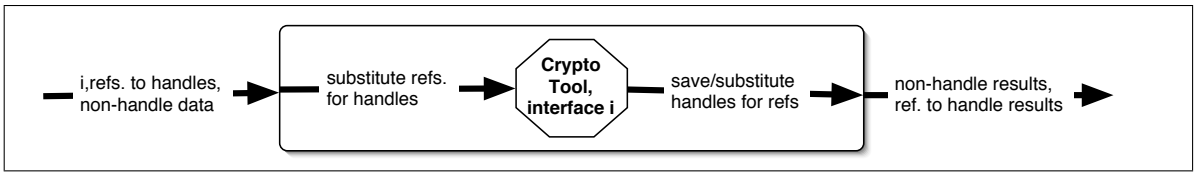


Figure 11: In our proof, Υ utilizes subroutines $\mathcal{A}$ or $\mathcal{B}$ to perform all calculation on handles. We illustrate their operation here: they take an interface identifier i, and arguments to that interface (but using handle references instead of handles); the subroutine then substitutes handles for handle references applies the cryptographic tool's ith interface, and erases all intermediate results; the subroutine then saves any new handles created (substituting handle references) and returns the results.

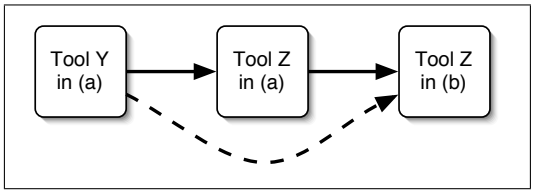


Figure 12: When examining the operation of Υ, we may note that the results of using tool $\mathcal{Y}$ in scenario (a) is indistinguishable from using tool $\mathcal{Z}$ in scenario (a). We may then observe that the result of using tool $\mathcal{Z}$ in scenario (a) is indistinguishable from using tool $\mathcal{Z}$ in scenario (b). We may thus conclude that using tool $\mathcal{Y}$ in scenario (a) gives indistinguishable results from using tool $\mathcal{Z}$ in scenario (b).

any protocol can be represented as a call-and-response style generic protocol between the adversaries Γ and honest players Υ. In this generic protocol, Γ is given the opportunity to send some data o_0 to Υ, who may respond with o'_1; Γ then responds with o_1. The protocol continues sequentially from there until the honest player makes the last move o'_{z-1} or the malicious player makes the last move o_z. Note that any real protocol may use any combination of possible communication between groups of malicious and honest adversaries; we have given a completely generic representation of any protocol $\mathcal{P}$ for purposes of our proof.

In this generic protocol, we can break the computation performed by Γ in the steps $0, \ldots, z$ into a series of functions $\Gamma_0, \ldots, \Gamma_z$. Each of these functions at step ℓ takes as input the last state of Γ, denoted $s_{\ell-1}$, and the last move of Υ, denoted $o'_{\ell-1}$, and outputs new state s_ℓ and output o_ℓ to be sent to Υ. We may also perform a similar decomposition on the operation of Υ. We describe a full functional decomposition, for both scenarios (a) and (b) in Figure 10.

Thus, given this representation, to prove that no Γ can distinguish scenarios (a) and (b), we prove that the view of Γ is indistinguishable between the two scenarios. All indistinguishability in this proof is given the previous execution visible to Γ. Formally, we must prove that: $\{s^a_0, \ldots, s^a_z, o^a_0, \ldots, o^a_z, {o'_0}^a, \ldots, {o'_{z-1}}^a\} \sim \{s^b_0, \ldots, s^b_z, o^b_0, \ldots, o^b_z, T({o'_0}^b), \ldots, T({o'_{z-1}}^b)\}$.

We prove this through use of induction on the steps of the protocol. In fact, we prove a more restrictive statement. Let $\nu(\cdot)$ output the set of handles included in its argument, and $\bar{\nu}(\cdot)$ output all non-handle data in its argument. We prove that $\{s^a_0, \ldots, s^a_z, o^a_0, \ldots, o^a_z, {o'_0}^a, \ldots, {o'_{z-1}}^a, \bar{\nu}({s'_0}^a), \ldots, \bar{\nu}({s'_{z-1}}^a)\} \sim \{s^b_0, \ldots, s^b_z, o^b_0, \ldots, o^b_z, T({o'_0}^b), \ldots, T({o'_{z-1}}^b), \bar{\nu}({s'_0}^b), \ldots, \bar{\nu}({s'_{z-1}}^b)\}$ and $\{\nu({s'_0}^a), \ldots, \nu({s'_{z-1}}^a), \nu(o^a_0), \ldots, \nu(o^a_z), \nu({o'_0}^a), \{\nu({s'_0}^b), \ldots, \nu({s'_{z-1}}^b), \nu(o^b_0), \ldots, \nu(o^b_z), T(\nu({o'_0}^b)), \ldots, T(\nu({o'_{z-1}}^b))\}$.

Our inductive base case is that the execution of the protocols $\mathcal{P}(\mathcal{Y})$ (in scenario (a)) and $\mathcal{P}(\mathcal{Z})$ (in scenario (b)) is indistinguishable. To prove this, we may simply observe that (by Lemma 10 and the

definitions of the variables in our functional decomposition of Figure 10):

$$\begin{aligned} \text{initial inputs} &\sim \text{initial inputs} \\ \Gamma_0(\text{initial inputs}) &\sim \Gamma_0(\text{initial inputs}) \\ \{s_0^a, o_0^a\} &\sim \{s_0^b, o_0^b\} \\ \nu(o_0^a) &\sim_\nu \nu(o_0^b) \end{aligned}$$

In the remainder of the proof, we make the inductive assumption that all execution up to the current step ℓ has been indistinguishable between scenarios (a) and (b). In order to apply the strong inductive principle, we must now prove that the next step of execution is also indistinguishable between scenarios (a) and (b). To do this, we divide the proof into two possible cases: one in which Υ last sent data to Γ, and one in which Γ last sent data to Υ.

Honest mover. We now prove that $\{s_0^a, \ldots, s_\ell^a, o_0^a, \ldots, o_\ell^a, {o'_0}^a, \ldots, {o'_\ell}^a, \bar{\nu}({s'_0}^a), \ldots, \bar{\nu}({s'_\ell}^a)\} \sim \{s_0^b, \ldots, s_\ell^b, o_0^b, \ldots, o_\ell^b, T({o'_0}^b), \ldots, T({o'_\ell}^b), \bar{\nu}({s'_0}^b), \ldots, \bar{\nu}({s'_\ell}^b)\}$ and $\{\nu({s'_0}^a), \ldots, \nu({s'_\ell}^a), \nu(o_0^a), \ldots, \nu(o_\ell^a), \nu({o'_0}^a), \ldots, \nu({o'_\ell}^a)\} \sim_\nu \{\nu({s'_0}^b), \ldots, \nu({s'_\ell}^b), \nu(o_0^b), \ldots, \nu(o_\ell^b), T(\nu({o'_0}^b)), \ldots, T(\nu({o'_\ell}^b))\} \rightarrow \{s_{\ell+1}^a, o_{\ell+1}^a\} \sim \{s_{\ell+1}^b, o_{\ell+1}^b\}$ and $\nu(o_{\ell+1}^a) \sim_\nu \nu(o_{\ell+1}^b)$

We may prove the truth of this statement as follows (by Lemma 10, the definition of correct translation, and the definitions of the variables in our functional decomposition of Figure 10):

$$\begin{aligned} \{s_\ell^a, {o'_\ell}^a\} &\sim \{s_\ell^b, T({o'_\ell}^b)\} \\ \Gamma_{\ell+1}(s_\ell^a, {o'_\ell}^a) &\sim \Gamma_{\ell+1}(s_\ell^b, T({o'_\ell}^b)) \\ \{s_{\ell+1}^a, o_{\ell+1}^a\} &\sim \{s_{\ell+1}^b, o_{\ell+1}^b\} \\ \nu(o_{\ell+1}^a) &\sim_\nu \nu(o_{\ell+1}^b) \end{aligned}$$

Malicious mover. We now prove that $\{s_0^a, \ldots, s_\ell^a, o_0^a, \ldots, o_\ell^a, {o'_0}^a, \ldots, {o'_{\ell-1}}^a, \bar{\nu}({s'_0}^a), \ldots, \bar{\nu}({s'_{\ell-1}}^a)\} \sim \{s_0^b, \ldots, s_\ell^b, o_0^b, \ldots, o_\ell^b, T({o'_0}^b), \ldots, T({o'_{\ell-1}}^b), \bar{\nu}({s'_0}^b), \ldots, \bar{\nu}({s'_{\ell-1}}^b)\}$ and $\{\nu({s'_0}^a), \ldots, \nu({s'_{\ell-1}}^a), \nu(o_0^a), \ldots, \nu(o_\ell^a), \nu({o'_0}^a), \ldots$ $\{\nu({s'_0}^b), \ldots, \nu({s'_{\ell-1}}^b), \nu(o_0^b), \ldots, \nu(o_\ell^b), T(\nu({o'_0}^b)), \ldots, T(\nu({o'_{\ell-1}}^b))\} \rightarrow \{\bar{\nu}({s'_\ell}^a), {o'_\ell}^a\} \sim \{\bar{\nu}({s'_\ell}^b), {o'_\ell}^b\}$ and $\{\nu({o'_\ell}^a), \nu({s'_\ell}^a)\} \sim_\nu \{\nu(T({o'_\ell}^b)), \nu({s'_\ell}^a)\}$

In order to complete this section of the inductive proof, we must examine the class of functions that Υ may compute as part of a replacement-friendly protocol. The most important properties, at this juncture, are that: (1) $\mathcal{P}(\cdot)$ requires no player to compute a function of any handle; and (2) $\mathcal{P}(\mathcal{Z})$ (resp., $\mathcal{P}(\mathcal{Y})$) requires no non-black-box usage of $\mathcal{Z}$ (resp., $\mathcal{Y}$). We may observe from this that Υ does not require the handles themselves in order to calculate Υ_ℓ, if it can indirectly specify the handles needed for cryptographic computation.

We will therefore, without loss of generality, represent the execution of the function Υ_ℓ as a function f (identical in scenarios (a) and (b)) computed upon non-handle data, with access to the cryptographic tool via a subroutine. In order to specify a handle for cryptographic computation, Υ_ℓ will utilize a *reference* to the handle, denoted $\mathcal{R}(\cdot)$. These references are simply non-handle data utilized to specify a handle, such as numbers assigned in the order the handles were calculated.

Instead of allowing f to calculate black-box cryptographic results directly, we give it access to an subroutine, run internally by Υ. All workings of this subroutine, including the intermediate results of black-box computation, are erased by Υ after use. Formally, we define two subroutines, $\mathcal{A}$ and $\mathcal{B}$, used by Υ to perform all computation on handles from $\mathcal{Y}$ and $\mathcal{Z}$, respectively. These subroutines take as input an interface identifier i ($1 \leq i \leq F$), all non-handle data to be used as input to the interface $\mathcal{Y}_i$, and a list of handle references for all handles to be used as input to the cryptographic tool. The subroutine the

applies the tool, removes and saves any handles in the result (substituting a reference to the handle), and returns the result to f. We illustrate the operation of these subroutines in Figure 11.

A request from f for cryptographic tool computation by $\mathcal{A}$ (or $\mathcal{B}$) is denoted: the interface identifier i^a (or i^b) ($1 \leq i \leq F$), non-handle arguments x^a (or x^b), and a list of references to the handle arguments L^a (or L^b). By the inductive assumption:

$$\begin{aligned} \{s_\ell^a, o_\ell^a\} &\sim \{s_\ell^b, o_\ell^b\} \\ \bar{\nu}(o_\ell^a) &\sim \bar{\nu}(o_\ell^b) \\ \{\nu(o_\ell^a), \nu({s'_{\ell-1}}^a)\} &\sim_\nu \{\nu(o_\ell^b), \nu({s'_{\ell-1}}^b)\} \end{aligned}$$

Thus, by Lemma 10 for the first such call, note that the requests made of $\mathcal{A}$ (in scenario (a)) and $\mathcal{B}$ (in scenario (b)) are indistinguishable:

$$\begin{aligned} \{\bar{\nu}({s'_{\ell-1}}^a), \bar{\nu}(o_\ell{}^a), \mathcal{R}(\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\})\} &\sim \{\bar{\nu}({s'_{\ell-1}}^b), \bar{\nu}(o_\ell{}^b), \mathcal{R}(\{\nu({s'_{\ell-1}}^b), \nu(o_\ell{}^b)\})\} \\ f(\bar{\nu}({s'_{\ell-1}}^a), \bar{\nu}(o_\ell{}^a), \mathcal{R}\{\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}\})) &\sim f(\bar{\nu}({s'_{\ell-1}}^b), \bar{\nu}(o_\ell{}^b), \mathcal{R}(\{\nu({s'_{\ell-1}}^b), \nu(o_\ell{}^b)\})) \\ \{i^a, x^a, L^a\} &\sim \{i^b, x^b, L^b\} \end{aligned}$$

We may also note that, by Lemma 10, the set of handles specified by the handle references L^a or L^b, denoted $\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a}$ or $\{\nu({s'_{\ell-1}}^b), \nu(o_\ell{}^b)\}_{L^b}$, are handleset-indistinguishable:

$$\begin{aligned} L^a &\sim L^b \\ \{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\} &\sim_\nu \{\nu({s'_{\ell-1}}^b), \nu(o_\ell{}^b)\} \\ \{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a} &\sim_\nu \{\nu({s'_{\ell-1}}^b), \nu(o_\ell{}^b)\}_{L^b} \end{aligned}$$

Thus, by the correctness of translation and Lemma 10, all non-handle data computed in response to the request by f is indistinguishable, and the handles computed are handleset-indistinguishable: (we illustrate the transitivity property exploited in this section of the proof in Figure 12)

$$\begin{aligned} \bar{\nu}(\mathcal{Y}_{i^a}(x^a, \{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a})) &\sim \bar{\nu}(\mathcal{Z}_{i^a}(x^a, T'(\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a}))) \\ \nu(\mathcal{Y}_{i^a}(x^a, \{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a})) &\sim_\nu \nu(\mathcal{Z}_{i^a}(x^a, T'(\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a}))) \\ \bar{\nu}(\mathcal{Z}_{i^a}(x^a, T'(\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a}))) &\sim \bar{\nu}(\mathcal{Z}_{i^b}(\{\nu({s'_{\ell-1}}^b), \nu(x^b, o_\ell{}^b)\}_{L^b})) \\ \nu(\mathcal{Z}_{i^a}(x^a, T'(\{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a}))) &\sim_\nu \nu(\mathcal{Z}_{i^b}(\{\nu({s'_{\ell-1}}^b), \nu(x^b, o_\ell{}^b)\}_{L^b})) \\ \bar{\nu}(\mathcal{Y}_{i^a}(x^a, \{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a})) &\sim \bar{\nu}(\mathcal{Z}_{i^b}(\{\nu({s'_{\ell-1}}^b), \nu(x^b, o_\ell{}^b)\}_{L^b})) \\ \nu(\mathcal{Y}_{i^a}(x^a, \{\nu({s'_{\ell-1}}^a), \nu(o_\ell{}^a)\}_{L^a})) &\sim_\nu \nu(\mathcal{Z}_{i^b}(\{\nu({s'_{\ell-1}}^b), \nu(x^b, o_\ell{}^b)\}_{L^b})) \end{aligned}$$

We may now observe that, for the first call made to $\mathcal{A}$ or $\mathcal{B}$, $\mathcal{A} \sim \mathcal{B}$; the output of the subroutines is indistinguishable. Similarly, by following the proof given above for the first call (but using Lemma 11 to examine the subsequent operation of f), we may observe that for all calls made to $\mathcal{A}$ or $\mathcal{B}$ by f, the data returned to f is indistinguishable: $\mathcal{A} \sim \mathcal{B}$. Thus, by Lemma 11, we give the concise proof of our inductive statement:

$$\begin{aligned} \bar{\nu}({s'_{\ell-1}}^a, o_\ell^a) &\sim \bar{\nu}({s'_{\ell-1}}^b, T'(o_\ell^b)) \\ \nu({s'_{\ell-1}}^a, o_\ell^a) &\sim_\nu \nu({s'_{\ell-1}}^b, T'(o_\ell^b)) \\ \bar{\nu}(f^{\mathcal{A}}({s'_{\ell-1}}^a, o_\ell^a)) &\sim \bar{\nu}(f^{\mathcal{B}}({s'_{\ell-1}}^b, T'(o_\ell^b))) \\ \bar{\nu}(\Upsilon_\ell({s'_{\ell-1}}^a, o_\ell^a)) &\sim \bar{\nu}(\Upsilon_\ell({s'_{\ell-1}}^b, T'(o_\ell^b))) \\ \nu(\Upsilon_\ell({s'_{\ell-1}}^a, o_\ell^a)) &\sim_\nu \nu(\Upsilon_\ell({s'_{\ell-1}}^b, T'(o_\ell^b))) \\ \nu(\Upsilon_\ell({s'_{\ell-1}}^a, o_\ell^a)) &\sim T(\nu(\Upsilon_\ell({s'_{\ell-1}}^b, T'(o_\ell^b)))) \end{aligned}$$

□

Corollary 13. *Let $\mathcal{Y}$ be a workalike of $\mathcal{Z}$ with respect to ideal tool I. Let $\mathcal{P}(\mathcal{Z})$ be a replacement-friendly protocol. Let the translator T' ($T' = \{T'_{\mathcal{H}_1}, \ldots, T'_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T'_i : Range(\mathcal{Y}_i) \rightarrow Range(\mathcal{Z}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Y}$ to $\mathcal{Z}$. Let the translator T ($T = \{T_{\mathcal{H}_1}, \ldots, T_{\mathcal{H}_{|\mathcal{H}|}}\}$, $T_i : Range(\mathcal{Z}_i) \rightarrow Range(\mathcal{Y}_i)$ $(i \in \mathcal{H})$) correctly translate handles from $\mathcal{Z}$ to $\mathcal{Y}$.*
If $\mathcal{P}(\mathcal{Z})$ and $\mathcal{P}(I)$ are cryptographically secure against any allowed adversary Γ, then $\mathcal{P}(\mathcal{Y})$ is also cryptographically secure against Γ.

C Translating Pedersen Commitments

Pedersen commitments are an example of a tool where a nearly-correct translator can be constructed under standard assumptions [27]. Given the parameters q (prime), $g, g_1, g_2 \leftarrow Z_q^*$, recall that the computation of a commitment to x with opening data r is defined as

$$\begin{aligned} r &\leftarrow Z_q \\ \mathsf{Com}_{g,g_1}(x) &= g^x g_1^r \end{aligned}$$

Note that the commitment opening $\mathsf{DCom}_{g,g_1}(x, r)$ returns `true`.
We may observe the following:

$$\begin{aligned} \exists_y\ g_1 &= g_2^y \\ \mathsf{Com}_{g,g_1}(x) &= g^x g_1^r \\ &= g^x\,(g_2^y)^r \\ &= \mathsf{Com}_{g,g_2}(x) \end{aligned}$$

However, $\mathsf{DCom}_{g,g_2}(x, r)$ returns `false`, while $\mathsf{DCom}_{g,g_2}(x, yr)$ returns `true`. We must mark thus the opening data as a handle, and construct a translator function for it. Let T' be a translator for handles of $(\mathsf{Com}_{g,g_1}, \mathsf{DCom}_{g,g_1})$ to handles of $(\mathsf{Com}_{g,g_2}, \mathsf{DCom}_{g,g_2})$, and T a translator for handles of $(\mathsf{Com}_{g,g_2}, \mathsf{DCom}_{g,g_2})$ to handles of $(\mathsf{Com}_{g,g_1}, \mathsf{DCom}_{g,g_1})$. We can easily construct these translator functions under standard assumptions, given knowledge of y.

$$\begin{aligned} T'(r) &= yr \\ T(r') &= \frac{r'}{y} \end{aligned}$$

The functions T', T form a correct translation between the handles of the cryptographic tools Com_{g,g_1} and Com_{g,g_2}. While this translation does not allow us to appreciably reduce the complexity of a protocol, the example also points to the possibility of finding translators for more interesting pairs of cryptographic tools, while utilizing only standard cryptographic assumptions.

D Modifying the Simulator for Non-Malleable Commitments

For clarity, we include in Figure 13 the non-malleable commitment protocol we modify in this paper, due to [15].

The original proof employs an "Algorithm Q" that makes use of the equivocable property of the commitment scheme. We show this algorithm in Figure 14. The main idea of the proof is to first show

$\mathsf{Com}(1^n, \sigma, b)$
1 Parse σ as $\alpha_1, \ldots, \alpha_n, \beta$
2 Let $s \leftarrow_R \{0,1\}^n$. Parse the bits of s as $s_1, \ldots, s_n$
3 for $t = 1, \ldots, n$
4. Let $(\mathsf{ACom}_i, \mathsf{ADCom}_i) = A(\alpha_i, s_i)$
5.Let $\mathsf{ACom} = \mathsf{ACom}_1 \circ \cdots \circ \mathsf{ACom}_n$. Let $d_1, \ldots, d_n$ be its binary expansion.
6.Write β as $\beta_{1,0} \circ \beta_{1,1} \circ \cdots \circ \beta_{m,0} \circ \beta_{m,1}$
7.for $j = 1, \ldots, m$
8. Let $(\mathsf{CCom}_j, \mathsf{CDCom}_j) = C(\beta_{j,d_j}, b)$
9.Let $\mathsf{CCom} = \mathsf{CCom}_1 \circ \cdots \circ \mathsf{CCom}_m$
10. Let $q = 2^{|\mathsf{CCom}|}$ and $z = G(s)$
11. Parse z as $a \circ b$. Interpret a and b as elements of $GF(q)$.
12. Let $tag = a \cdot (\mathsf{CCom}) + b$ in $GF(q)$
13. Let $\mathsf{ADCom} = \mathsf{ADCom}_1 \circ \cdots \circ \mathsf{ADCom}_n$ and $\mathsf{CDCom} = \mathsf{CDCom}_1 \circ \ldots \mathsf{CDCom}_m$
14. Output $\mathsf{NMCom} = (\mathsf{ACom}, \mathsf{CCom}, tag)$ and $\mathsf{NMDCom} = (\mathsf{ADCom}, \mathsf{CDCom})$

$\mathsf{DCom}(1^n, \sigma, b, \mathsf{NMCom} = (\mathsf{ACom}, \mathsf{CCom}, tag), \mathsf{NMDCom} = (\mathsf{ADCom}, \mathsf{CDCom}))$
1.For $i = 1, \ldots, n$
2. verify that $B(\alpha_i, \mathsf{ACom}_i, \mathsf{ADCom}_i) \neq \perp$
3.For $i = 1, \ldots, n$
4. Let $s_i = B(\alpha_i, \mathsf{ACom}_i, \mathsf{ADCom}_i)$ and let $s = s_1 \circ \cdots \circ s_n$
5.Let $d_1 \circ \cdots \circ d_m$ be the binary expansion of ACom.
6.Verify that $D(\beta_{j,d_j}, \mathsf{CCom}_j, \mathsf{CDCom}_j) = b$ for $j = 1, \ldots, m$
7.Set $q = 2^{|\mathsf{CCom}|}$ and $z = G(s)$
8.Parse z as $a \circ b$. Interpret a and b as elements of $GF(q)$.
9.Verify that $tag = a \cdot (\mathsf{CCom}) + b$ in $GF(q)$
10.If all verifications pass, accept, else reject.

Figure 13: The non-interactive non-malleable commitment scheme of Di Crescenzo et al. [15]. Here σ is the CRS and b is the bit to which we wish to commit, G is a PRG, (A, B) is a standard commitment scheme in the CRS model, and (C, D) is an equivocable commitment scheme in the CRS model.

that an adversary cannot distinguish interactions with Q from interactions with the real protocol. As a result, the adversary's advantage when interacting with Q must be close to its advantage when interacting with the real protocol; see the adversary's advantage with respect to Q in Figure 15. Next, they argue that the adversary's choice of output bit is the same regardless of whether it sees dec_0 or dec_1 from Q. Finally, they argue that if the adversary succeeds with non-negligible probability when interacting with Q, then the adversary could break the non-malleability of the protocol by simply running Q on its own.

At first it seems that we could simply apply translators to interactions between the adversary and Q and retain the exact same argument. A subtlety arises, however, because Q uses the equivocation interface, which is not available in our "target" tool of non-equivocable commitments. As a result, it at first appears that translation of the proof fails. Our key insight for overcoming this problem is that while algorithm Q *can* open a commitment to either 0 or 1, in its interactions with the adversary, Q only *needs* to open the commitment in *one* way that is picked independently of the adversary's action. Therefore, we can delay translation until after the value of the commitment has been "determined." We model this

Algorithm $Q(1^n)$:
1. Let $s \leftarrow_R \{0,1\}^n$ and $s_1 \circ \cdots \circ s_n$ be its binary expansion.
2. For $i = 1, \ldots, 2n$
3. uniformly choose string α_i
4. Let $(\mathsf{ACom}_i, \mathsf{ADCom}_i) \leftarrow A(\alpha_i, s_i)$
5. Let $\mathsf{ACom} = \mathsf{ACom}_1 \circ \cdots \circ \mathsf{ACom}_n$ and $d_1 \circ \cdots \circ d_n$ be its binary expansion.
6. For $j = 1, \ldots, m$
7. Let $(\tau_j, \mathsf{CCom}_j, \mathsf{CDCom}_{j,0}, \mathsf{CDCom}_{j,1})$ be the public random string from the equivocable commitment faking algorithm, the commitment value, and decommitments for 0 and 1 respectively.
8. Set $\beta_{j,d_j} = \tau_j$ and uniformly choose $\beta_{k,1-d_j}$
9. Set $\beta = \beta_{1,0} \circ \beta_{1,1} \circ \cdots \circ \beta_{m,0}\beta_{m,1}$
10 Set $\sigma' = \alpha_1 \circ \cdots \circ \alpha_n \circ \beta$
11. Set $\mathsf{CCom} = \mathsf{CCom}_1 \circ \cdots \circ \mathsf{CCom}_m$ and $q = 2^{|\mathsf{CCom}|}$
12. Compute $G(s) = a \circ b$. Interpret a and b as elements of $GF(q)$.
13. Compute $tag' = a \cdot (\mathsf{CCom}) + b$ in $GF(q)$
14. Let $\mathsf{QCom}' = (\mathsf{ACom}, \mathsf{CCom}, tag')$
15. Let $\mathsf{ADCom} = \mathsf{ADCom}_1 \circ \cdots \circ \mathsf{ADCom}_n$
16. Let $\mathsf{CDCom}_i = \mathsf{CDCom}_{1,i} \circ \cdots \circ \mathsf{CDCom}_{m,i}$ for $i = 0, 1$
17. Output $(\sigma', \mathsf{QCom}', \mathsf{QDCom}_0, \mathsf{QDCom}_1)$

Figure 14: The algorithm Q used in the proof of non-malleability for the scheme of [15]. As before, G is a pseudo-random generator, (A, B) is a standard commitment scheme, and (C, D) is an equivocal commitment. The algorithm Q' takes an extra bit b as an argument and feeds it to the equivocal commitment scheme as a *hint*.

formally by adding a "hint" to the ideal tool, which represents the way that the commitment might be opened in the future. We then define a new algorithm Q' that acts exactly like Q, except that Q' uses the hint.

Theorem 8. *Let the translator T' correctly translate handles of $\mathcal{Y}$ to handles of $\mathcal{Z}$. Let the translator T correctly translate handles of $\mathcal{Z}$ to handles of $\mathcal{Y}$. Then the protocol $\mathcal{P}(\mathcal{Y})$ is a non-malleable bit commitment scheme.*

Proof. (Sketch) Assume for contradiction that we have an adversary Γ that breaks the non-malleability of $\mathcal{P}(\mathcal{Y})$. That is, there does not exist an adversary simulator Γ' such that $\mathbf{Adv}_{a-posteriori}(\Gamma, \mathcal{R}) - \mathbf{Adv}_{a-priori}(\Gamma', \mathcal{R})$ is negligible.

We first argue that Γ cannot distinguish between translated interactions with algorithm Q' and interactions in the experiment $\mathbf{Exp}_{a-posteriori}(\Gamma, \mathcal{R})$ for $\mathcal{P}(\mathcal{Y})$. Observe that the distribution of the view of Γ is identical in the experiments $\mathbf{Exp}_Q(\Gamma, \mathcal{R})$ and $\mathbf{Exp}_{Q'}(\Gamma, \mathcal{R})$; in both cases it sees a decommitment string, where the string is chosen via a coin flip from $\mathcal{D}$. By Theorem 4, therefore, if Γ succeeds at distinguishing Q' from $\mathcal{P}(\mathcal{Y})$, then Γ translated suceeds at distinguishing algorithm Q' from $\mathcal{P}(\mathcal{Z})$, which is a contradiction.

This tells us $\mathbf{Exp}_{Q'}(\Gamma, \mathcal{R})$ is non-negligible. Thus, by the argument of [15], the algorithm Γ' that consists of running Algorithm Q', translating and interacting with Γ succeeds in the "a priori" experiment almost as often as Γ succeeds in the "a posteriori" experiment. That is, $\mathbf{Adv}_{a-posteriori}(\Gamma, \mathcal{R}) - \mathbf{Adv}_{a-priori}(\Gamma', \mathcal{R})$ is negligible. This is a contradiction, as we assumed non-malleability of $\mathcal{P}(\mathcal{Y})$, and

Experiment $\mathbf{Exp}_{Q}(\Gamma, \mathcal{R})$:
$1.(\sigma, com, dec_0, dec_1) \leftarrow Q$
$2.com' \leftarrow \Gamma(\sigma, com)$
$3.b \leftarrow \mathcal{D}$
$4.dec' \leftarrow \Gamma(\sigma, com, com', dec_b)$
$\mathbf{Adv}_{Q}(\Gamma, \mathcal{R}) = \Pr[\mathsf{DCom}(\sigma, com, dec_b) = b \wedge$
$\mathsf{DCom}(\sigma, com', dec') = d \wedge$
$com' \neq com \wedge \mathcal{R}(b, d) = 1]$

Experiment $\mathbf{Exp}_{Q'}(\Gamma, \mathcal{R})$:
$1.b \leftarrow \mathcal{D}$
$2.(\sigma, com, dec_b) \leftarrow Q'(b)$
$3.com' \leftarrow \Gamma(\sigma, com)$
$4.dec' \leftarrow \Gamma(\sigma, com, com', dec_b)$
$\mathbf{Adv}_{Q'}(\Gamma, \mathcal{R}) = \Pr[\mathsf{DCom}(\sigma, com, dec_b) = b \wedge$
$\mathsf{DCom}(\sigma, com', dec') = d \wedge com' \neq com \wedge \mathcal{R}(b, d) = 1]$

Figure 15: The Algorithm Q uses the equivocal property of the commitment scheme to potentially open a commitment to either a 0 or a 1. The adversary, however, only ever sees one such opening, determined by a coin drawn from $\mathcal{D}$. Experiment (a) is the original security experiment of [15]. Experiment (b) uses a modified algorithm Q' that passes in a "hint" to the ideal tool.

therefore the non-existence of such a Γ'. □

E Mutually Independent Announcements

For completeness, we now give the definition due to Liskov et al. [22] of a *mutually independent announcement protocol*, as well as the proofs required for secure substitution in this protocol. As noted above, this is two-party protocol that ensures non-correlation of secret committed values provided that both parties open their commitments. Liskov et al. consider several other notions, but we choose to focus only on mutually independent announcements.

First, however, we fix some notation. A protocol (A, B) is a pair of probabilistic polynomial time interactive Turing Machines A and B. We further divide into a pair of machines (A_C, B_C) that make up the *commit stage* of the protocol, and a pair (A_R, B_R) that make up the *reveal stage* of the protocol. On each protocol run, both machines receive a security parameter 1^k. The machine A further receives a private input a and a private random tape r_A, while the machine B receives a private input b and a private random tape r_B.

Then, during a protocol run, in the commit stage the machines A_C and B_C interact and each outputs either "accept" or "reject." Then in the reveal stage, the machines A_R and B_R interact (we assume state is kept between stages). At the end of the reveal stage, the machine A_R outputs the value β, which may be a string or the special symbol "reject"; this value β is the value revealed to A by B. The machine B_R outputs the value α, which is the value revealed to B by A; this may also be a string or "reject." For convenience, we impose a consistency condition: if A_C outputs "reject" then A_R must output "reject" and similarly for B_C and B_R.

Finally, we denote the output of A_R from the interaction between A and B on inputs $(1^k, a, b, r_a, r_b)$ by $OUT_A(1^k, a, b, r_A, r_B)$. We denote the output of B_R by $OUT_B(1^k, a, b, r_A, r_B)$. When necessary, we refer to the outputs of A_C and B_C by making the appropriate substitution of subscripts. When an input is replaced by the symbol $\cdot$, we mean the probability space induced when that input is picked uniformly at random.

Definition 11. *A protocol* (A, B) *is a* mutually independent announcement protocol *if it satisfies the following properties:*

- A-completeness. *If A and B are honest, then A can commit and reveal her value successfully with*

only a negligible probability of failure. That is, for all inputs a and b and $neg(k)$ negligible, it holds that $\Pr[OUT_B(1^k, a, b, \cdot, \cdot) \neq a] = 1 - neg(k)$.

- A-soundness. *If A is honest, then for all cheating players B', the cheating player B' cannot influence which value is committed to by A. More formally, for all inputs a, b, and for all random tapes t_C, t_R, r_A, r_B, we have that if $OUT_{A_C}(1^k, a, r_A, t_C) =$ "accept" and $OUT_{B_R}(1^k, b, r_B, t_C \circ t_R) = \alpha$, then $\alpha = a$.*

- Computational A-hiding. *No cheating adversary B' interacting only with A_C can break the GM-security of A's commitments. That is, for all bit-strings a_0 and a_1 and $neg(k)$ negligible, we have that* $\Pr[v \leftarrow \{0,1\}; z \leftarrow OUT_{B'}(a_v, (a_0, a_1), \cdot, \cdot) : z = v] < \frac{1}{2} + neg(k)$.

- Perfect A-binding. *If the commit stage B_C of B outputs "accept," then the reveal stage B_R will accept only one revealed value. This value depends only on the transcript of the reveal stage, not on the private input of B.*

- A-non-correlation at opening. *The main idea of this definition is that for any polynomial-time relation R, any cheating adversary B' that engages in a protocol with A and then opens his committed value as β has no more chance of achieving $R(a, \beta)$ than a simulator that does not engage in any interaction with A at all. We explicitly require that $R(a,$ "$reject$"$) = 0$, so that forcing A to reject does not allow B' to do better than a simulator simply by rejecting always. We call polynomial-time relations that meet this requirement* allowable*; note that the identity relation is allowable, so we do not allow B' to copy A's commitment string.*

  More formally then, we require for all B', there exist a simulator S such that for all allowable R and all efficiently sampleable distributions D and $neg(k)$ negligible, the following holds: $\Pr[a \leftarrow D; \beta \leftarrow OUT_A(1^k, a, -, \cdot, \cdot) : R(a, \beta) = 1] < \Pr[a \leftarrow D; \beta \leftarrow S(1^k, D) : R(a, \beta) = 1] + neg(k)$

- *The protocol must also satisfy the versions of these properties defined analogously with respect to the party B.*

E.1 The Ideal Coin-Flipping Tool

Lemma 6. *Let $\mathcal{Y}$ be the commitment scheme* $(\mathsf{Com}, \mathsf{DCom})$. *Let U_k denote the uniform distribution over $\{0,1\}^k$. Suppose* $\forall x, \mathsf{Com}(x) \sim U_k$. *Suppose the domain of* DCom *is $\{0,1\}^k$. Then $\mathcal{Y} \sqsubset \mathcal{I}$.*

Proof. Note that the adversary has access only to the $(\mathsf{ICom}, \mathsf{IDCom})$ interfaces of I. We set the required algorithm S to be the identity function. Now we claim that no adversary can distinguish interactions with $\mathcal{Y}$ from interactions with I. To do so, notice that by construction, for all x, $\mathsf{ICom}(x) = U_k$. By our hypothesis, for all x, $\mathsf{Com}(x) \sim U_k$. Therefore, for all x, $\mathsf{Com}(x) \sim \mathsf{ICom}(x)$. To finish the proof, we note that by the correctness of DCom, for all strings $c \in \{0,1\}^k$, the behavior of $\mathsf{DCom}(c)$ is the same as the behavior of $\mathsf{ICom}(c)$. □

E.2 Removing the CRS

Let the protocol $\mathcal{P}'(\mathcal{Y})$ be identical to $\mathcal{P}(\mathcal{Y})$, except that it functions in a model without a common reference string. As $\mathcal{Y}$, unlike $\mathcal{Z}$, does not utilize the CRS, we may now prove that $\mathcal{P}'(\mathcal{Y})$ is secure.

Theorem 14. *Let the translator T' correctly translate handles of $\mathcal{Y}$ to handles of $\mathcal{Z}$. Let the translator T correctly translate handles of $\mathcal{Z}$ to handles of $\mathcal{Y}$. Then the protocol $\mathcal{P}'(\mathcal{Y})$ is simulatable and provides mutually independent announcements against all allowed adversaries.*

Proof. Proof sketch. The main idea is that $\mathcal{P}'(\mathcal{Y})$ differs from $\mathcal{P}(\mathcal{Y})$ only in the presence of the common reference string. This common reference string is generated independently of other parts of the view of an adversary against $\mathcal{P}(\mathcal{Y})$. Furthermore, we know that $\mathcal{P}(\mathcal{Y})$ has the desired security properties. We can then transform an adversary $\Gamma_{\mathcal{P}'(\mathcal{Y})}$ against $\mathcal{P}'(\mathcal{Y})$ that breaks one of the security properties into an adversary $\Gamma_{\mathcal{P}(\mathcal{Y})}$ against $\mathcal{P}(\mathcal{Y})$ by randomly generating a CRS and providing it to $\Gamma_{\mathcal{P}'(\mathcal{Y})}$. Thus, the adversary $\Gamma_{\mathcal{P}'(\mathcal{Y})}$ can compromise the security of $\mathcal{P}'(\mathcal{Y})$ with only negligible probability; $\mathcal{P}'(\mathcal{Y})$ is thus secure. □