Foundations of Secure Interactive Computing

Donald Beaver *
AT&T Bell Laboratories

Abstract.

The problem of secure multiparty computation is usually described as follows: each of n players in a network holds a private input x_i. Together they would like to compute a function $F(x_1, \ldots, x_n)$ without revealing the inputs, even though no particular player can be trusted. Attempts to contrive formal definitions for the problem have treated properties of the solution separately (correctness, privacy, *etc.*), giving an *ad hoc* collection of desirable properties and varied definitions that do not support clear or comparable proofs.

We propose a clear, concise, and unified definition for security and reliability in interactive computations. We develop a reduction called *relative resilience* that captures all desired properties at a single blow. Relative resilience allows one to classify and compare arbitrary protocols in terms of security and reliability, in the same way that Turing reductions allow one to classify and compare algorithms in terms of complexity. Security and reliability reduce to a simple statement: a protocol for F is *resilient* if it is as resilient as an *ideal* protocol in which a trusted host *is* available to compute F. Relative resilience captures the notions of security and reliability for a wide variety of interactive computations, including zero-knowledge proof systems, Byzantine Agreement, oblivious transfer, two-party oblivious circuit evaluation, among others.

Relative resilience provides modular proof techniques that other approaches lack: one may compare a sequence of protocols ranging from the real-world protocol to the ideal protocol, proving the relative resilience of each successive protocol with greater clarity and less complexity. Folk theorems about the "transitivity" of security and the security of concatenated protocols are now provable; and the proofs reveal that such folk theorems fail under subtle conditions that have previously gone unnoticed. The conciseness[1] and modularity of our definitions and proof techniques provide great clarity in designing and reasoning about protocols and have already lead to provably secure protocols that are significantly more efficient than those appearing in the literature.

*This research was supported in part under NSF grant CCR-870-4513 at Harvard University, and by an AT&T Bell Laboratories postdoctoral fellowship. Contact: Donald Beaver, 313 Whitmore, Penn State Univ., State College, PA 16802; (814) 863-0147; beaver@cs.psu.edu.

[1] We have developed our definitions with great care and precision, and we believe them well-suited to culling a meaningful 15-page abstract. A full version is available on request (see also [2, 4]).

J. Feigenbaum (Ed.): Advances in Cryptology - CRYPTO '91, LNCS 576, pp. 377-391, 1992.

1 Introduction

The purpose of cryptographic research, especially in the theoretical domain, is threefold:

1. to develop new techniques for secure communication and computation;
2. to investigate and improve efficiency;
3. to provide proofs of security.

Whether it address secure communication, reliable file storage, operating system security, or computations performed on private data, cryptographic research must provide clear and provable results. Without clarity, efficiency and implementation are impossible; without proofs, as a long history of broken systems shows, no system can be relied upon without fear.

We investigate and define the nature of security for interactive computations, in which results need to be computed based on values supplied by two or more participants. The results must satisfy an intuitive idea of correctness, and often are required to preserve the privacy of certain information (*eg.* the input values).

Typically, an interactive computation is described as a *multiparty protocol*: n players, each holding a private input x_i, wish to compute some function $F(x_1, \ldots, x_n)$ without revealing anything about the inputs than what is computable solely from learning $F(x_1, \ldots, x_n)$. The results should be robust against an adversary who attacks the protocol, gaining information from some players and perhaps even causing faulty messages to be sent. A well-known example is the *secret ballot*, in which the participants compute the sum of their votes without revealing how any individual voted. An adversary might choose or learn some of the votes, but it should not learn anything more than the overall tally and the choices of the players it corrupts.

A wide variety of network models (private channels, broadcast, oblivious transfer), computational complexity assumptions, and adversarial powers is examined in the literature [27, 19, 8, 11, 5, 24, 2]. The purpose of this paper, however, is not to enumerate the various mechanical aspects of the various models, but to present a unifying, model-independent set of definitions for security and reliability for interactive computation. Though defining properties of interactive computations has proven subtle and elusive in the past, a concise and easily-understood property we call *resilience* captures a long list of desired security properties and provides a sturdy framework for modular protocol design and verification.

Previous work. Research into multiparty protocols has provided a variety of robust algorithms, satisfying one major goal of cryptography while simultaneously failing another: to provide confidence that the techniques are in fact reliable. The distinction between using methods based on unproven complexity-theoretic assumptions and using methods that are simply unproven is vast.

The primary reason for a lack of proofs is the lack of good definitions. Good definitions, like cryptographic research, should be:

a. easy to understand;
b. easy to use (providing simple, modular proofs);
c. unified and sufficiently robust to cover many situations.

Ill-chosen or ill-coordinated definitions make the proofs of even easily-understood ideas like secret sharing into complicated, ugly affairs. Worse, researchers are led to believe that all proofs must be complicated, eliminating the motivation to uphold the basic precept of cryptographic research: to demonstrate provably reliable techniques.

In the case of multiparty protocol security, a host of security properties have arisen in the gradual, *ad hoc* progress of research. The definitions vary with the computational and communication models (*eg.* private channels *vs.* public-key-encrypted messages). Properties like correctness and privacy are intuitively easy to understand but extremely sensitive to subtle issues in their formulation. The definition of an "input" is crucial, and while seemingly simple, it has been fraught with problems. What if a player is given an input, for example, but behaves "properly" as though it were given a different input? "Intuitive proofs" and unwieldy or inflexible approaches are often used to finesse such problems. For example, encryptions of each input may be supplied to all players, thereby fixing the inputs [19], or a player may be required to *commit* to its input, say by secretly sharing it [8, 11]. Techniques that specify the input as a function based on the transcript as a whole [3] may avoid fixing a particular committal technique in the definition of "input," but they are less than elegant.

With *ad hoc* definitions, there is no guarantee that new properties will not arise. After correctness and privacy were considered, researchers began to worry about less apparent properties. For example, faulty players should choose their inputs independently of non-faulty players; otherwise, a 2/3 majority vote might be impossible, or a global random coin could be biased. Satisfactory definitions of input independence can be obtained in an information theoretic sense, but in a resource-bounded model, where one player might choose its input to be the encryption broadcast by another, independence becomes tricky. The separate analysis of properties gives an ill-coordinated and perhaps endless list of definitions.

Even the inspiring idea of *zero-knowledge proof systems* [16], which has produced many new techniques and a greater understanding of how to measure information transfer, avoids the question of "choosing" input $x \in L$ by quantifying over all possible x. Despite the wide appeal and strength of ZK and simulation-based ideas in addressing issues of privacy (the preservation of *information*), analyzing privacy alone is insufficient to treat correctness, independence, and other properties related to *influence* wielded by an adversary. As we shall see, however, ZK uses a crucial tool — Yao's notion of indistinguishable ensembles [26] — that provides a solid foundation on which to develop a unified definition of security.

The new approach. In considering diverse properties and focusing on ZK simulations, researchers seem to have turned from the primary goal: *achieving what an ideal protocol*

achieves. Normally we assume no player is above corruption; in fact, any t-subset is vulnerable. In the ideal case, however, there is one player, the trusted host, whom the adversary cannot corrupt. The trusted host receives all inputs over private channels, computes F, and returns the results.

The *influence* and *information* of the adversary in the ideal setting are precisely delineated. The adversary may learn inputs for corrupted players, but it has no information regarding nonfaulty inputs except for the function value. It may substitute inputs for corrupted players, but it has no other influence over the nonfaulty outputs.

In order to state, however, that a protocol is secure by virtue of achieving what an ideal protocol achieves, one must define what it means to achieve the same results. A fixed comparison of a protocol to an ideal is one approach, but it is inflexible and inconvenient for constructing proofs.

Relative resilience: a security reduction. We present a reduction among protocols that allows one to compare *any* two protocols and to show that one implements the other with the same or better degree of security and reliability (including privacy, correctness, independence, fairness, *etc.*). Our thesis is as follows: *if any adversary attacking α cannot gain more information or wield more influence than when it attacks β, then α is at least as secure and reliable* — i.e. *as* resilient — *as β*. Defining such an intuitive thesis formally is the crux of our work.

An adversary attacking α may not be compatible with attacking β; we introduce an interface that translates the adversary's attacks on α to attacks on β. The interface should not itself give extra advantage to the adversary (*e.g.* through resource-unbounded computations). By considering ensembles describing *all* outputs in protocol executions (not just adversary views) and employing Yao's notion of indistinguishability [26] we give a formal means to compare attacks on α with interface-assisted attacks on β. Our use of indistinguishability is far broader than that in the analysis of ZK or of pseudorandom number generators.

Given a formal definition for *relative resilience*, an absolute measure of resilience becomes simply a comparison to some standard: a *t-resilient* protocol for F is one that is as resilient as the ideal protocol $\mathtt{ID}(F)$ for F.

Modular proof techniques. The ability to compare arbitrary pairs of protocols provides simple proofs, broader applications of the definitions, and a conceptually easier approach. Rather than using a single yardstick, we design rulers, tape measures, and micrometers, and gain a thorough understanding of the fundamental relations among the objects we measure.

The most important property of our definitions is that they provide an extremely simple and modular way to treat security. Relative resilience is transitive, supporting step-by-step modular proofs (see [4, 2]) in an arena where many protocols lack proofs because of the lack of definitions or the vast complexity of such proofs. Relative resilience makes it easy to analyze the security of concatenating secure protocols, an affair that fixed-comparisons to an ideal fail to support (*cf.* the discussion of alternative approaches in §3, notably *fault-oracles* [13, 3, 21] and *legal-versions* [5, 14]). Proofs of "folk theorems" (transitivity, concatenability, and the share-compute-reveal paradigm) using our

definitions reveal insecurities under certain circumstances, clarifying an intuitive understanding of security and contributing to a key purpose of cryptography: giving accurate security ratings. Our definitions have elucidated the important distinction between static and dynamic attacks in the "cryptographic" (computationally-bounded) setting, giving conditions under which protocols provably secure against *dynamic* adversaries can be designed [6] (previously, no proof of security against a dynamic adversary in the cryptographic setting has appeared in the literature).

The modularity in proof and protocol design has inspired vastly simplified, more efficient, and provably secure protocols for cryptographic multiparty protocols [6]. Moreover, formal proofs are short and clear.

Relative resilience unifies the idea of security and reliability for a host of interactive computations: Zero Knowledge, Byzantine Agreement, Oblivious Transfer, Instance-Hiding Schemes, *etc.*. Relative resilience explains the quite disparate collection of alternative and previous formal approaches to multiparty protocol security as particular aspects of a single principle, like electricity and magnetism, without recourse to an intricate and lengthy theory.

Thus, relative resilience satisfies the three important properties of good definitions: it is easily understood, it is easily used to give clear and formal proofs, and it captures at a single blow all desirable security properties — not just for multiparty protocols, but for virtually any kind of interactive computation.

History of ideas. In 1988, Beaver [3] developed and generalized the notion of fault-oracles introduced by Galil, Haber, and Yung [12]. Micali and Rogaway [21] independently pursued the same approach. In 1989, after a brief collaboration and exchange of ideas about fault-oracles, Beaver, Micali, and Rogaway employed the fault-oracle approach to security in a joint paper [7].

Beaver found the fault-oracle approach unsatisfactory (for reasons presented within) and went on to develop the ideas presented here. Micali and Rogaway developed the fault-oracle approach further [21] (see also these Proceedings). The reader is strongly encouraged to compare approaches[2] and to decide which is the more natural, the more concise, and the more powerful.

Contents. Relative resilience is defined in §2.2 and resilience in §2.3. In §3 we show how resilience captures zero-knowledge and other interactive computations concisely, and we compare it to other approaches to defining security. In §4 we give theorems supporting a modular approach to protocol design and proof, describing the transitivity of security and the security of protocol concatenation.

[2] including those presented in [5, 14]

2 Defining Security

2.1 Ensembles Induced by Protocol Execution

Preliminaries. Let $\Sigma = \{0,1\}$ with extra symbols $(\#, \mathbb{C}, \langle, \rangle)$ encoded naturally. Let $[n] = \{1,\ldots,n\}$ and let $\vec{x} = (x_1,\ldots,x_n)$. Let $\mathtt{dist}(X)$ be the set of distributions on some set X (usually finite). Let PFF be the set of functions mapping $(\Sigma^m)^n \to \mathtt{dist}((\Sigma^m)^n)$ described by probabilistic circuit families $\{C_F(n,m)\}$.

The difference between distributions $P, Q \in \mathtt{dist}(X)$ is $|P-Q| = \sum_X |\Pr_P[x] - \Pr_Q[x]|$. A **probabilistic function** is a function whose range contains distributions, namely $f : X \to \mathtt{dist}(Y)$. The **composition** of probabilistic functions g and f is given by $\Pr_{g \circ f(x)}[z] = \sum_y \Pr_{f(x)}[y] \Pr_{g(y)}[z]$. An **ensemble** is a probabilistic function $P : \Sigma^* \times \mathbf{N} \to \mathtt{dist}(\Sigma^*)$ such that $\Pr_{P(z,k)}[x] = 0$ for $|x| > k^c$ and some fixed c. Two ensembles P and Q are $O(\delta(k))$-**indistinguishable** ($P \approx^{\delta(k)} Q$) if $(\exists k_0)(\forall k \geq k_0)(\forall z)|P(z,k) - Q(z,k)| < \delta(k)$. Definitions of **perfect** ($O(0)$), **exponential** ($O(d^{-k})$ for some d), **statistical** ($O(k^{-d})$ for all $d > 0$), and **computational** (for all PPT-tests T and all $d > 0$, $T(P)$ is $O(k^{-d})$-indistinguishable from $T(Q)$) are standard.

Networks and Protocols. A **player** is an interactive PPTM having a random tape, input tape, output tape, and work tape. The I/O tapes may encode several (n) different tapes for communication with several machines. The superstate q_i of machine P_i is a string describing its finite control, current state, and contents of all tape squares read or written so far. With messages $m_1 \cdots m_j$ written on its input tape, P_i induces a transition $(Q_i, (C_1, M_1) \cdots (C_J, M_J)) \leftarrow \delta(q_i, m_1 \cdots m_j)$, where (C, M) requests that message M be sent on channel C.

A **channel** is a probabilistic function $C : \Sigma^* \to \mathtt{dist}(2^{\mathbf{N} \times \mathbf{N} \times \Sigma^*})$. For example, a **private channel** from i to j satisfies $\Pr_{C(m)}[\{(i,j,m)\}] = 1$, while a **broadcast channel** from i is $\Pr_{C(m)}[\{(i,1,m),\ldots,(i,n,m)\}] = 1$. An **oblivious transfer channel** from i to j gives $\Pr_{C(m)}[\{(i,j,(0,0)\}] = \frac{1}{2}$ and $\Pr_{C(m)}[\{(i,j,(1,m)\}] = \frac{1}{2}$. A **network** is a set of channel functions.

Ensembles describing protocol execution. A **protocol** is a collection of sets of players $\alpha = \{(P_1,\ldots,P_n)\}_{n \in \mathbf{N}}$. For each n (number of players), m (size of inputs), k (security parameter), $\vec{x} \in (\Sigma^m)^n$ (inputs), and $\vec{a} \in (\Sigma^*)^n$ (auxiliary inputs), a protocol α induces a distribution $\alpha(\vec{x} \cdot \vec{a}, k)$ on outputs and final views;[3] as z ranges over possible $\vec{x} \cdot \vec{a}$ values, this gives an ensemble $[\alpha]$. The messages sent from players in set A to those in B during round r are denoted $\mu(A, B, r)$. Let $R(n,m,k)$ bound the number of rounds until all nonfaulty players halt. Let $\vec{\mu} = \langle \mu([n],1,r),\ldots,\mu([n],n,r)\rangle$ denote a set of incoming messages, and denote by $\vec{q} \cdot \vec{\mu}$ a global state of the system. Let $Y_i : \Sigma^* \to \Sigma^*$ be an **output function**; the **output** of P_i is $y_i = Y_i(q_i^{R(n,m,k)})$. The **view** v_i^r of a player at round $r \leq R(n,m,k)$ is the list of states and messages it has seen; in the **memoryless model**, an intermediate view contains only the current state, tape contents, and incoming and outgoing messages. Let $\mathbf{Reformat}(\vec{q} \cdot \vec{\mu}) = \vec{y} \cdot \vec{v}$. A round and an execution of a synchronous

[3] If $\vec{x}$ or $\vec{a}$ does not have n components or if some x_i does not have m bits, the distribution gives probability 1 to the empty string.

Synchronous Protocol Execution

1 For $i = 1..n$ do $\mu^{in}(i, i, 0) \leftarrow n\#m\#k\#x_i\#a_i$ /* (function Init) */

2 For $r = 1..R(n, m, k)$ do

. For $i = 1..n$ do in parallel

2.1 /* Compute locally (function Local): */

. $(q_i^r, \mu^{out}(i, [n], r)) \leftarrow \delta(q_i^{r-1}, \mu^{in}([n], i, r-1))$

2.2 /* Apply channel functions (function Channel): */

. $\mu^{del}([n], [n], r) \leftarrow C(\mu^{out}([n], [n], r))$

2.3 /* Deliver messages output from channels (function Deliver): */

. $\mu^{in}([n], [n], r+1) \leftarrow \mu^{del}([n], [n], r)$

Figure 1: Algorithmic description of synchronous, distributed protocol execution.

protocol can be expressed as probabilistic functions:

$$\begin{aligned} \mathbf{Round}(\vec{q}\cdot\vec{\mu}) &= \mathbf{Deliver}(\mathbf{Channel}(\mathbf{Local}(\vec{q}\cdot\vec{\mu}))) \\ \mathbf{Exec}(\vec{x}\cdot\vec{a}) &= \mathbf{Reformat}(\mathbf{Round}^{R(n,m,k)}(\mathbf{Init}(\vec{x}\cdot\vec{a}))). \end{aligned}$$

Figure 1 describes protocol execution algorithmically.

Adversaries. An **adversary** $\mathcal{A}$ is an interactive TM with one communication line, on which it makes *corruption requests.* A **Byzantine** adversary requests either the view of a player i or requests to replace outgoing messages from corrupted players. Note that an adversary may change input and random tapes before the protocol starts (*i.e.* before any messages are generated). A **passive** adversary cannot change messages. If the adversary superstate is q_A, let $T = T(q_A)$ denote the set of players it has corrupted. A **t-adversary** satisfies $|T(q_A)| \leq t$; an **ideal t-adversary** also satisfies $T(q_A) \subseteq [n]$. A **static** adversary satisfies $T(q_A) = T_0$ for some fixed T_0. A **rushing, dynamic** adversary sees $\mu^{del}(\overline{T}, T, r)$ (step 2.2) before choosing whom to corrupt and how to corrupt them.

The function **Fault** provides $\mathcal{A}$ with requested information and allows it to compute a new request. The function **Replace** allows $\mathcal{A}$ to change outgoing messages $\mu^{out}(T, [n], r)$, which are then passed through channels, changing some of the messages in $\mu^{del}(T, [n], r)$ in step 2.2. Let a_A be an auxiliary input for $\mathcal{A}$, let y_A denote its output $Y_A(q_A)$, and let $\mathbf{Reformat}(\vec{q}\cdot\vec{\mu}\cdot q_A) = \vec{y}\cdot\vec{v}\cdot y_A$. An execution of a protocol with adversary is:

$$\begin{aligned} \mathbf{RoundA}(\vec{q}\cdot\vec{\mu}\cdot q_A) &= \mathbf{Deliver}(\mathbf{Replace}(\mathbf{Fault}^n(\mathbf{Channel}(\mathbf{Local}(\vec{q}\cdot\vec{\mu}\cdot q_A))))) \\ \mathbf{ExecA}(\vec{x}\cdot\vec{a}\cdot a_A) &= \mathbf{Reformat}(\mathbf{RoundA}^{R(n,m,k)}(\mathbf{Init}(\vec{x}\cdot\vec{a}\cdot a_A))). \end{aligned}$$

An execution thus maps $\vec{x}\cdot\vec{a}\cdot a_A$ to $\vec{y}\cdot\vec{v}\cdot y_A$. The ensemble of outputs induced by a **simple adversary attack** is the following:

$$[\alpha, \mathcal{A}]' \quad = \quad \vec{y}\cdot y_A \quad = \quad (Y_1(q_1), \ldots, Y_n(q_n), Y_A(q_A))$$

2.2 Relative Resilience

Interfaces. The principle behind ZK [16] states that the information revealed during a proof is bounded by the fact "$x \in L$" because a simulator can produce an accurate verifier (adversary) view based only on "$x \in L$." This brilliant use of the notion of ensemble indistinguishability [26] covers only half the picture of interaction, though. In addition to *information*, there is the *influence* an adversary has on the outputs of nonfaulty players. ZK-simulation ignores this side because a faulty verifier doesn't "influence" the final output of the prover, which is irrelevant in ZK proof systems — only the verifier's decision is considered. In an interactive protocol, the influence of the adversary is reflected in the distributions on final outputs of nonfaulty players. By examining the ensemble of all outputs, we unify the long list of desired properties (correctness, independence, *etc.*).

To say that α is as secure — *ie.* resilient — as β is to say that attacks on α do no better than attacks on β. An adversary $\mathcal{A}$ attacking α gains *information* and wields *influence* on α; its information and influence should be the same as with β. But $\mathcal{A}$ may be incompatible with β for many reasons: the network may differ from protocol α, the communication format may differ, *etc.* To allow $\mathcal{A}$ to attack β, we give it an *interface* $\mathcal{I}$. Interface $\mathcal{I}$ accepts corruption requests from $\mathcal{A}$, performs direct attacks on β, obtains views v_i^β of β, and returns *pseudo-views* $\dot{v}_i^\alpha$ (apparently of α) to $\mathcal{A}$ in response to $\mathcal{A}$'s requests.

The interface should not give $\mathcal{A}$ extra power, though; the combined machine $\mathcal{I}(\mathcal{A})$ acting as an adversary against β should not exceed the power allowed to adversaries attacking β. In cases where adversaries are polynomial-time bounded, this means that the interface itself must be polynomial-time.

Definition 1 *An* **interface** *is an interactive TM $\mathcal{I}$ with two tapes. On its "environment simulation" tape, $\mathcal{I}$ receives and responds to messages from an adversary; on its "adversarial" tape, $\mathcal{I}$ sends and receives messages as an adversary in its own right. An* **interface from** α **to** β *satisfies $\mathcal{I}(\mathcal{A}) \in \mathtt{ADV}_\beta$ for all $\mathcal{A} \in \mathtt{ADV}_\alpha$, where $\mathtt{ADV}_\alpha, \mathtt{ADV}_\beta$ are the respective classes of allowed adversaries for α, β.*

A preliminary definition illustrates the use of an interface to demonstrate that the information (adversary output) and influence (player outputs) are the same in attacks against α as in β:

Definition 2 (Weak Relative Resilience) *Protocol α is* **weakly as resilient as** *protocol β if there exists an interface $\mathcal{I}$ from α to β such that for all $\mathcal{A} \in \mathtt{ADV}_\alpha$,*

$$[\alpha, \mathcal{A}]' \approx [\beta, \mathcal{I}(\mathcal{A})]'.$$

Post Protocol Corruption. To support protocol concatenation and other issues in modularity, we strengthen this preliminary definition somewhat. In particular, an interface $\mathcal{I}$ from α to β should be able to do a good job not only during the execution of β but later on, when it may be called upon (as a result of subsequent protocol executions) to respond to requests for new corruptions. Its eternal job is to return outputs and pseudo-views seemingly from α. Definition (2), however, says nothing about its ability to do so after protocol β is complete.

A **post-protocol corrupting** adversary may, after it sees the end of α, generate a request "PPC" to enter a *post-protocol corruption* stage, at which point it receives *all* outputs $\vec{y}$ (but not the views) and then continues to make requests for corruptions in α. The interface sees neither the PPC request nor the response $\vec{y}$, but it must continue to answer corruption requests i from $\mathcal{A}$ with (final) pseudo-views v_i^α. $\mathcal{I}$ can itself obtain y_i and view v_i^β from β, but the pseudo-α-view $\dot{v}_i^\alpha$ that it generates must be accurate enough that, *even knowing all the outputs*,[4] the adversary cannot detect a difference between information from $\mathcal{I}$ and information from corrupting "the real thing," protocol α. The interface must satisfy, as usual, $(\forall \mathcal{A} \in \mathtt{ADV}_\alpha)\mathcal{I}(\mathcal{A}) \in \mathtt{ADV}_\beta$.

An execution of a protocol with a post-protocol corrupting adversary induces an ensemble on player and adversary outputs.[5] We denote this ensemble, and the ensemble restricted to adversary output alone or to player outputs alone, as:

$$\begin{array}{lcll} [\alpha,\mathcal{A}] & = & \vec{y}\cdot y_A & = & (Y_1(q_1),\ldots,Y_n(q_n),Y_A(q_A)) \\ [\alpha,\mathcal{A}]^{Y_A} & = & y_A & = & Y_A(q_A) \\ [\alpha,\mathcal{A}]^{Y_{[n]}} & = & \vec{y} & = & (Y_1(q_1),\ldots,Y_n(q_n)). \end{array}$$

When an interface is involved, we write the induced ensembles respectively as $[\beta,\mathcal{I}(\mathcal{A})]$, $[\beta,\mathcal{I}(\mathcal{A})]^{Y_A}$, and $[\beta,\mathcal{I}(\mathcal{A})]^{Y_{[n]}}$.

Relative Resilience. With interfaces, post-protocol corruption, and the induced ensembles in hand, the notion of relative resilience can be stated concisely:

Definition 3 (Relative Resilience) *Protocol α is* as resilient as β, *written $\alpha \succeq \beta$, if there exists an interface $\mathcal{I}$ from α to β such that for all $\mathcal{A} \in \mathtt{ADV}_\alpha$,*

$$[\alpha,\mathcal{A}] \approx [\beta,\mathcal{I}(\mathcal{A})].$$

The use of the $\succeq$ symbol is intended. Using Theorem 3 below, it is not hard to show that $\succeq$ defines a partial order. Complexity-theoretic reductions among problems employ a polynomial-time transducer to map one problem to another, thereby comparing their difficulty. Relative resilience is a reduction among protocols, employing an interface to map one protocol to another, thereby comparing their security and reliability.

Privacy and correctness. Relative resilience captures *a priori* the notions of privacy and correctness: α is as **private** as β if $[\alpha,\mathcal{A}]^{Y_A} \approx [\beta,\mathcal{I}(\mathcal{A})]^{Y_A}$ and it is as **correct** as β if $[\alpha,\mathcal{A}]^{Y_{[n]}} \approx [\beta,\mathcal{I}(\mathcal{A})]^{Y_{[n]}}$. We write these as $\alpha \succeq^{\text{priv}} \beta$ and $\alpha \succeq^{\text{correct}} \beta$, respectively.

Computational and other issues. Statistical and computational versions of these definitions are easy modifications using statistical and computational indistinguishability. Some qualifications are necessary in moving from the information-theoretic to the polynomially-bounded scenario. Statements of theorems presented here include polynomial bounds (*eg.* on the number of protocols to concatenate) so that they may apply for each degree of resilience; stronger statements are often possible.

[4] Why should $\mathcal{A}$ have *all* outputs? To allow protocol concatenation, we must consider the case that some (naturally ridiculous) later protocol *legitimately* reveals *all protocol-computed* information, which includes all y's — but not all views. $\mathcal{I}$'s efforts must be accurate even in this worst-case situation.

[5] Note that the *player* outputs are the same whether the adversary does post-protocol corruption or not.

2.3 Resilience

A real protocol permits an adversary class for which no player is above corruption. An ideal protocol contains one or more *trusted hosts* (players $(n+1),(n+2),\ldots$) who cannot be corrupted. Given a probabilistic finite function $F \in \mathrm{PFF}$, the **ideal protocol for** F, $\mathtt{ID}(F)$, has two rounds: (1) each player i sends x_i to host $(n+1)$; (2) host $(n+1)$ computes (or samples) $F(x_1,\ldots,x_n)$ and returns the values (one per player).

Definition 4 (Resilience) *α is a t-resilient protocol for F if $\alpha \succeq \mathtt{ID}(F)$, against t-adversaries.*

3 Unifying All Interactive Computations

To demonstrate the conciseness, clarity, and unity of our approach, we redefine a few well-known problems using relative resilience.

Zero-knowledge. In the **ideal zero-knowledge proof system**, denoted $\mathtt{ID}(L)$, player P sends x to trusted host TH, who calculates whether $x \in L$ and sends "$x \in L$" to V if so. The host otherwise sends "?" to indicate a failed proof. The classical notion of ZKPS [16] is defined in one sentence: A two-party protocol $\alpha = \langle P, V\rangle$ is a **zero-knowledge proof system** for L iff it is as resilient as $\mathtt{ID}(L)$, against static 1-adversaries.

For clarity, let us examine the relation between the two approaches to defining ZK. If the 1 adversary corrupts neither P nor V, then resilience ensures that V accepts the correct statement "$x \in L$" from P and checked by the trusted host; *completeness* is satisfied. If the 1-adversary corrupts P, resilience ensures that a nonfaulty V never accepts a false statement "$x \in L$"; *soundness* is satisfied. If the 1-adversary corrupts V, resilience ensures that the adversary gains no more information than $\mathcal{I}$ corrupting V in the ideal case, where the host ensures that only the statement "$x \in L$" is transmitted.

Notice that a corrupt P does have *influence* over the output of V: it can cause V not to believe a proof or it can convince V of a true statement, but its influence is bounded by that permitted in the ideal case. Relative resilience maps the adversary's limitations in the ideal case to the limitations it *should* have in the real case, without having to enumerate all desired properties.

Secret sharing. The **ideal vacuous protocol**, $\mathtt{ID}(0)$, returns no result. A **t-threshold scheme** is a pair of protocols $(\textsc{Sha}, \textsc{Rec})$ computing probabilistic functions $(\mathbf{sha}, \mathbf{rec})$ such that

1. $\mathbf{rec}$ is t-robust (*i.e.*, insensitive to $\leq t$ changes in inputs);
2. $\mathbf{sha}$ is t-private (*i.e.*, $\mathtt{ID}(\mathbf{sha}) \succeq \mathtt{ID}(0)$);
3. $\mathbf{rec} \circ \mathbf{sha}(x) = x$.

Other examples. Byzantine Agreement, Oblivious Transfer, Coin Flipping, Instance-Hiding Schemes, and a host of interactive computations have simple descriptions using relative resilience. These are left to the reader.

Other recent definitions: their disadvantages; a unified theory. Several authors [13, 3, 21] have considered the idea of a fault-oracle that performs a single function computation and leaks a subset of the inputs. Extending the ZK-approach, they require a simulator with access to a fault-oracle to generate an adversarial view based on seeing and changing a subset of inputs and receiving a single computation of F. Rephrased in the language of relative resilience, a fault-oracle represents a trusted host, and a simulator provides a fixed comparison to the "ideal" protocol. Oracles lack the flexibility of allowing comparisons among various protocols (and exclude those not designed to "compute a function") and do not support an obvious modular framework. For example, it is not clear how the concatenation of two secure protocols may be proven secure without altering the definitions, since the security of each is proven with respect to a separate oracle call, while the simulator for the concatenated protocol may make only one oracle call. The notion of transitivity of security is not expressible in terms of fault-oracles.

The legal-version approach says that a protocol is "legal-robust" if every execution with an adversary corresponds to an execution in which all players behave, except that the inputs of some subset may be different [5, 14]. This approach is again a fixed comparison to an essentially ideal situation, lacking flexibility and modularity. In the language of relative resilience: define the **ideal version** $\mathrm{ID}(\alpha)$ of a protocol α to be the ideal protocol in which the trusted host (1) receives inputs and random tapes from each player and (2) *internally* runs α (without corruption), returning the outputs (not the internal views) to the players. Then, a protocol α is **legal-robust** if the *fixed* comparison $\alpha \succeq \mathrm{ID}(\alpha)$ holds.

4 Folk Theorems and Modular Proof Techniques

Many intuitively justified approaches become provable using relative resilience. Resilience brings to light several pitfalls overlooked by imprecisely stated folk theorems. The contribution of this section is a formal statement of provable theorems and the necessary conditions under which they hold. We sketch the proofs; full versions require a little more space than here permitted.

The composition of $f(z,k)$ ensembles is $P^f(z,k) = P_{f(z,k)} \circ \cdots \circ P_1(z,k)$. Protocol concatenation is defined naturally. The concatenation of $f(n,m,k)$ protocols is $\alpha^f(\vec{x}\cdot\vec{a},k) = \alpha_{f(n,m,k)} \circ \cdots \circ \alpha_1(\vec{x}\cdot\vec{a},k)$.

The mathematical distinction between uniform and pointwise convergence of functions has a probabilistic counterpart: two **families** $\{P_i\}$ and $\{Q_i\}$ of ensembles are **uniformly** $O(\delta_i(z,k))$**-indistinguishable** if $(\exists k_0)(\forall i)(\forall k \geq k_0)(\forall z)|P_i(z,k) - Q_i(z,k)| < \delta_i(z,k)$. The term *uniform* does *not* refer to Turing machine computability. We say that protocol family $\{\alpha_i\}$ is **uniformly as** t**-resilient as** family $\{\beta_i\}$ if the ensemble families $\{[\alpha_i, \mathcal{A}]\}$ and $\{[\beta_i, \mathcal{I}(\mathcal{A})]\}$ are uniformly indistinguishable.

Uniform convergence is necessary to the following theorems; counterexamples are otherwise easy to find (see below and [2]), showing that folk theorems cannot be applied without care. Luckily for many presumably secure applications, any two *finite* families of ensembles are uniformly indistinguishable if each pair of corresponding ensembles are indistinguishable.

Lemma 1 (Composing poly-many ensembles) *If $\{P_i\}$ is* uniformly *indistinguishable from $\{Q_i\}$ and if $f(z,k)$ is polynomially bounded, then $P^f \approx Q^f$.*

Proof. We sketch a proof for perfect and statistical indistinguishability appearing in [2, 4]; the proof for computational indistinguishability is direct and follows the same lines. Unlike probability-walk proofs in other settings, we must explicitly consider the convergence rates — serendipity does not allow us to omit them. Assume by way of contradiction that $\{P_i\}$ is uniformly $\delta_i(z,k)$-indistinguishable from $\{Q_i\}$ and f is poly-bounded, but that P^f is not $\delta(z,k)$-indistinguishable from Q^f, where $\delta(z,k) = \sum_{i=1}^{f(z,k)} \delta_i(z,k)$. Let k_0 be the uniform convergence parameter. Define $R_l = P_{f(z,k)} \circ \cdots \circ P_l \circ Q_{l-1} \circ \cdots \circ Q_1$. Then for some $k \geq k_0$ and for some z,

$$\begin{aligned}
\sum_{i=1}^{f(z,k)} \delta_i(z,k) &< \sum_z |P_{f(z,k)} \circ \cdots \circ P_1(z,k) - Q_{f(z,k)} \circ \cdots \circ Q_1(z,k)| \\
&\leq \sum_z |R_0 - R_1 + R_1 - R_2 + \cdots + R_{f(z,k)-1} - R_{f(z,k)}| \\
&\leq \sum_z \sum_{i=1}^{f(z,k)} |R_{i-1}(z,k) - R_i(z,k)|.
\end{aligned}$$

Reversing the order of summation, it follows that for some i, $\delta_i(z,k) < \sum_z |R_{i-1}(z,k) - R_i(z,k)|$. From here it is straightforward to show that this demonstrates P_i is not $\delta_i(z,k)$-indistinguishable from Q_i for this $k \geq k_0$, a contradiction. □

A counterexample is easy to describe for *nonuniform* convergence [2]; for example, define:

$$\begin{aligned}
\Pr_{P_i(z,k)}[0] &= 1 \quad \text{if } k < 2i \\
\Pr_{P_i(z,k)}[1] &= 1 \quad \text{if } k \geq 2i \\
\Pr_{Q_i(z,k)}[1] &= 1 \quad \text{always.}
\end{aligned}$$

Clearly, for all i, $P_i \approx^{O(2^{-k})} Q_i$ since $(\forall i, z)(\forall k \geq 2i) P_i(z,k) = Q_i(z,k)$. Letting $f(z,k) = k$, we see $(\forall z,k)\Pr_{P^f(z,k)}[0] = 1$, because $P^f(z,k) = P_k(P_{k-1}(\cdots),k)$ and $k < 2k$. On the other hand, $(\forall z,k)\Pr_{Q^f(z,k)}[0] = 0$. Hence $(\forall z,k)|P^f(z,k) - Q^f(z,k)| = 2 \neq O(k2^{-k})$, a contradiction. Uniformity of convergence is also necessary when comparing polynomially-many protocols:

Theorem 2 (Concatenating poly-many protocols) *If $\{\alpha_i\}$ is* uniformly *as t-resilient as $\{\beta_i\}$ and if $f(n,m,k)$ is polynomially bounded, then $\alpha^f \succeq \beta^f$.*

Proof. We sketch a few salient points, following our proof for statistical resilience in [2]. The main idea is to construct an interface $\mathcal{I}$ from α^f to β^f, using interfaces $\mathcal{I}_i$ between each pair α_i and β_i. Each interface is taken without loss of generality[6] to be *canonical*, *ie.*

[6] A simple argument demonstrates that no generality is lost for information-theoretic settings; a qualification must be made when defining computational resilience.

the set of corruption requests by $\mathcal{I}_i$ against β_i is always a subset of those requested by $\mathcal{A}_i$. Using Lemma 1, the two ensembles $[\alpha^f, \mathcal{I}(\mathcal{A})]$ and $[\beta^f, \mathcal{I}(\mathcal{A})]$ are shown indistinguishable. □

Theorem 3 (Transitivity of $\succeq$) *$\alpha \succeq \beta$ and $\beta \succeq \gamma$ implies $\alpha \succeq \gamma$. Polynomially many applications work as well: If $\{\alpha_{i+1}\}$ is* uniformly *as t-resilient as $\{\alpha_i\}$,*[7] *and if $f(n, m, k)$ is polynomially bounded, then $\alpha^f \succeq \alpha_0$.*

Proof. See [2, 4]; the interface $\mathcal{I}$ from α^f to α_0 internally runs $f(n, m, k)$ nested subinterfaces $\mathcal{I}_{f,f-1} \circ \cdots \circ \mathcal{I}_{2,1} \circ \mathcal{I}_{1,0}$. We consider canonical interfaces as in the proof of Theorem 2. □

4.1 The Share-Compute-Reveal Paradigm

Define $\mathbf{hide}(H) = \mathbf{sha} \circ H \circ \mathbf{rec}$, the probabilistic function that reconstructs a secretly-shared value, computes H, and shares it again. The share-compute-reveal paradigm [18, 19, 8, 11] for multiparty protocols is to express F as $F^f = \circ_1^f F_i$ and compute $\mathbf{rec} \circ [\circ_1^f \mathbf{hide}(F_i)] \circ \mathbf{sha}$. That is, inputs are secretly shared; intermediate values are secretly computed but not revealed; then the final output is reconstructed. The formal and fully provable statement of this methodology is as follows:

Theorem 4 ("Completeness" paradigm) *Let* (SHA, REC) *be a t-threshold scheme, and let $F = F^f = \circ_1^f F_i$ for some polynomially bounded $f(n, m, k)$. If $\{\alpha_i\}$ is* uniformly *as t-resilient as $\{\mathrm{ID}(\mathbf{hide}(F_i))\}$, then*

$$\mathrm{REC} \circ [\circ_1^f \alpha_i] \circ \mathrm{SHA} \succeq \mathrm{ID}(F).$$

Proof. *Uniformity* is essential. The proof uses Theorems 2 and 3 and the robustness and privacy of $(\mathbf{rec}, \mathbf{sha})$ but is omitted (*cf.* [2, 4], however). □

5 Summary

A correct, concise, and useful formulation of security and fault-tolerance is necessary to a proper foundation of the theory and practice of cryptography. We are happy to provide proofs of our theorems upon request and are even happier to discuss them openly and to develop our methodology further.

We have emphasized the important point that protocol security must be treated in a unified manner: security and reliability (privacy and correctness) are two sides of the same coin. Treating a variety of properties separately leads to confusion.

[7] Roughly speaking, $(\forall i)\alpha_{i+1} \succeq \alpha_i$ — with uniform convergence.

To say that a protocol is secure is to say it achieves the results of an ideal protocol. Our formal definition of relative resilience states precisely what it means for one protocol to achieve the results of another, avoiding the inflexibility inherent in fault-oracle approaches. Relative resilience provides a natural *reduction* among protocols that supports concise and modular proofs, inspiring newfound confidence in theoretical cryptography and resulting in clearer and more efficient design of provably secure protocols.

References

[1] M. Abadi, J. Feigenbaum, J. Kilian. "On Hiding Information from an Oracle." J. Comput. System Sci. 39 (1989), 21-50.

[2] D. Beaver. "Secure Multiparty Protocols and Zero Knowledge Proof Systems Tolerating a Faulty Minority." To appear, *J. Cryptology.* An earlier version appeared as "Secure Multiparty Protocols Tolerating Half Faulty Processors." *Proceedings of Crypto 1989,* ACM, 1989.

[3] D. Beaver. "Formal Definitions for Secure Distributed Protocols." *Proceedings of the DIMACS Workshop on Distributed Computing and Cryptography,* Princeton, NJ, October, 1989, J. Feigenbaum, M. Merritt (eds.).

[4] D. Beaver. *Security, Fault Tolerance, and Communication Complexity in Distributed Systems.* PhD Thesis, Harvard University, Cambridge, 1990.

[5] D. Beaver, S. Goldwasser. "Multiparty Computation with Faulty Majority." *Proceedings of the* 30^{th} *FOCS,* IEEE, 1989, 468–473.

[6] D. Beaver, S. Haber. "Cryptographic Protocols Provably Secure Against Dynamic Adversaries." Submitted to FOCS 91.

[7] D. Beaver, S. Micali, P. Rogaway. "The Round Complexity of Secure Protocols." *Proceedings of the* 22^{st} *STOC,* ACM, 1990, 503–513.

[8] M. Ben-Or, S. Goldwasser, A. Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation." *Proceedings of the* 20^{th} *STOC,* ACM, 1988, 1–10.

[9] R. Blakley. "Security Proofs for Information Protection Systems." *Proceedings of the 1980 Symposium on Security and Privacy,* IEEE Computer Society Press, New York, 1981, 79–88.

[10] G. Brassard, D. Chaum, C. Crépeau. "Minimum Disclosure Proofs of Knowledge." *J. Comput. System Sci.* **37** (1988), 156–189.

[11] D. Chaum, C. Crépeau, I. Damgård. "Multiparty Unconditionally Secure Protocols." *Proceedings of the* 20^{th} *STOC,* ACM, 1988, 11–19.

[12] Z. Galil, S. Haber, M. Yung. "Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model." *Proceedings of Crypto 1987,* Springer–Verlag, 1988, 135–155.

[13] Z. Galil, S. Haber, and M. Yung. "Minimum-Knowledge Interactive Proofs for Decision Problems." *SIAM J. Comput.* 18:4 (1989), 711–739.

[14] S. Goldwasser, L. Levin. "Fair Computation of General Functions in Presence of Immoral Majority." *Proceedings of Crypto 1990.*

[15] S. Goldwasser, S. Micali. "Probabilistic Encryption." *J. Comput. System Sci.* **28** (1984), 270–299.

[16] S. Goldwasser, S. Micali, C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems." *SIAM J. Comput.* 18:1 (1989), 186–208.

[17] S. Goldwasser, M. Sipser. "Private Coins vs. Public Coins in Interactive Proof Systems." *Proceedings of the* 18^{th} *STOC,* ACM, 1986, 59–68.

[18] O. Goldreich, S. Micali, A. Wigderson. "Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design." *Proceedings of the* 27^{th} *FOCS,* IEEE, 1986, 174–187.

[19] O. Goldreich, S. Micali, A. Wigderson. "How to Play Any Mental Game, or A Completeness Theorem for Protocols with Honest Majority." *Proceedings of the* 19^{th} *STOC,* ACM, 1987, 218–229.

[20] S. Haber. *Multi-Party Cryptographic Computation: Techniques and Applications,* PhD Thesis, Columbia University, 1988.

[21] S. Micali, P. Rogaway. "The Notion of Secure Computation." Unpublished Manuscript, 1990.

[22] S. Micali, P. Rogaway. "Secure Computation." These Proceedings (*Crypto 1991*), page 9.8.

[23] Y. Oren. "On the Cunning Power of Cheating Verifiers: Some Observations about Zero Knowledge Proofs." *Proceedings of the* 19^{th} *STOC,* ACM, 1987, 462–471.

[24] T. Rabin, M. Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority." *Proceedings of the* 21^{st} *STOC,* ACM, 1989, 73–85.

[25] A. Shamir. "How to Share a Secret." *Communications of the ACM,* **22** (1979), 612–613.

[26] A. Yao, "Theory and Applications of Trapdoor Functions." *Proceedings of the* 23^{rd} *FOCS,* IEEE, 1982, 80–91.

[27] A. Yao. "How to Generate and Exchange Secrets." *Proceedings of the* 27^{th} *FOCS,* IEEE, 1986, 162–167.