Provably Secure Substitution of Cryptographic Tools

Lea Kissner
leak@cs.cmu.edu

David Molnar
dmolnar@cs.berkeley.edu

January 4, 2006

Abstract

With the development of provable security techniques, cryptographers have seen tension arise between what is practical and what can be proven. In particular, the use of simulation to prove security of protocols against malicious parties requires that "special powers" be available to the simulator. To allow the simulator to exploit its special powers, cryptographers modify their protocols. These modifications can complicate the protocol and increase its overhead.

In this paper, we take preliminary steps towards resolving this tension. We propose a novel framework and techniques for security proofs that leverage existing techniques while allowing the provably-secure use of more efficient cryptographic primitives. Our framework utilizes *translators* that map outputs of one cryptographic primitive to corresponding outputs of a different primitive. We describe conditions under which one cryptographic primitive may be securely substituted for another in a large class of protocols. As an example, we apply our techniques to a simulatable coin-flipping protocol.

1 Introduction

Traditionally, cryptographers had no systematic method for establishing the security of protocols; a protocol was considered 'secure' if sustained cryptanalytic efforts failed to prove it insecure. As security research progressed, however, new attacks were discovered against old protocols. Researchers then modified the protocol to avoid these attacks, cryptanalysts searched for new flaws, and the cycle repeated endlessly. The evolution of SRP password authentication protocol illustrates the elusiveness of true security in the absence of principled analysis [35, 33, 34, 5, 36].

With the development of provable security techniques such as reduction and simulation, we have the tools to break this cycle. Today, we can provide proofs of security subject to cryptographic assumptions, such as the hardness of factoring [30]. It appears to be difficult or impossible to prove the security of many reasonable protocols using these techniques. For example, certain zero-knowledge protocols cannot be proved secure using black-box techniques [2, 13]. Researchers modify their protocols in order to construct proofs of security using these standard techniques. Some of these modifications introduce additional overhead, make the protocol more complicated, or rely on cryptographic assumptions irrelevant to the original protocol. For example, several variations on the basic commitment primitive, such as trapdoor and equivocal commitments, are designed specifically to aid these modifications [24, 18, 22]. Unfortunately, these commitment schemes are less efficient and more complicated than the standard commitment schemes they replace. Because these modifications have negative repercussions, there is tension between efficiency and provable security in cryptographic protocols.

Consider the standard method for proving the security of a protocol against malicious players. In such a proof, one compares the execution of a protocol against the execution of an ideal trusted third party

(TTP) that computes the same function. A TTP takes all players' inputs, computes the appropriate output function for each player, and returns the results. In order to argue that a cryptographic protocol is 'as secure' as utilizing a TTP, a *simulator* interacts with the malicious parties through the protocol, obtains their private inputs, submits these inputs to the TTP, and returns the results. If the simulator can perform this task without detection by the malicious players, then the protocol is secure. We describe such a proof in Figure 1. The fact that the simulator can perform these tasks in the execution of a secure protocol implies that the simulator must have knowledge or powers not accessible to even malicious players during execution in the 'real world'. For example, the simulator may be able to 'rewind' players and provide them with suitably chosen inputs, or may 'open' equivocal commitments to a value of its choice. The modifications that create the tension between practicality and provable security are designed to allow the simulator to utilize its special powers; they often lead to a protocol that employs expensive, seemingly useless primitives.

We take a step towards resolving this tension by introducing a new framework for proofs of security that leverage existing simulator proofs, while removing the need for the simulator's special powers. Our framework utilizes *translation* between cryptographic primitives; translation is a formal way of expressing an intuitive equivalence between two primitives that 'behave equivalently'. We introduce novel techniques for proofs of security in our framework. These techniques allow us to construct proofs of security for a large class of protocols that cannot be proved secure using standard techniques. In this framework, we can transform adversaries against one protocol into adversaries that attack a different protocol. By comparing the behavior of these transformed adversaries against the behavior of the original adversaries, we can obtain a proof of security. Depending on the cryptographic primitives employed in the protocols, it is possible that our framework may be achieved in the standard model.

We demonstrate our techniques by applying them to a simulatable coin-flipping protocol. A simulator may manipulate the output of the simulatable coin-flipping protocol; this ability allows proofs of security of larger protocols that utilize coin-flipping. By proving the security of a more efficient coin-flipping sub-protocol in our translator framework, we conserve the security of the larger protocol while improving its efficiency.

We make the following contributions:

- We introduce a new framework for proving security in which special *translators* map outputs of one cryptographic primitive to outputs of a different cryptographic primitive. We suggest several scenarios for realizing such translators.
- We define conditions that allow us to carry out *secure substitution* of one cryptographic primitive for another in a protocol. We show that if these conditions are met, then security properties are preserved after substitution. Our security proofs allow us to re-use *existing* proofs of security for cryptographic protocols.
- We apply secure substitution to a coin-flipping protocol, greatly increasing its efficiency. We show that in our translator framework, the new protocol retains guarantees of security for protocols that incorporate the transformed protocol. We leave the specific construction of the necessary translators as an open problem.

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 *composition* of cryptographic protocols. 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 protocols.

Outline. We discuss related work in Section 2, before proceeding to preliminaries in Section 3 and definitions in Section 4. We then prove the security of tool substitution for a large class of protocols under translation assumptions in Section 5. We discuss these assumptions in Section 6. We then describe the use of secure substitution, in Section 7, as well as a specific example of its application. Finally, we conclude in Section 8 by discussing directions for future work. In addition, we provide in Appendix A a list of the notation used in this paper.

2 Related Work

We introduce a new proof technique in this work, namely the use of special translators that are used in the proof of security. These translators resemble the "imaginary angels" in the Los Angeles Network-Aware Security framework of Prabhakaran and Sahai [29]. In their framework, angels are super-polynomial entities that exist only for purposes of the proof. All parties in the protocol are given oracle access to the angels. These angels are capable of finding collisions in hash functions used as part of the protocol. The key idea of the framework is that the angels answer only certain collision queries but not others, carefully chosen to aid a simulation proof of the protocol in question. Under a cryptographic assumption about the hash function, access to the angels' output does not aid the adversary in breaking the protocol's security. They are able to show strong composition results for many natural protocols in the resulting framework.

Our work differs from Prabhakaran and Sahai in that our translators do not find hash collisions or otherwise explicitly 'break' the security of any cryptographic tool. Instead, our translators map outputs of one tool to outputs of a different tool; an encryption of x under one cryptosystem is translated to an encryption of x under another cryptosystem. Moreover, we leverage existing proofs of security to show that protocols using workalike cryptographic tools are secure; it is not required to give a wholly new proof of security for a protocol in our framework. To do so, we leverage the concept of *indifferentiability*, as introduced by Maurer et al. [23]. Indifferentiability provides a framework for reasoning about when one tool is "basically equivalent to" a different cryptographic tool. Maurer et al. introduced the concept to show negative results about the Random Oracle Model, generalizing the negative results of Canetti, Goldreich, and Halevi [7]. Here we use indifferentiability to show *positive* results about the substitution of one cryptographic tool for another.

Finally, a great deal of recent work has studied composability of cryptographic protocols, including the Los Angeles Network-Aware Security framework and Canetti's Universal Composability [29, 6]. Our present work does not discuss composability. Instead, we focus on leveraging existing proofs of security for a protocol to justify replacing an expensive cryptographic tool with an inexpensive one. Future work will focus on how our techniques affect composition of cryptographic protocols.

3 Preliminaries

Allowed malicious adversaries. A malicious adversary may behave arbitrarily. In particular, we cannot hope to prevent malicious parties from refusing to participate in the protocol, choosing arbitrary values for its private input set, or aborting the protocol prematurely.

We will frequently make use of 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. Except where noted, 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., [12]) is based on a comparison between the ideal model and a trusted third party (TTP), where a malicious party may

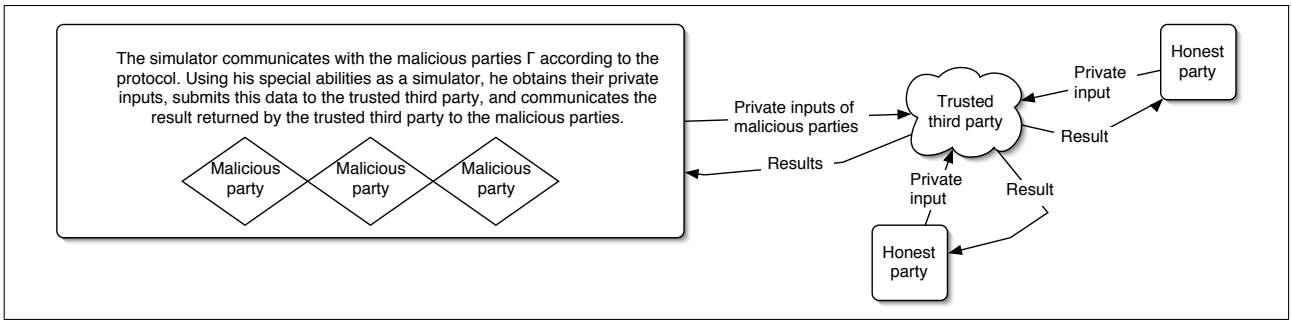


Figure 1: Basic outline of a standard simulation proof.

give arbitrary input to the TTP. The 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) $\mathcal{D}$, $|\Pr[\mathcal{D}(x_1) = 1] - \Pr[\mathcal{D}(x_2)]|$ 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 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].

Turing machines and circuits. A Turing machine is a theoretical computing device with a finite state machine attached to am infinitely long tape. The machine may write and read symbols from the current position on the tape, as well as move the tape in either direction. Turing machines form the basis of a standard model of computation [32]. Note that polynomial-sized uniform computational circuits, another standard model of computation, are essentially equivalent in computational power to polynomial-time Turing machines [27].

4 Definitions

In this section, we formally define the novel terms used in this paper. 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 $\mathcal{Y}$. To prove this result, we utilize a framework that incorporates *translators*. These translators translate *handles* of tool $\mathcal{Z}$ to handles of tool $\mathcal{Y}$, or vise versa.

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.*

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 $\mathcal{I}_1, \ldots, \mathcal{I}_F$.*

Definition 3. *A tool $\mathcal{Z}$ is* **simulation-secure with respect to** *some ideal tool I if there exists a simulator for the tool $\mathcal{Z}$ 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.*

Definition 4. *Tool $\mathcal{Y}$ is a* **workalike** *of tool $\mathcal{Z}$ for ideal tool I if:*

- *$\mathcal{Z}$is simulation-secure with respect to I*
- *$\mathcal{Y} \sqsubset I$ 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 $\mathcal{Y} \not\sqsubset \mathcal{Z}$ for most pairs of workalike cryptographic tools. For example, the Pederson commitment scheme [28] is differentiable from the Naor commitment scheme [25], as they utilize different domains for their commitments.

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**.

Definition 6. *A* **replacement-friendly** *protocol using tool $\mathcal{Z}$ is one in which:*

- *no player is required to compute a function of any handle from $\mathcal{Z}$, other than through black-box invocation of an interface of $\mathcal{Z}$*
- *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 [7], that 'detect' that they are using a certain tool and purposely break.

Definition 7. *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.*

Definition 8. *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 $\{\mathcal{CD}((\mathcal{H}_1)_1), \ldots, \mathcal{CD}((\mathcal{H}_1)_{|\mathcal{H}_1|})\} \sim \{\mathcal{CD}((\mathcal{H}_2)_1), \ldots, \mathcal{CD}((\mathcal{H}_2)_{|\mathcal{H}_2|})\}$.*

Definition 9. *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 *$\mathcal{T}$.*

5 Proof of Secure Replacement

In order to complete our proofs, we prove several lemmas in Section 5.1. We then give a general proof 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 Lemmas

Lemma 1. *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 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 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

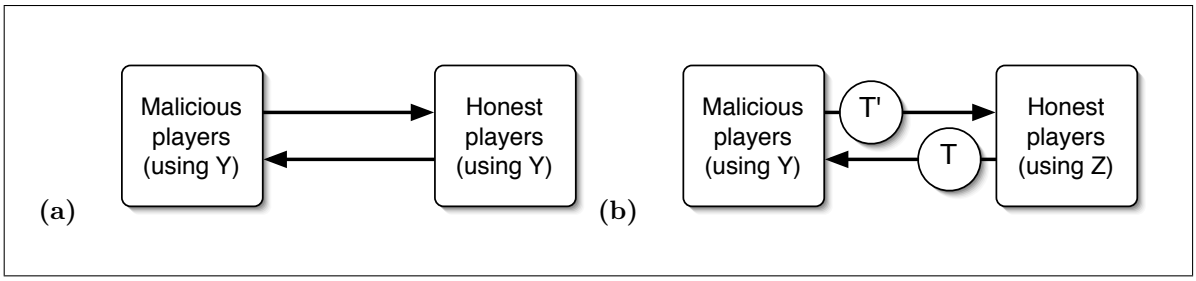


Figure 2: To prove security, we prove that Γ cannot distinguish between scenarios: (a) the normal operation of the protocol using $\mathcal{Y}$; (b) translations of handles to and from the honest players, where Υ utilizes $\mathcal{Z}$.

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 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 2.

Proof. We prove that any allowed adversary Γ cannot distinguish between scenarios (a) and (b) of Figure 2 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 3. 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 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 3.

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 $\{s^a_0, \ldots, s^a_z, o^a_0, \ldots, o^a_z, {o'_0}^a, \ldots, {o'_{z-1}}^a, \bar{\nu}({s'_0}^a), \ldots, \bar{\nu}({s'_{z-1}}^a)\} \sim$

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 3: 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 2. We call this a *functional representation* of a protocol.

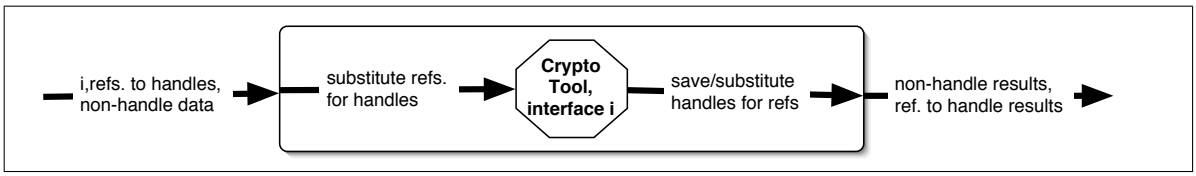


Figure 4: In our proof, Υ utilizes subroutines $\mathcal{A}$ or $\mathcal{B}$ to perform all calculation on handles. We illustrate their operation here: they take an interface identifier i, and arguments to that interface (but using handle references instead of handles); the subroutine then substitutes 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.

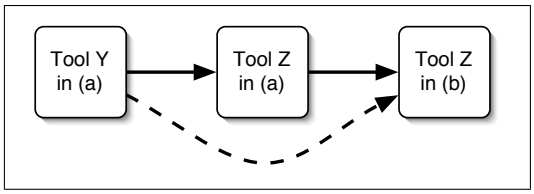


Figure 5: When examining the operation of Υ, we may note that the results of using tool $\mathcal{Y}$ in scenario (a) is indistinguishable from using tool $\mathcal{Z}$ in scenario (a). We may then observe that the result of using tool $\mathcal{Z}$ in scenario (a) is indistinguishable from using tool $\mathcal{Z}$ 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).

$\{s_0^b, \ldots, s_z^b, o_0^b, \ldots, o_z^b, T({o'_0}^b), \ldots, T({o'_{z-1}}^b), \bar{\nu}({s'_0}^b), \ldots, \bar{\nu}({s'_{z-1}}^b)\}$ and $\{\nu({s'_0}^a), \ldots, \nu({s'_{z-1}}^a), \nu(o_0^a), \ldots, \nu(o_z^a), \nu({o'_0}^a),$
$\{\nu({s'_0}^b), \ldots, \nu({s'_{z-1}}^b), \nu(o_0^b), \ldots, \nu(o_z^b), T(\nu({o'_0}^b)), \ldots, T(\nu({o'_{z-1}}^b))\}$.

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 2 and the definitions of the variables in our functional decomposition of Figure 3):

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

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 2, the definition of correct translation, and the definitions of the variables in our functional decomposition of Figure 3):

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

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 4.

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 2 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 2, 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 2, 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 5)

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

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 3 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 3, 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 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 Instantiating Translators

In the general proof of security (Theorem 4), we utilize translators to compare the security of $\mathcal{P}(\mathcal{Y})$ and $\mathcal{P}(\mathcal{Z})$. While these translators are *never* utilized during execution of the protocol, we still examine the conditions under which they can be constructed. To do so, we discuss several generic assumptions that may be utilized to construct translators between workalike cryptographic tools, as well as specific examples that point to the possibility of useful constructions in the standard model.

6.1 Generic Assumptions

The following assumptions can be applied to any pair of cryptographic tools. The plausibility of each assumption will vary, however, for each translator construction.

Non-constructive assumptions. One method for constructing a translator is to simply stipulate that one exists, without actually constructing it. For example, we may construct a non-uniform circuit that utilizes 'hints' for handle translation, without specifying any method for computation of these hints. In some scenarios, this may be an acceptable assumption, as a translator of this type does not seem to allow any sort of new attack on a protocol in a non-composed setting. We intend to examine this question in future work.

Non-black-box assumptions. Let us assume that the translator can extract the parameters to the cryptographic tool utilized to create each handle, or determine the handle is invalid. With this data, the translator may utilize the other cryptographic tool to create a translated handle. One method for this data extraction is through a non-black-box assumption. In such a scenario, the translator observes Γ calculating the output of a cryptographic tool; from these observations, the translator can determine the parameters to the tool. While these assumptions are controversial, as they are not generally efficiently falsifiable, they have been studied in several contexts [8, 15, 16, 3, 2]. Note that Υ may cooperate with the translator, making a non-black-box assumption necessary only for $\mathcal{Y}$.

Oracle assumptions. Similar to the use of non-black-box assumptions, use of oracles can allow a translator to extract the parameters used to create a handle. Unlike non-black-box assumptions, however, oracle assumptions may never be realized in the real world. As there exist protocols secure in an oracle model that may never be executed securely in the real word (e.g., [7]), a certain level of caution must be utilized when allowing oracle-based translation.

6.2 Toward Real-World Translators

We have identified several cryptographic tools in previous work that act as translators, or for which translators can be constructed under standard assumptions. We stress that these tools only allow for a restricted form of translation in the sense that we translate from one secret key for the tool to a different secret key. Nevertheless, they show that some kind of translation facility is realizable without extra assumptions.

The most intuitive of these tools is proxy re-encryption, in which a semi-trusted proxy translates ciphertexts under one key to ciphertexts under an unrelated key [4, 9]. Importantly, the proxy does not learn the plaintext being translated during this process. While a re-encryption proxy is semi-trusted, unlike a translator, it points to the possibility of translators that function in the standard model and do not extract private information. Similarly, proxy re-signatures convert signatures under one key to signatures under an unrelated key. Ateniese and Hohenberger give several constructions that translate signatures, and list translation as an intriguing open problem. For example, one might map, RSA to Schnorr signatures [1].

Pedersen commitments are an example of a tool where a nearly-correct translator can be constructed under standard assumptions [28]. 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.

7 How to Apply Secure Substitution

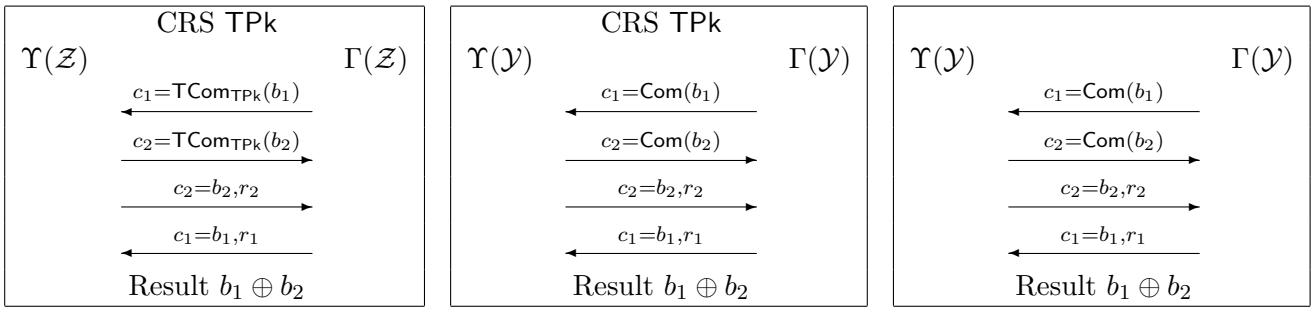
In this section, we describe the general procedure for applying secure substitution to a replacement-friendly protocol, before applying our techniques to a specific example.

7.1 An Outline of Application

For convenience in applying our techniques, we describe the process of proving the security of substitution. Let a protocol using the tool $\mathcal{Z}$ be denoted $\mathcal{P}(\mathcal{Z})$. A general process for proving the security of $\mathcal{P}(\mathcal{Y})$ is as follows:

1. Verify that $\mathcal{P}(\mathcal{Z})$ is a replacement-friendly protocol secure against all allowed adversaries.
2. Identify a suitable ideal tool I, available to all players like an oracle, such that:
   - $\mathcal{Z}$ is simulation-secure with respect to I against all allowed adversaries.
   - $\mathcal{Y} \sqsubset \mathrm{I}$ for all players.
   - $\mathcal{P}(\mathrm{I})$ is secure against all allowed adversaries.

   This is generally a simple process; the difficult parts of this proof have almost always been completed in the proof of security for $\mathcal{P}(\mathcal{Z})$. To prove security of $\mathcal{P}(\mathcal{Z})$, one generally proves the security of $\mathcal{P}(\mathrm{I})$, then proves that $\mathcal{Z}$ is simulation-secure with respect to I. Thus, in most cases, all that remains is to prove that $\mathcal{Y} \sqsubset \mathrm{I}$.
3. Construct translators T and T' that translate handles of $\mathcal{Y}$ to handles of $\mathcal{Z}$ and vice versa. We provide an outline of several methods of translator construction in Section 6.
4. Cite Theorem 4 to prove cryptographic security of $\mathcal{P}(\mathcal{Y})$ against all allowed adversaries. All necessary preconditions for this proof of standard cryptographic security have been fulfilled in steps 1-3.



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

Figure 6: Three variants of a coin-flipping protocol. In (a), a trapdoor commitment scheme which uses a common reference string (CRS) allows a simulator to replace Alice and manipulate the result of the protocol. The protocol in (b) more efficiently duplicates this functionality for all real-world players; it does not preserve the powers of the simulator. In protocol (c), we have removed the need for the common reference string needed for protocol (a).

7.2 An example: Coin-Flipping

Coin-flipping protocols are utilized as sub-protocols in a wide variety of applications [17, 19, 20, 26, 14]. In many of these uses, a simulator must dictate the outcome of the coin flip in order to prove the protocol's security [11, 14]. We apply our techniques for secure substitution to increase the efficiency of a simulatable coin flipping sub-protocol (illustrated in Figure 6(a)), while retaining the provable security of the full protocol.

Our protocol follows that of Liskov et al. [21] for "mutually independent announcements" that uses non-trapdoor commitments. In the protocol, Alice first commits to a bit b_1, followed by Bob committing to a bit b_2. Then Bob reveals his bit and randomness r_2 required to check that the bit matches his previous commitment. Finally, Alice does the same with randomness r_1. Liskov et al. show that the output bits are guaranteed to be non-correlated when the commitments are opened; as a result, the final result is a fair coin flip if both parties open. We could further simplify the protocol by having Bob simply send b_2 directly, but we keep this presentation to show the symmetry in Alice and Bob's algorithms.

We use the trapdoor property of the commitment, however, to obtain a 'cheating' interface for the simulator. By using the "fake" commitment interface of the trapdoor commitment, the simulator can obtain a "fake" commitment for Alice. Then, at reveal time, the simulator can use the trapdoor to obtain randomness for Alice's opening stage that will allow the simulator to force the final output to any desired value.

We stress that our example is intended only to be illustrative of how our techniques are applied, and that we do not give concrete constructions for the translators utilized in this section. To construct, for example, a univerally-composable coin-flipping protocol adds many complications not considered in this paper [31].

The trapdoor property of the commitment scheme utilized in the coin-flipping protocol of Figure 6(a) allows the simulator to control the result of the coin flip. (We describe the ideal version of a trapdoor commitment tool in Figure 7.) Unfortunately, in practice, trapdoor commitments require more computation or stronger assumptions than standard commitment schemes, as well as use of a common reference string (CRS) available to all players. Pedersen trapdoor commitments require a modular exponentiation and rely on a number-theoretic assumption. Feige and Shamir showed that trapdoor commitments can

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 7: The ideal tool I for a trapdoor commitment scheme.

be constructed from one-way functions, but with high overhead [10]. In contrast, the commitment scheme of Naor requires only the computation of a PRG [25]. In practice this might be instantiated using AES in counter mode, yielding an extremely efficient construction.

We address the inefficiency of trapdoor commitment schemes by applying our translator framework to provide secure substitution. Recall that in the "real world" the secret trapdoor for the commitment scheme is unknown to all players; no party can access the 'cheating' interfaces $\mathsf{TFakeCom}$ or $\mathsf{TFakeDCom}$, though these interfaces are needed for the proof of security of the protocol using coin-flipping. Therefore, we can use our techniques to substitute an efficient standard commitment scheme, which we denote $\mathcal{Y}$, for the expensive trapdoor commitment $\mathcal{Z}$ to obtain a new coin-flipping sub-protocol $\mathcal{P}(\mathcal{Y})$. We illustrate these protocols in Figure 6(a) and (b).

To prove that a protocol utilizing coin-flipping as a sub-protocol remains secure when utilizing $\mathcal{Y}$ in the place of $\mathcal{Z}$, we need only prove that the coin-flipping sub-protocol $\mathcal{P}(\mathcal{Y})$ is secure. Observe that we may consider the any player executing the large protocol as an adversary; if this adversary cannot distinguish between the results of utilizing $\mathcal{P}(\mathcal{Y})$ in place of $\mathcal{P}(\mathcal{Z})$ in the real world, then the large protocol must remain secure.

To complete this proof of secure substitution, we isolate an ideal tool I and verify that our preconditions for secure substitution hold with respect to this ideal tool. Then we utilize our proof of secure subsitution to prove the security of $\mathcal{P}(\mathcal{Y})$. As $\mathcal{P}(\mathcal{Y})$ does not utilize the common reference string (as illustrated in Figure 6(b)), we may securely remove it; we prove the security of this final modification (as illustrated in Figure 6(c)) in Appendix B.1.

7.3 Security of Substitution in Coin-Flipping

We define the ideal tool I for a trapdoor commitment scheme as a cryptographic tool with the interfaces $(\mathsf{ICom}, \mathsf{IDCom}, \mathsf{IFakeCom}, \mathsf{IFakeDCom})$ defined as in Figure 7. Consider a protocol $\mathcal{P}(\mathrm{I})$ defined as $\mathcal{P}(\mathcal{Z})$, except that all parties, including the simulator in the proof of security, utilize oracle access to I instead of the tool $\mathcal{Z}$. Note that only the simulator may access all interfaces; the players in the protocol may only utilize ICom and IDCom. We claim that the protocol $\mathcal{P}(\mathrm{I})$ also satisfies the properties of a mutually independent announcement, as defined by Liskov et al. [21], i.e. the commitments of both players are guaranteed not to be correlated with each other at the time of announcing the decommitments. For completeness, we include the relevant definitions in the Appendix.

By the definition of a secure trapdoor commitment scheme, we see that such a scheme is simulation-secure with respect to the ideal tool I. Also, a standard commitment scheme with appropriate domain and range is indifferentiable from the ideal tool I:

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 the following:

Lemma 7. $\mathcal{Y} = (\mathsf{Com}, \mathsf{DCom})$ *is a workalike of* $\mathcal{Z} = (\mathsf{TCGen}, \mathsf{TCom}, \mathsf{TDCom}, \mathsf{TFakeCom}, \mathsf{TFakeDCom})$ *with respect to the ideal tool I.*

Finally, we observe that $\mathcal{P}(\mathcal{Z})$ is a replacement-friendly protocol, because the honest player utilizes the commitment scheme as a black-box tool. We therefore conclude that if we can translate handles of $\mathcal{Z}$ to $\mathcal{Y}$ (and vice versa), we may apply Theorem 5 to prove the security of $\mathcal{P}(\mathcal{Y})$ in our translator framework.

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 $\mathcal{Y}$. 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, $\mathcal{P}(\mathrm{I})$ is secure against all allowed adversaries, and $\mathcal{Y} \sqsubset \mathrm{I}$ 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 Conclusion and Future Work

In this paper we have presented a new model for proving security of a protocol against malicious parties. We have introduced new proof techniques to this area; we believe they will be applicable to a wide range of cryptographic problems. We then applied this proof technique to a coin-flipping protocol, greatly increasing its efficiency while removing the need for a common reference string. Our work opens up several avenues of exploration that we believe will be fruitful:

- In this work, we require several characteristics of a replacement-friendly protocol. We believe that these requirements are overly restrictive, and that a much larger class of protocols can be proved secure using the techniques we introduce in this paper. Determining a minimal set of requirements, or a less-restrictive set of requirements, remains an open problem.
- When we prove security of the protocol $\mathcal{P}(\mathcal{Y})$ in Theorem 5, we concentrate only on the standard definition of security against malicious players (i.e., [12]). There is nothing in this proof, however, that depends on the definition of security; it simply proves indistinguishability of the state of Γ. This implies that different notions of security, as well as other interesting properties, may be conserved under this proof. Finding useful properties preserved between $\mathcal{P}(\mathcal{Z})$ and $\mathcal{P}(\mathcal{Y})$ by our proof of security remains an open problem.
- In this paper, we have outlined the characteristics required by a translator. It remains an open problem, however, to find useful pairs of workalike tools (and useful protocols in which they may be utilized) for which translators may be constructed under plausible cryptographic assumptions. In addition, we have not sufficiently explored the security implications of utilizing non-constructive translator assumptions.

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

References

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

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

[3] 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.

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

[5] Daniel Bleichenbacher. An attack against srp based on a flaws in the primality test. GNU Crypto mailing list, January 2004.

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

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

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

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

[10] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 526–544, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[11] Juan A. Garay, Philip MacKenzie, and Ke Yang. Efficient and secure multi-party computation with faulty majority and complete fairness. Cryptology ePrint Archive, Report 2004/009, 2004. `http://eprint.iacr.org/2004/009/`.

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

[13] Oded Goldreich and Hugo Krawczyk. On the composition of Zero-Knowledge Proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[14] J. Groth. Cryptography in subgroups of z_n^*. In *TCC*, 2005. `http://www.brics.dk/~jg/TCC05Subgroups2.pdf`.

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

[16] 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".

[17] R. Johnson and J. Staddon. FAIR: Fair audience inference. In *ACM Digital Rights Management*, 2002.

[18] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Proc. of Crypto*. Springer-Verlag, 2004.

[19] A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions, and anonymity from trapdoor-holders, 2004. `eprint.iacr.org/2004/076.pdf`.

[20] Richard J. Lipton and Rafail Ostrovsky. Micro-payments via efficient coin-flipping. In *Proceedings of Second Financial Cryptography Conference*, February 1998. `http://www.argreenhouse.com/papers/rafail/33.pdf`.

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

[22] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400. Springer, 2004.

[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] A. Nicolosi, M. Krohn, Y. Dodis, and D. Mazieres. Proactive two-party signatures for user authentication, 2003.

[27] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1995.

[28] 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.

[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] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report 212, M.I.T. Laboratory for Computer Science, 1979.

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

[32] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[33] Version 2 of the secure remote password protocol.

[34] Version 3 of the secure remote password protocol.

[35] T. Wu. The secure remote password protocol. In *Proceedings of NDSS*, pages 97–111, March 1998.

[36] T. Wu. Srp-6: Improvements and refinements to the secure remote password protocol. Submission to the IEEE P1363 Working Group, October 2002.

A Notation

- a, b - scenarios used in the general proof of security (see Figure 3)
- $[x]$ - 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
- G - simulator party from Figure 1
- $\mathcal{D}$ - 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
- 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 Proofs

In this section, we give the proofs for several lemmas not included in the body of the paper.

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 $\mathsf{ICom}(c)$. □

B.1 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 9. *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. □

C Mutually Independent Announcements

For completeness, we now give the definition due to Liskov et al. [21] of a *mutually independent announcement 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 10. *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* $R(a, \text{"}reject\text{"}) = 0$*, 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.