# Adaptively Secure Oblivious Transfer

Donald Beaver *

Transarc Corp.

**Abstract.** Oblivious Transfer (OT) is a ubiquitous cryptographic tool that is of fundamental importance in secure protocol design. Despite extensive research into the design and verification of secure and efficient solutions, existing OT protocols enjoy "provable" security only against static attacks, in which an adversary must choose in advance whom it will corrupt.

This model severely limits the applicability of OT, since it provides no verifiable security against attackers who choose their victims adaptively (anytime during or after the protocol) or may even corrupt both players (which is not a moot point in a larger network protocol). This issue arises even if the communication model provides absolutely secure channels.

Recent attention has been given to accomplishing adaptive security for encryption, multiparty protocols (for $n > 3$ participants, with faulty minority), and zero-knowledge proofs.

Our work fills the remaining gap by demonstrating the first (provably) adaptively secure protocol for OT, and consequently for fully general two-party interactive computations. Based on the intractability of discrete logarithms, or more generally on a minimally restricted type of one-way trapdoor permutation, our protocols provably withstand attacks that may compromise Alice or Bob, or both, at any time.

## 1 Introduction

In the *Millionaires' Problem* [Yao82a], Alice and Bob wish to determine who has more money, without revealing how much each one respectively has. This problem is a special case of the more general *two-party function computation* problem, in which Alice and Bob wish to compute some arbitrary discrete function $f(x, y)$, where Alice holds $x$ and Bob holds $y$, without revealing anything more about $x$ and $y$ than what $f(x, y)$ reveals.

Kilian [K88] provided an elegant general solution based on the fundamental primitive known as *Oblivious Transfer* (OT). Introduced by Rabin [R81], OT is a process by which Alice transmits a bit $b$ to Bob over a "noisy" channel: Bob learns $b$ with probability 1/2, but Alice does not discover whether Bob succeeded or failed in learning $b$. This simple asymmetry in knowledge provides the basis not only for two-party function computation but for a variety of other

* Transarc Corp., 707 Grant St., Pittsburgh, PA, 15219, USA; 412-338-4365; beaver@transarc.com, http://www.transarc.com/~beaver

K. Ohta and D. Pei (Eds.): ASIACRYPT'98, LNCS 1514, pp. 300-314, 1998.

cryptographic tasks, including bit commitment and zero-knowledge proofs [K88, BCC88, GMW87, GMR89].

Because of its importance, many implementations for OT exist [R81, EGL85, BCR86a, BCR86b, BM89, KMO89, HL90, dB91, B92], based on a variety of unproven intractability assumptions and providing varying degrees of efficiency and security. Some provide unconditional security for Alice; some provide unconditional security for Bob.

For most, a proof of security against static 1-adversaries has been offered or is straightforward to construct. In other words, most approaches support a case-by-case analysis: an always-honest Alice is protected against Bob (adversary corrupted Bob in advance), or an always-honest Bob is protected against Alice (adversary corrupted Alice in advance).

Such verification is technically insufficient, in and of itself, to demonstrate security against *adaptive* attacks. This does not immediately provide a means to break existing protocols, but it does mean that they remain at best intuitively secure. Worse, a partial verification (*i.e.* for static attacks only) is misleading when it suggests robustness against adaptive attacks.

In addition, certain applications of OT protocols introduce dangerous logical deficiencies, even when only static adversaries are involved. That is, if a statically-verified OT protocol is used in a non-black-box manner within a larger protocol, then the "obvious" deduction that the larger protocol is secure against merely static attacks may be dangerously incorrect. (See §1.4.)

To utilize Kilian's foundational result in the most robust and general fashion, an OT implementation is needed that enjoys a proof of security against adaptive 2-attacks. This paper provides such an implementation for the first time, and it uses common intractability assumptions.

### 1.1 Network Security and Adaptive Attacks

*What is Adaptive Security?* Unlike static adversaries, adaptive adversaries are able to corrupt one or more players at any time during or after a protocol.

Often, security is argued through a simulator-based approach *à la* Goldwasser, Micali and Rackoff [GMR89] (*cf.* [MR91, B91]). A simulator $\mathcal{S}$ is given access to some ideal setting (*e.g.*, the rock-hard exterior of an absolutely secure channel), and it must provide a realistic virtual environment for the adversary. If the adversary cannot tell the difference between this environment and an actual execution, then the actual execution does not leak any more information (or provide more influence on results) than the ideal setting.

Encryption over public channels provides the simplest illustration of the process. A static security proof typically arranges for $\mathcal{S}$ to create a fake encryption[2] $E(k, 0, r)$ of 0. In accomplishing this, $\mathcal{S}$ clearly needs (and is granted) no access to the message protected by the ideal channel.

[2] It is convenient to consider just public-key encryption, although even private-key encryption such as DES suffers the same problems with adaptive attacks. Here, $k$ is an encryption key; $r$ is a random string.

If an adversary can later corrupt the sender, then it (as well as $\mathcal{S}$) is now entitled to learn the cleartext, $m$. But because $\mathcal{S}$ cannot generally find $k'$ and/or $r'$ such that $E(k', m, r') = E(k, 0, r)$, an adversary can easily detect the inconsistency. Resetting the adversary is not viable, either, particularly when in the meantime it has examined thousands of other (simulated) messages being delivered within a larger-scale interaction.

Even if $\mathcal{S}$ uses a random or cleverly chosen message $m'$ instead of 0, it is highly likely that it will be mistaken. The very security of the ideal channel itself makes this problem fundamentally inevitable.

*Why is Adaptive Security Important?* At first glance, the distinction seems an obscure technicality. In reality, however, adaptive security reflects a more natural and applicable threat model. Although analyzing a protocol according to each possible corruption pattern appears to be a convincing argument for security, the fundamental problem is that real-world attackers need not choose in advance whom they will corrupt; nor are they restricted to corrupting at most one party.

These factors are particularly evident when OT (or encryption) is used as a pluggable component in a larger-scale protocol involving many parties. Any particular OT execution might be overrun by an adversary who eventually chooses to corrupt *both* parties – whether immediately or later.

The technical issue would be moot if there were an obvious mapping from static arguments to the adaptive case. No such mapping is known. Indeed, the opposite seems to be true for certain protocols, which enjoy proofs of static security but are unlikely to enjoy proofs of adaptive security, at least using simulator-based approaches [B95a, B96].

*What are the Obstacles?* Even though computational encryption makes it difficult to discover the cleartext, it binds the sender and receiver to the cleartext. That is, because there is no *equivocation* of the message given the cleartext, neither sender nor receiver can find a different key or random input to map the ciphertext to a different cleartext.

This holdover from Shannon is a curse on adaptive security in the computational setting. Not only are the sender and receiver bound to the cleartext (even though hidden!), so is the *simulator* itself.

### 1.2 Specifics of Oblivious Transfer

For OT, the ideal setting contains a trusted third party who receives $b$ from Alice and decides randomly whether to send $(0, 0)$ or $(1, b)$ to Bob. The simulator can inspect and control those ideal parties (Alice/Bob) if and precisely when the adversary has requested their corruption.

In particular, $\mathcal{S}$ must provide a view of the conversation between $A$ and $B$ over a public channel, even before any corruptions have been requested. When Alice (or Bob) is then corrupted, $\mathcal{S}$ should "back-patch" its view to show an internal history of Alice (or Bob) that is consistent with the conversation. For security against 2-adversaries, $\mathcal{S}$ must also be prepared to provide a fake history

for Bob (resp. Alice) in the future, if the adversary Adv later requests a second corruption.

*Example.* For concreteness, consider Rabin's OT protocol [R81]. Alice generates $n = pq$ as a product of large Blum primes, then sends $n$ and (for simplicity, say) $s = (-1)^b r^2 \bmod n$ to Bob, for a random $r \bmod n$. Bob chooses a secret $x \bmod n$ and sends $z = x^2 \bmod n$ to Alice. Alice chooses one of the four square roots $\{x, -x, y, -y\}$ of $z$ and sends it to Bob. If Alice chose $\pm x$, Bob learns nothing, but if Alice chose $\pm y$, Bob can factor $n$ and discover $b$.

Now, say that Adv corrupts "real" Bob. Even if the whole conversation had been encrypted, Adv now learns the traffic described above, and $\mathcal{S}$ must simulate it. $\mathcal{S}$ is entitled to learn what a trusted third party handed over to "ideal" Bob in the ideal case. With probability 1/2, $\mathcal{S}$ failed to learn $b$, yet it must present Adv with some $s$.

We might try the approach that seems to suffice for the static case: just make $s$ up using a guessed $b$, and Adv will never know the difference. But Adv may choose to corrupt "real" Alice a hundred years later (even for reasons completely independent of this OT execution), at which point $\mathcal{S}$ has to report a consistent internal history for Alice. Indeed, $\mathcal{S}$ is now entitled to learn $b$ by corrupting the "ideal" Alice. But the fake value of $s$ can be "decoded" in only one way, and with probability 1/2, $\mathcal{S}$'s earlier faked value will be inconsistent with $b$, causing the simulation to fail.[3]

### 1.3 Adaptive Security: Related Work

The fundamental importance of *adaptively*-secure solutions is underlined by recent solutions for several fundamental cryptographic tasks, including:

- *Encryption* [CFGN96]
- *Multiparty computation* (for $n > 3$) [CFGN96]
- *Zero-knowledge proofs and arguments* [B95a] (*cf.* [BCC88, FS90a])
- *Bit committal* [B95a] (*cf.* [BCC88, FS90a])

*Erasing.* Simulation can be finessed in settings where erasing internal information is allowed [BH92, F88]. By deleting sensitive information, players remove the evidence that might otherwise indicate a simulator's mistaken guess. If a private key is no longer available, then the adversarial view, although information-theoretically improper, will reveal no contradiction.

*Fake Ciphertexts with Equivocation.* Using public channels while maintaining complete internal records is a significantly greater challenge. Recently, Canetti, Feige, Goldreich and Naor [CFGN96] developed a secure encryption scheme without erasing, based on honest parties' refraining from learning certain bits. This

[3] A simulator for the static case doesn't ever have to face this possibility; it only produces fake information for Alice when Alice is corrupt from the start, in which case it knows $b$ already.

important idea enables the simulator to construct fake ciphertexts that can be made consistent with either 0 or 1.

Naturally, the facsimiles are imperfect (otherwise a receiver could not tell whether the message was 0 or 1), but it is computationally difficult to distinguish them from actual ciphertexts.

### 1.4 Previous Work Insufficient for OT

There is good reason why OT is conspicuously missing from the preceding list. Generally speaking, the earlier settings demand at most *one-sided* privacy, whereas OT requires *two-sided* privacy.

That is, in earlier settings, at most one of the parties is hiding information from the other. Therefore $\mathcal{S}$ holds *no* information from the ideal setting, until it gains *all* information as soon as a sensitive party (sender/receiver/prover/committer) is corrupted. Thus, $\mathcal{S}$ need only prepare for one "surprise" event, namely when it suddenly gains the private information and must back-patch its current simulation.

In OT, however, each party withholds information from the other. Achieving equivocation in both directions *simultaneously* is a significantly different and harder task. The simulator must be prepared to back-patch flexibly with two kinds of newly-gained data, depending on which player is first compromised. Even thereafter, the ongoing simulation must still be prepared for an eventual back-patching needed to show consistency with the still-unknown data held by the other player. This remains true even if the interaction occurs over an absolutely secure channel.

*Two-Sided Equivocation.* Beaver recently characterized two equivocation properties for OT [B96]: An OT implementation is **content-equivocable**[4] (C.E.) if $\mathcal{S}$ can generate views (whether or not $B$ is yet corrupt) so that if $A$ is suddenly corrupted, the views can be made consistent with $A$ having transmitted $b = 0$ or $b = 1$. Likewise, the implementation is **result-equivocable** (R.E.) if $\mathcal{S}$ can patch a view consistently with "received" or "didn't receive" when Bob is suddenly corrupted.

Weaker equivocation properties are also useful to consider, particularly when the traffic itself between $A$ and $B$ is also encrypted. An OT protocol is **weakly content-equivocable** if $\mathcal{S}$ need do the appropriate patching only when Bob is already corrupt. An OT protocol is **weakly result-equivocable** if $\mathcal{S}$ need do the appropriate patching only when Alice is already corrupt.

For example, the Rabin protocol is result-equivocable but not weakly content-equivocable. According to need, $\mathcal{S}$ can use an appropriate choice from $\{x, -x, y, -y\}$ as the "actual" $x$ that Bob chose, thereby switching whether Bob received $b$ or not. But as described earlier, the announced value of $s$ prevents $\mathcal{S}$ from adapting $b$ itself.

[4] The term "equivocable" means "can be made to appear equivocal." Equivocal *ciphertexts* convey no information and are useless for communication; but equivocal *facsimiles* enable flexible back-patching.

Weak content-equivocation is virtually identical to the "chameleon" property [BCC88] for bit commitment. Unfortunately, the methods in [BCC88] do not generalize to achieve both C.E. and R.E. properties simultaneously for OT.

Notably, no known OT protocol is both weakly content- and weakly result- equivocable [B96]. This includes the protocols described in Rabin [R81], Even/Goldreich/Lempel [EGL85], Goldreich/Micali/Wigderson [GMW87], Bellare/Micali [BM89], Den Boer [dB91], and Beaver [B96].

*Insufficiencies for* Static *Attacks.* Even ignoring adaptive attacks altogether, there are subtle dangers in using OT protocols in larger protocols. Unless the protocol is used in a black-box manner, it can be incorrect to deduce that a larger protocol is secure against merely static attacks based on a proof that the OT subprotocol is secure against static attacks.

As an illustration, recall Rabin's protocol for OT, in the case where Alice is honest, *i.e.* where $\mathcal{S}$ does not have access to $b$. If $\mathcal{S}$ uses the encryption-style simulation, it sets $b = 0$ (or guesses a random $b$).

Now, imagine using this kind of OT protocol for commitment purposes, to tie Alice to each bit $b$. As long as Alice does not ever reveal the $r$ value, it is always possible to "reveal" a $b'$ value that is inconsistent with $s = (-1)^b r^2$. (Discovering this inconsistency is just the Quadratic Residuosity problem.)

A "black-box" use of OT would never instruct Alice to "decommit" $b$ by revealing $r$. (Thus, even if $b$ is revealed later in a larger protocol, it remains infeasible to detect whether $\mathcal{S}$'s facsimile has the wrong quadratic residuosity.) But a less well-bred protocol might indeed use this attractive ability to decommit $b$. In that case, it would be incorrect to extend a claim of security to the larger protocol, even against static attacks. This is because the quadratic residuosity of the simulator's fake $s$ value will match an unknown $b$ value only half the time, and $\mathcal{S}$'s attempts at simulation will fail. This problem is particularly acute where *encryptions* are used as committals in such a non-black-box way.

Again, the protocol may not be obviously breakable, but the deduction that it is provably secure would be incorrect.

### 1.5 Results

Our results are complementary to recent advances in adaptive security in the related but distinct domains of encryption, proofs and committal.

We give the first known protocol for Oblivious Transfer that admits a proof of security against attacks by adaptive 2-adversaries:

**Theorem 1.** *There exists an implementation of Oblivious Transfer that is secure against adaptive* 2*-adversaries, if the Diffie-Hellman Assumption holds.*

Our methods require a small constant number of exponentiations and are comparable to the complexity of statically-secure OT implementations.

Similar results hold for other cryptographic assumptions such as the intractability of factoring or breaking RSA. More generally, they hold for a slightly

restricted type of one-way trapdoor permutation, one which allows the selection of a permutation without knowing the trapdoor [CFGN96].

Although secure channels are insufficient, we make use of methods in [B97] that employ (statically-secure) key exchange in a bizarre fashion, intentionally *revealing* the keys that are mutually generated.

*Contents.* §2 describes notation, formalities, and OT variants. §3 presents our solution based on the Diffie-Hellman assumption. §4 describes a proof of security against adaptive attacks. §5 discusses generalizations of the techniques.

## 2 Background and Notation

*Notation.* Let $\$(S)$ denote the uniformly random distribution over finite set $S$.

Let $p$ be a prime. Let $\mathbf{Z}_p^* = \{1, 2, \ldots, p-1\}$ and let $\mathbf{Z}_{p-1} = \{0, 1, 2, \ldots, p-2\}$.

*Attacks: Static or Adaptive.* An adversary is a probabilistic poly-time TM (PPTM) that issues two sorts of messages: "*corrupt* $i$," "*send* $m$ from $i$ to $j$." It receives two sorts of responses: "*view of* $i$," "*receive* $m$ from $j$ to $i$." Whether its send/receive message is honored depends on whether it has issued a request to corrupt $i$.

A **static $t$-adversary** is an adversary who issues up to $t$ *corrupt* requests before the protocol starts. An **adaptive $t$-adversary** may issue up to $t$ such requests at any time.

*OT specification.* The **specification protocol for OT** is a three-party protocol consisting of $\hat{A}$, $\hat{B}$, and incorruptible party OT. $\hat{A}$ has input $b$, which it is instructed to send to OT. OT flips a coin, $?b$, and sends $(?b, ?b \wedge b)$ to $\hat{B}$.[5] The communication channels between $\hat{A}$ and OT and between OT and $\hat{B}$ are absolutely private.

We also consider two variants on OT: one-out-of-two OT ($\frac{1}{2}$OT), in which Alice holds $(b_0, b_1)$ and Bob receives $(c, b_c)$ for a random $c$ unknown to Alice [EGL85]; and chosen one-out-of-two OT ($\binom{2}{1}$OT), in which Alice holds $(b_0, b_1)$ and Bob receives $b_c$ for a $c$ of his choice, but unknown to Alice.

*Simulation-based security.* The definition of simulator-based static security is the standard approach: find an appropriate simulator for the case in which Alice is bad, and another simulator for when Bob is bad. We focus on the adaptive case.

In the adaptive case, there is a single simulator, $\mathcal{S}$, who receives requests from and delivers responses to the attacker, `Adv`, creating an environment for `Adv` as though `Adv` were attacking a given implementation. $\mathcal{S}$ is itself an attacker acting within the specification protocol for OT, which is run with $\hat{A}$ on input $b$. When `Adv` corrupts player $i$, $\mathcal{S}$ issues a corruption request and is given $\hat{i}$'s

[5] Thus, $(0, 0)$ means "failed," while $(1, b)$ means "received $b$."

information.[6] $\mathcal{S}$ responds to Adv with a facsimile of the "*view of* $i$" response that Adv expects. $\mathcal{S}$ receives all of Adv's "*send* $m$" requests and provides Adv with facsimiles of "*receive* $m$" responses. Finally, Adv (or $\mathcal{S}$ on Adv's behalf) writes its output, $y_{\mathtt{Adv}}$.

Let Adv, with auxiliary input $x_{\mathtt{Adv}}$, attack a given OT implementation OT in which Alice holds input $b$. The execution induces a distribution $(A(b), B, \mathtt{Adv}(x_{\mathtt{Adv}}))$ on output triples, $(y_A, y_B, y_{\mathtt{Adv}})$.

Let $\mathcal{S}(\mathtt{Adv}(x_{\mathtt{Adv}}))$ attack the OT specification. The execution induces a distribution $(\hat{A}(b), \hat{B}, \mathcal{S}(\mathtt{Adv}(x_{\mathtt{Adv}})))$ on output triples, $(y_{\hat{A}}, y_{\hat{B}}, y_{\mathcal{S}})$.

An extra, "security parameter" $k$ may be included. This provides a sequence of distributions on output triples in each scenario. Let $\approx$ denote *computational indistinguishability*, a notion whose formal definition is omitted for reasons of space (*cf.* [GMR89]).

The implementation $\mathcal{OT}$ is **secure against adaptive $t$-adversaries** if, for any adaptive $t$-adversary Adv, there is a PPTM simulator $\mathcal{S}$ such that for any $b$, $(A(b), B, \mathtt{Adv}(x_{\mathtt{Adv}})) \approx (\hat{A}(b), \hat{B}, \mathcal{S}(\mathtt{Adv}(x_{\mathtt{Adv}})))$. In other words, the simulator maps attacks on the implementation to equivalent attacks on the specification.

*Assumptions.* Let $p$ be a "safe" prime, namely $p - 1 = 2q$, where $q$ are prime. Let $\hat{g}$ be a generator of $\mathbf{Z}_p^*$, and define $g = \hat{g}^2 \bmod p$; $g$ generates a subgroup denoted $\langle g \rangle$.

In the **Diffie-Hellman protocol**, Alice selects an exponent $a \leftarrow \$(\mathbf{Z}_{p-1})$ and sends $x \leftarrow g^a \bmod p$ to Bob. Alice selects $b \leftarrow \$(\mathbf{Z}_{p-1})$ and sends $y \leftarrow g^b \bmod p$ to Bob. Alice and Bob then individually calculate the shared "key" $z = g^{ab} \bmod p$. (Alice uses $z \leftarrow y^a$ and Bob uses $z \leftarrow x^b$.)

Define the **Diffie-Hellman distribution** $D_p$ as the triple of random variables $(x, y, z)$ obtained from an execution of the DH protocol by honest parties.

The **Decision Diffie-Hellman Assumption** (DDHA) can be described as follows:

> (DDHA) Let $p$ be a safe prime and $g$ a subgroup generator selected as described above. Then $D_p$ is computationally indistinguishable from $(\$(\langle g \rangle), \$(\langle g \rangle), \$(\langle g \rangle))$.

Note that without the precaution of moving to a subgroup, typical Diffie-Hellman triples can be distinguished from three random elements. The quadratic residuosity of $g^{ab}$ can be deduced from that of $g^a$ and $g^b$, hence a random element would be distinguishable from $g^{ab}$.

## 3 Solution Employing Diffie-Hellman

By Crépeau's reductions, it suffices to implement $\binom{2}{1}$OT [C87]. Alice and Bob attempt to set up a valid $\binom{2}{1}$OT execution on random bits, as in [B95b]: if successful, they can later apply this execution to the desired input bits.

[6] $\hat{i}$ is a player in the specification protocol and is unaware of messages being passed in a given implementation. In particular, $\hat{A}$ knows only its input $b$ (and its message to OT), and $\hat{B}$ knows only its message from OT.

The attempt consists of four invocations of the Diffie-Hellman key-exchange protocol [DH76], some of which are "garbled" according to Beaver's approach [B97]. If an appropriate invocation remains ungarbled, then Alice and Bob have established a valid $\binom{2}{1}$OT execution, otherwise they must try again.

### 3.1 Honest Players

Assuming initially that neither Alice nor Bob misbehaves, a simple overview is possible. Essentially, Alice encodes bits $b_0$ and $b_1$ using a $2 \times 2$ table $\alpha_{ij}$ of bits, where $\alpha_{ij} = 0$ iff $b_i = j$. Bob encodes a choice $c$ and mask $m$ using a table $\beta_{ij}$, where $\beta_{cm} = 0$ and all other values are 1. They engage in four DH executions, some of which are garbled. Alice "garbles" whenever $\alpha_{ij} = 1$ and Bob "garbles" whenever $\beta_{ij} = 1$. Bob can detect when they both left instance $cm$ ungarbled, in which case $\alpha_{cm} = 0$, hence $b_c = m$. Otherwise, Bob requests a retry.

The "garbling" of the Diffie-Hellman protocol occurs in one of two ways. Instead of choosing an exponent $e$ and computing $r = g^e$, a player can choose $r$ directly without knowing its discrete logarithm. (Thus, the player will be unable to calculate or verify the final DH key, $g^{ab}$.) Second, a player can garble $g^{ab}$ by likewise choosing a uniformly random residue whose discrete logarithm is unknown. In particular, define the following random variables, which either report a deterministic output or produce a uniform, garbled distribution:

$$G(\sigma, s) = \begin{cases} \$(\mathbf{Z}_p^*) & \text{if } \sigma = 0 \\ g^s \bmod p & \text{if } \sigma = 1 \end{cases}$$

$$G(\sigma, s, r) = \begin{cases} (\$(\mathbf{Z}_p^*), \$(\mathbf{Z}_p^*)) & \text{if } \sigma = 0 \\ (g^s \bmod p, r^s \bmod p) & \text{if } \sigma = 1 \end{cases}$$

The first version is for the player who sends out the initial DH message (Alice, in the original DH protocol; but this will vary below), having made choice $\sigma$ whether to garble or not. The second version is for the player who responds, having made his own choice $\sigma$ about whether to garble or not.

Fig. 1 describes the details of the protocol.

*Why Garble?* Recall that the simulator $\mathcal{S}$ plays the hand of an honest player (within the proposed OT protocol) when it constructs an environment for the adversary. But $\mathcal{S}$ can play that hand dishonestly (for the desired purpose of deceiving an attacker, after all!), by always producing ungarbled instances. By withholding suitable exponents, $\mathcal{S}$ can nevertheless make any ungarbled instance look garbled, since no computationally-bounded judge can detect the difference between four DH triples (fake distribution) and four triples of which one is DH and three are wholly random (real distribution). This remains true even when the adversary obtains all information that Alice and Bob would hold; $\mathcal{S}$ keeps the logarithms up his sleeve.

### 3.2 Malicious Players

Although these techniques are strongly motivated by the application of DH to adaptively-secure encryption in [B97], note that they suffice only for the case of honest players. Malicious misbehavior is of little concern in encryption (*i.e.* it can be handled trivially), where the sender generally has little to gain by causing the receiver to accept nonsense messages. Here, however, both sender and receiver have something to gain by misbehaving.

Protecting against malicious behavior will consist of two parts: (1) using committal to enable suitable random number generation; (2) using zero-knowledge proofs of knowledge (ZKPK's) to extract effective values and ensure compliance with the rules (*cf.* [GMW86, TW87]).

The central problem with this "obvious" cryptographic solution is that the commitments and ZKPK's might defeat the simulator's ability to provide equivocal facsimiles. Thus, the tricks of [B97] are insufficient, by themselves, to achieve our desired goal.

By [B95a], however, it suffices to employ committals that are weakly content-equivocable (a.k.a. *chameleon* [BCC88], a.k.a. *trapdoor* [FS90b]). That is, the "receiver" should be able to "open" the committed bits to 0 or to 1, using knowledge held by the receiver.

Brassard, Chaum and Crépeau provide a discrete-logarithm-based implementation of chameleon blobs [BCC88]. This commitment scheme enables the simulator to extract the effective $\alpha_{ij}/\beta_{ij}$ values used by the adversary.

To complete the discrete-logarithm-based solution, we add the following straightforward complications (*cf.* [GMW87]). The random values $a_{ij}$ and $b_{ij}$ are constructed from precursors: $a_{ij} \leftarrow a'_{ij} + \Delta a_{ij}$; $b_{ij} \leftarrow b'_{ij} + \Delta b_{ij}$. Alice commits to $a'_{ij}$ and $\Delta b_{ij}$, and Bob commits conversely. They then reveal the $\Delta a_{ij}$, $\Delta b_{ij}$ values and proceed as before. (The guaranteed randomization of the $\alpha_{ij}$'s and $\beta_{ij}$'s is similar.) Each party must then give a ZKPK that they used the proper $a_{ij}$, $\alpha_{ij}$, or $b_{ij}$, $\beta_{ij}$ value. Bob must give a ZKPK that $x_{cm}^{b_{cm}} == z_{cm}$.

## 4 Proving Security

Recall that $\mathcal{S}$ must simultaneously create a fake environment for `Adv` while "attacking" an execution of the ideal specification.

*Actions in the Ideal Setting.* When $\mathcal{S}$ engages in an extraction of knowledge from `Adv` that fails, $\mathcal{S}$ then deliberately aborts the ideal protocol. (This reflects a malicious adversary's rightful ability to stop participating.) When the extraction succeeds, $\mathcal{S}$ uses the value on behalf of the corrupt $\hat{A}$ or $\hat{B}$ in the ideal protocol.

### 4.1 Equivocation

We first sketch how the weak equivocation properties are satisfied when Alice or Bob are initially corrupted, and then discuss strong equivocation.

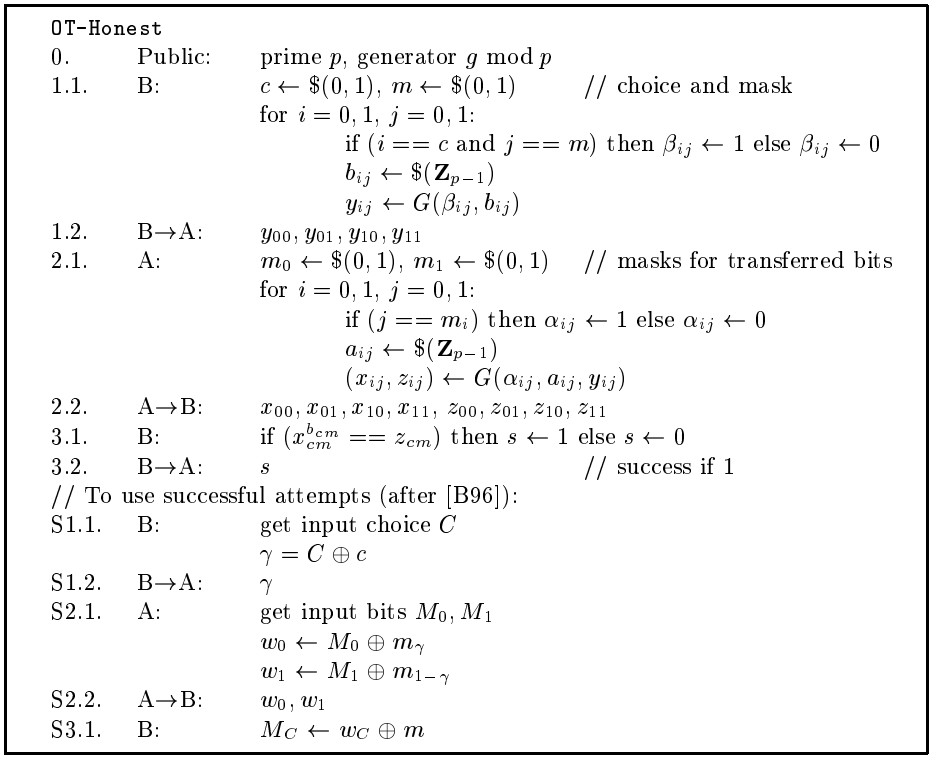


**Fig. 1.** Adaptively secure chosen-1/2-OT, for honest players.

*Weak Result Equivocation.* Assume that Alice is passively corrupted. To simulate Bob, generate $\hat{b}_{ij} \leftarrow \$(\mathbf{Z}_{p-1})$, but do not choose the $\beta_{ij}$'s yet. Set $y_{ij} \leftarrow g^{\hat{b}_{ij}}$, and hand these values to adversary `Adv`.

Extract `Adv`'s choices for $\alpha_{ij}$ from its proof of knowledge. Select $s \leftarrow \$(0,1)$.

If the attempt is to fail ($s = 0$), then choose the $\beta_{ij}$'s conditioned on failure. In particular, let $m_0, m_1$ be such that $\alpha_{0m_0} = \alpha_{1m_1} = 1$.[7] Set $c \leftarrow \$(0,1)$ and take $m \leftarrow 1 - m_c$, which enforces a failure. Set $\beta_{cm} \leftarrow 1$ and $\beta_{ij} \leftarrow 0$ for $(i,j) \neq (c,m)$. Set $b_{cm} \leftarrow \hat{b}_{cm}$ (which conveniently makes $y_{cm} == g^{b_{cm}}$) and $b_{ij} \leftarrow y_{ij}$ for $(i,j) \neq (c,m)$. Thus, even though the three $b_{ij}$ values were chosen with known discrete logarithms (*i.e.* known to the simulator), it appears as though they were chosen directly at random.

If the attempt is to succeed ($s = 1$), then we must enable result-equivocation. The nontrivial case occurs when Bob is corrupted after a successful transmission. (The analysis is similar, indeed trivial, if no bit has been transmitted.) The values $w_0, w_1$ are obtained from Alice. Because Bob is now corrupt, the simulator is

[7] Actually, $m_0, m_1$ may be read directly from the honest machine. Otherwise, they are calculated from the extracted $\alpha_{ij}$ values.

entitled to learn the choice $C$ that Bob made, along with the transmitted bit $M_C$. Set $c \leftarrow C \oplus \gamma$ and $m_c \leftarrow M_C \oplus w_C$. Set $\beta_{cm} \leftarrow 1$ and $\beta_{ij} \leftarrow 0$ for $(i,j) \neq (c,m)$. Set $b_{cm} \leftarrow \hat{b}_{cm}$ (which conveniently makes $y_{cm} == g^{b_{cm}}$) and $b_{ij} \leftarrow y_{ij}$ for $(i,j) \neq (c,m)$. Again, even though the three $b_{ij}$ values were chosen with known discrete logarithms, it appears as though they were chosen directly at random.

*Weak Content Equivocation.* Assume that Bob is passively corrupted. Extract $c, m$ such that $\beta_{cm} = 1$. To simulate Alice, simply follow Alice's program, except for the calculation of $(x_{ij}, z_{ij})$. Instead of using $(x_{ij}, z_{ij}) \leftarrow G(\alpha_{ij}, a_{ij}, y_{ij})$, set $(x_{ij}, z_{ij}) \leftarrow (g^{\hat{a}_{ij}}, y_{ij}^{\hat{a}_{ij}})$, for randomly selected $\hat{a}_{ij} \leftarrow \$(\mathbf{Z}_{p-1})$.

In case of failure ($\alpha_{cm} = 0$), simply withhold the known discrete logarithms. That is, set $a_{0,m_0} \leftarrow \hat{a}_{0,m_0}$, $a_{0,1-m_0} \leftarrow x_{0,1-m_0}$, $a_{1,m_1} \leftarrow \hat{a}_{1,m_1}$, $a_{1,1-m_1} \leftarrow x_{1,1-m_1}$.

In case of success ($\alpha_{cm} = 1$), withhold one of the discrete logarithms by setting $a_{cm} \leftarrow \hat{a}_{cm}$, $a_{c,1-m} \leftarrow x_{c,1-m}$. (The other pair remains "indeterminate" for now, so the simulator can withhold the discrete logarithm of either member of the pair, thereby effectively reversing the unknown bit.) Obtain Bob's final choice $C$ and the value $M_C$ he is entitled to learn. Set $w_C \leftarrow M_C \oplus m$ but $w_{1-C} \leftarrow \$(0,1)$ (this corresponds to the masked, unchosen bit $M_{1-C}$). When Alice is later corrupted and bit $M_{1-C}$ is obtained, equivocate $w_{1-C}$ as follows. Calculate $m_{1-c} \leftarrow w_{1-C} \oplus M_{1-C}$, and set $\alpha_{1-c,m_{1-c}} \leftarrow 1$, $\alpha_{1-c,1-m_{1-c}} \leftarrow 0$, Withhold a second discrete logarithm by taking $a_{1-c,m_{1-c}} \leftarrow \hat{a}_{1-c,m_{1-c}}$, $a_{1-c,1-m_{1-c}} \leftarrow x_{1-c,1-m_{1-c}}$.

*Strong Equivocation.* Note that in the absence of corruptions, $\mathcal{S}$'s calculations and "public traffic" will be consistent with the steps described above for both Alice and Bob. Thus, until `Adv` makes its first corruption request, $\mathcal{S}$ follows the steps described above for both Alice and Bob. If Alice is corrupted first, $\mathcal{S}$ follows the weak R.E. steps to create Alice's view, then continues with the weak C.E. steps. If Bob is corrupted first, the converse programs are followed.

### 4.2 Reduction to Diffie-Hellman (DDHA)

The distribution that `Adv` obtains by interacting with the simulator differs from that obtained in a regular execution in precisely one way: for certain triples $(x_{ij}, y_{ij}, z_{ij})$, the fake distribution follows the correlated Diffie-Hellman distribution $D_p$, whereas the "real-life" distribution contains three fully independent random variables.

These triples occur only in successful attempts, and only on indices where *neither* Alice *nor* Bob has chosen to know the discrete logarithm. That is, neither $\log_g x_{ij}$ nor $\log_g y_{ij}$ is known to Alice or Bob, and in particular, `Adv` *never* learns them as a result of corrupting Alice or Bob.

Straightforward arguments (see [B97], for example) show that distinguishing the simulator's faked view from an actual view would enable distinguishing $D_p$ from independently-random triples, violating the DDHA.

Note in particular that this is why failed attempts must be fully discarded.[8] If the protocol were changed to capitalize on mismatched attempts (intuitively, Alice's index choice remains secret and known only to the two of them even in case of mismatch!) the simulator proof would fail. For instance, if Bob knows $\log_g x_{ij}$, the simulation would be detectably fake, because $z_{ij} = y_{ij}^{\log_g x_{ij}}$ would hold in the simulated cases.

## 5 Generalizations and Applications

To use other intractability assumptions, such as RSA or factoring, a suitable key-exchange construction suffices. In particular, the dense secure public-key cryptosystems of DeSantis and Persiano are appropriate [DP92].

*General Assumptions.* A more general construction (*cf.* [DP92, CFGN96]) employs one-way trapdoor permutation families with the property that permutations can also be generated (indistinguishably) without simultaneously generating a trapdoor. Two modifications are needed.

First, in the honest OT protocol, Alice and Bob respectively choose and report four permutations, $f_{ij}$, $g_{ij}$. For garbled channels, each generates the permutation without the trapdoor, and sends random numbers. For ungarbled channels, each chooses an accompanying trapdoor. Bob sends $y_{ij} \leftarrow f_{ij}(b_{ij})$; Alice returns $z_{ij} \leftarrow g_{ij}(f_{ij}^{-1}(y_{ij}))$. Note that $z_{ij} = b_{ij}$ precisely when Alice and Bob garble the same channel.

Second, malicious behavior is again resisted through ZKPK's; the general constructions of Feige and Shamir [FS90a, FS90b] provide the needed trapdoor/chameleon/weak-equivocation property.

*Third Parties.* When third parties are available – as in the case of a multiparty computation – one-way trapdoor permutations without the extra oblivious-generation property can be used. These third parties need not be individually trusted, but at least one of them must remain honest. We also require a broadcast channel.

As in the clever construction used in [CFGN96] for encryption, the ultimate permutations are composed of permutations generated by the third parties, who allow Bob and Alice to learn trapdoors selectively. Receiver Bob learns one of four (rather than of two) trapdoors by way of EGL/GMW $\binom{2}{1}$OT. Unlike [CFGN96], however, Sender Alice also learns trapdoors: one from each of two pairs. The remainder of the protocol follows the Diffie-Hellman solution proposed in this work. Malicious behavior is avoided through network-based commitment and proofs, which do not require the set of faults to be a minority.

[8] Note: this does not mean *erased*; the simulator is choosing what to place in the details of a *full* player history. To simulate a failed attempt, the simulator behaves *perfectly* accurately on behalf of any corrupt party/parties. Only the *successful* attempts suffer any mathematical (but still negligible) distinction from the real-life distribution.

Although this approach relies on a weaker assumption, it is far less efficient, requiring network-wide interaction for each transfer. Note that applying [CFGN96] to encrypt an information-theoretically secure OT protocol using [BGW88, CCD88] would also suffice, but it requires that faults be a strict minority.

*Faulty Majority.* While adaptively-secure encryption enables one to construct adaptively-secure multiparty protocols when there is a faulty minority [BH92, CFGN96], it does not directly suffice when there is a faulty majority. In separate work [B96b], we show that the tools described in this paper make possible the construction of a provably fair and secure protocol for multiparty function evaluation even in the presence of a majority of faults, using techniques of Beaver, Goldwasser and Levin [BG89, GL90].

## References


[B91] D. Beaver. "Foundations of Secure Interactive Computing." *Advances in Cryptology – Crypto '91 Proceedings,* Springer–Verlag LNCS 576, 1992, 377–391.

[B92] D. Beaver. "How to Break a 'Secure' Oblivious Transfer Protocol." *Advances in Cryptology – Eurocrypt '92 Proceedings,* Springer–Verlag LNCS 658, 1993, 285–296.

[B95a] D. Beaver. "Adaptive Zero Knowledge and Computational Equivocation." *Proceedings of the* $28^{th}$ *STOC,* ACM, 1996, 629–638.

[B95b] D. Beaver. "Precomputing Oblivious Transfer." *Advances in Cryptology – Crypto '95 Proceedings,* Springer–Verlag LNCS 963, 1995, 97–109.

[B96] D. Beaver. "Equivocable Oblivious Transfer." *Advances in Cryptology – Eurocrypt '96 Proceedings,* Springer–Verlag LNCS 1070, 1996, 1996, 119–130.

[B96b] D. Beaver. "Fair and Adaptively Secure Computation with Faulty Majority." Manuscript, 1996, to be submitted.

[B97] D. Beaver. "Plug-And-Play Encryption." *Advances in Cryptology – Crypto '97 Proceedings,* Springer–Verlag LNCS 1294, 1997, 1997.

[BG89] D. Beaver, S. Goldwasser. "Multiparty Computation with Faulty Majority." *Proceedings of the* $30^{th}$ *FOCS,* IEEE, 1989, 468–473.

[BH92] D. Beaver, S. Haber. "Cryptographic Protocols Provably Secure Against Dynamic Adversaries." *Advances in Cryptology – Eurocrypt '92 Proceedings,* Springer–Verlag LNCS 658, 1993, 307–323.

[BGW88] 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.

[BM89] M. Bellare, S. Micali. "Non-Interactive Oblivious Transfer and Applications." *Advances in Cryptology – Crypto '89 Proceedings,* Springer–Verlag LNCS 435, 1990, 547–557.

[BCR86a] G. Brassard, C. Crépeau, J. Robert. "All or Nothing Disclosure of Secrets." *Advances in Cryptology – Crypto '86 Proceedings,* Springer–Verlag LNCS 263, 1987, 234–238.

[BCR86b] G. Brassard, C. Crépeau, J. Robert. "Information Theoretic Reductions among Disclosure Problems." *Proceedings of the* $27^{th}$ *FOCS,* IEEE, 1986, 168–173.

[BCC88] G. Brassard, D. Chaum, C. Crépeau. "Minimum Disclosure Proofs of Knowledge." *J. Comput. Systems Sci.* **37**, 1988, 156–189.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, M. Naor. "Adaptively Secure Multiparty Computation." *Proceedings of the* $28^{th}$ *STOC,* ACM, 1996, 639–648.

[CCD88] D. Chaum, C. Crépeau, I. Damgrd. "Multiparty Unconditionally Secure Protocols." *Proceedings of the* $20^{th}$ *STOC,* ACM, 1988, 11–19.

[C87] C. Crépeau. "Equivalence Between Two Flavours of Oblivious Transfers." *Advances in Cryptology – Crypto '87 Proceedings,* Springer–Verlag LNCS 293, 1988, 350–354.

[dB91] B. den Boer. "Oblivious Transfer Protecting Secrecy." *Advances in Cryptology – Eurocrypt '91 Proceedings,* Springer–Verlag LNCS 547, 1991, 31–45.

[DP92] A. DeSantis, G. Persiano. "Zero-Knowledge Proofs of Knowledge Without Interaction." *Proceedings of the* $33^{rd}$ *FOCS,* IEEE, 1992, 427–436.

[DH76] W. Diffie, M. Hellman. "New Directions in Cryptography." *IEEE Transactions on Information Theory* **IT-22**, November 1976, 644–654.

[EGL85] S. Even, O. Goldreich, A. Lempel. "A Randomized Protocol for Signing Contracts." *Comm. of the ACM* **28**:6, 1985, 637–647. (Early version: *Proceedings of Crypto 1982,* Springer–Verlag, 1983, 205–210.)

[F88] P. Feldman. Manuscript, 1988. (Personal communication, Cynthia Dwork.)

[FS90a] U. Feige, A. Shamir. "Witness Indistinguishable and Witness Hiding Proofs." *Proceedings of the* $22^{nd}$ *STOC,* ACM, 1990, 416–426.

[FS90b] U. Feige, A. Shamir. "Zero Knowledge Proofs of Knowledge in Two Rounds." *Advances in Cryptology – Crypto '89 Proceedings,* Springer–Verlag LNCS 435, 1990, 1990, 526–544.

[GMW86] 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.

[GMW87] 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.

[GL90] S. Goldwasser, L. Levin. "Fair Computation of General Functions in Presence of Immoral Majority." *Proceedings of Crypto 1990.*

[GM84] S. Goldwasser, S. Micali. "Probabilistic Encryption." *J. Comput. Systems Sci.* **28**, 1984, 270–299.

[GMR89] S. Goldwasser, S. Micali, C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems." *SIAM J. on Computing* **18**:1, 1989, 186–208.

[HL90] L. Harn, H. Lin. "Noninteractive Oblivious Transfer." *Electronics Letters* **26**:10, May 1990, 635–636.

[K88] J. Kilian. "Founding Cryptography on Oblivious Transfer." *Proceedings of the* $20^{th}$ *STOC,* ACM, 1988, 20–29.

[KMO89] J. Kilian, S. Micali, R. Ostrovsky. "Minimum Resource Zero-Knowledge Proofs." *Proceedings of the* $30^{th}$ *FOCS,* IEEE, 1989, 1989, 474–479.

[MR91] S. Micali, P. Rogaway. "Secure Computation." *Advances in Cryptology – Crypto '91 Proceedings,* Springer–Verlag LNCS 576, 1992, 392–404.

[R81] M.O. Rabin. "How to Exchange Secrets by Oblivious Transfer." TR-81, Harvard, 1981.

[RSA78] R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM* **21**:2, 1978, 120–126.

[TW87] M. Tompa, H. Woll. "Random Self-Reducibility and Zero-Knowledge Proofs of Possession of Information." *Proceedings of the* $28^{th}$ *FOCS,* IEEE, 1987, 472–482.

[Yao82a] A. Yao. "Protocols for Secure Computations." *Proceedings of the* $23^{rd}$ *FOCS,* IEEE, 1982, 160–164.

[Yao82b] A. Yao. "Theory and Applications of Trapdoor Functions." *Proceedings of the* $23^{rd}$ *FOCS,* IEEE, 1982, 80–91.

[Yao86] A. Yao. "How to Generate and Exchange Secrets." *Proceedings of the* $27^{th}$ *FOCS,* IEEE, 1986, 162–167.